Table of Contents

1 Introduction

This thesis focuses on improvement of the communication between components of a system which is integrated on a single chip. In this chapter, we introduce the System on Chip and different methods for connecting components of the system. We will also introduce the area and problems handled in the thesis.

1.1 System on Chip

A core is an individual component that has a particular, often advanced, functionality. Today, it is possible to integrate a large number of cores (e.g. general purpose processors, embedded memories, DSP cores, FPGA blocks, I/O blocks, ASIC blocks, etc.) on a single silicon chip. Integrating the entire system on one chip reduces the size and increases the performance of electronic systems. For example, STMicroelectronics announced **FLI7540**, a new **TV System-on-Chip** lately. 1700+ DMIPS CPU with 256 KBytes of Level 2 cache offers a high performance TV [13]. These independent blocks can be of unequal size. Interconnecting pre-designed cores (resources) or IP-cores (Intellectual Property) becomes harder and harder by increasing the number of cores. Reducing design complexity and power consumption are some of the most important issues for SoC design [3].

Figure 1-1.FLI7540 - Digital TV System-on-Chip and Video-Enhancement IC (Photo from ST Website)

1.2 Options for Interconnecting the Cores in a SoC

Choosing a proper way for interconnecting the cores in a SoC design is an important step. A good interconnection reduces manufacturing cost and complexity. It decreases energy consumption as well. It also improves the performance of a system, for instance by decreasing communication delay between the cores [1][2].

1.2.1 Point to Point Connections

The first option for interconnecting the cores was to use direct point to point connections between cores, as illustrated in *Figure 1-2*.

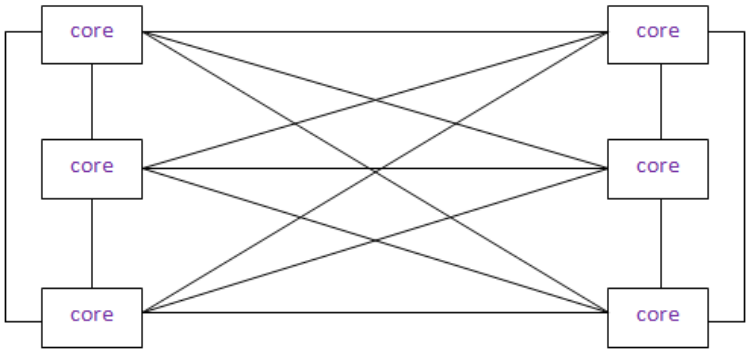


Figure 1-2.An Illustration of Direct Interconnect for On-Chip Communication

Main problems of this type of interconnection are that, it requires a lot of wires, I/O pins and big routing area and the system is not scalable. It is too hard to reuse this system and the routing resources are not utilized very well.

1.2.2 Bus-Based System on Chip

The basic idea is to share wires to connect several cores. Many of the existing SoCs are bus-based [3].

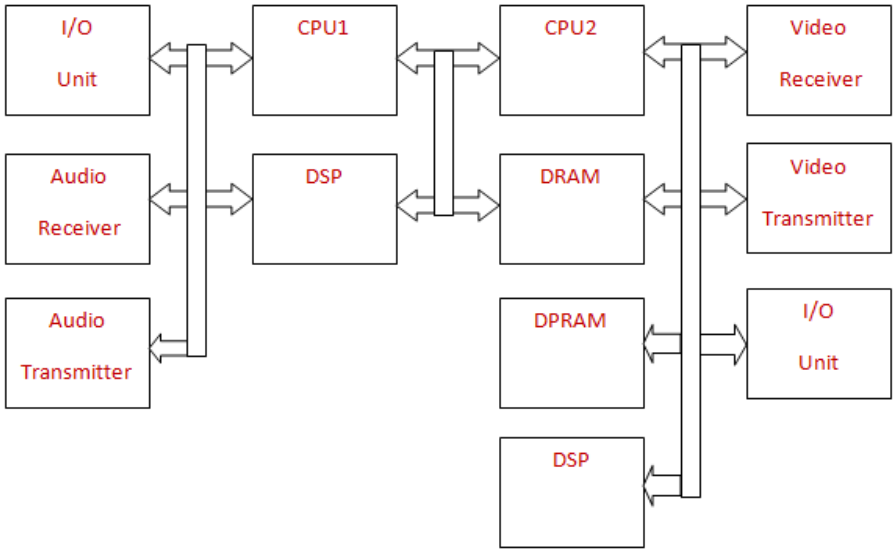


Figure 1-3.Bus-based System on Chip

A bus arbiter chooses a component to be granted bus access. Communication is fast, but bus access-time is increased by increasing the number of users. Speed of bus and the delay depend on the length of the bus and the longest physical distance between two resources, respectively. One pair can communicate to each other at a time and the cores compete for the bus. The hierarchical, segmented and pipelined buses are some of the advanced ways for bus-based systems. Shared buses may be suitable for systems with less than 8 resources and when the communication requirement is low on average, few resources are sources and the majority of resources are destinations [4].

1.2.3 Network on Chip (NoC)

Network on Chip (NoC) has emerged as a dominant paradigm for synthesis of multi-core SoCs. As illustrated in *Figure 1-4*, in NoC paradigm, cores are connected to each other through a network of routers and they communicate among themselves through packet-switched communication. A large number of different NoC architectures have been proposed by different research groups based on this paradigm [2][4]. Network topology and routing algorithm are the two most important aspects which distinguish various proposed NoC architectures. Router is the most important component for design of the communication back-bone of a NoC system (like any other network). In a packet switched network, the functionality of the router is to forward an incoming packet to the destination if it is directly connected to it, or to forward the packet to another router connected to it. The protocols used in NoC are generally simplified versions of general communication protocols used in data networks. In the context of NoC, scarcity of silicon resources requires that the router design should be as simple as possible.

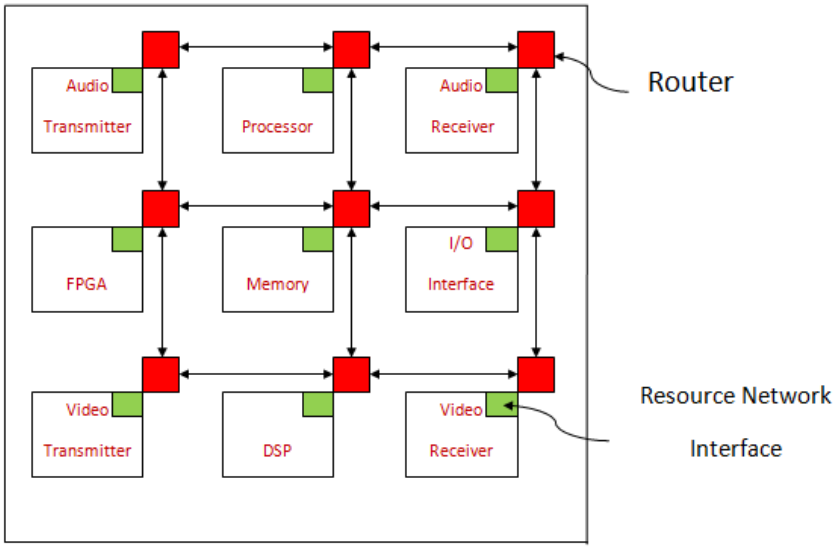


Figure 1-4.NoC-based System on Chip

1.3 Issues in NoC-Based SoC Design

1.3.1 Topology

Network topology is defined as the interconnection of various elements (links, nodes, etc.) of a network. Design of NoC router architecture depends upon the network topology. The mesh topology is one of the most common network topologies to use. We use mesh topology in this thesis.

1.3.2 Routing Algorithms

Routing schemes have been classified in several ways in literature. In a scheme called *source routing*, the source node selects the entire path before sending the packet. The major drawback of this approach is that each packet must carry this routing

information, thus increasing the packet size. In addition, the path cannot be changed after the packet has left the source. A more common solution is the use of *distributed routing*. Here a router upon receiving a packet decides, based on the destination address, whether it should be delivered to the local resource or forwarded to a neighboring router. In the latter case, a routing algorithm is invoked (or a routing table is accessed) to determine which neighbor the packet should be sent.

Source routing was not considered suitable for very large and dynamic networks because of the overhead on packet size. But it is likely to have some advantages for small networks with regular topologies, especially with networks having an upper limit on the number of output ports in the routers. Mesh topology NoC is one such network. This will simplify the design of the router since the routing information is directly available in the packet. The overhead may also be reduced since we do not need to carry destination address.

In 2009, a master thesis [4] evaluated the possibility of using source routing for mesh topology NoC platforms and compared its performance with distributed routing. It was shown that source routing has a very good potential for NoC platforms. It was also shown that router design for source routing will be simpler than distributed routing.

1.4 Project Objectives and Tasks

Source routing is not considered scalable and efficient for large networks since the overhead of appending path information in the packet header increases with network size. No efficient solution exists in literature regarding this problem so far.

The main objective of this project will be to develop a new routing scheme, called Junction Based Routing, which will make source routing in large NoCs systematic, scalable and efficient. The goal of the project will be to complete the theory regarding this new idea for routing, work out its implementation details and evaluate and compare the new algorithm with existing routing algorithms. The evaluation will be simulation based.

The project consists of the following tasks:

i. Analytical analysis of routing algorithms and completing the theory regarding the idea of a new routing algorithm. Development of the new routing algorithm and identifying the contexts in which it will work better than other algorithms. Finding out the advantages and disadvantages of the new routing algorithm and the basic hardware for implementing the mentioned routing algorithm.

ii. Computing all paths from sources to destinations and select one path for each communicating pair based on the best link load distribution. An efficient encoding scheme should be developed to encode paths for this routing algorithm.

iii. Development of a simulator (or modification of an existing simulator) to evaluate the new technique.

iv. Evaluation of the new algorithm and its comparison with conventional source routing algorithm.

1.5 Thesis Layout

In first chapter, we described integrating a system on one silicon chip and primitive connection methods. Chapter 2 presents basic knowledge in network on chip approach. Third chapter defines the concepts in the new technique, called Junction-Based Routing (JBR). There are many interesting issues related to this technique that are discussed and solved in Chapter 4. Path computation for efficient deadlock free routing is the most important problem. A simulator has been developed to evaluate the performance of JBR that is explained in Chapter 5. Chapter 6 gives conclusions and proposals of future works.

2 Theoretical Background

Basic concepts related to NoC are described in this chapter. Routing algorithms are discussed in more details due to their important role in the performance of a network. This chapter also presents some of the parameters used to evaluate the performance of a network.

2.1 Network on Chip

Shared buses and dedicated wires can be used to connect only a few numbers of cores. The other disadvantages are low scalability and low reusability for new SoCs. They are inefficient for high communication performance. In the year of 2000 a new paradigm, called Network on Chip (NoC), was proposed for synthesis of multi-core SoCs. As illustrated in *Figure 2-1*, in NoC paradigm, cores communicate to each other through a network of routers. The pre-routed wires reduce the design complexity and make the testing and verifying of the system easier [1][2].

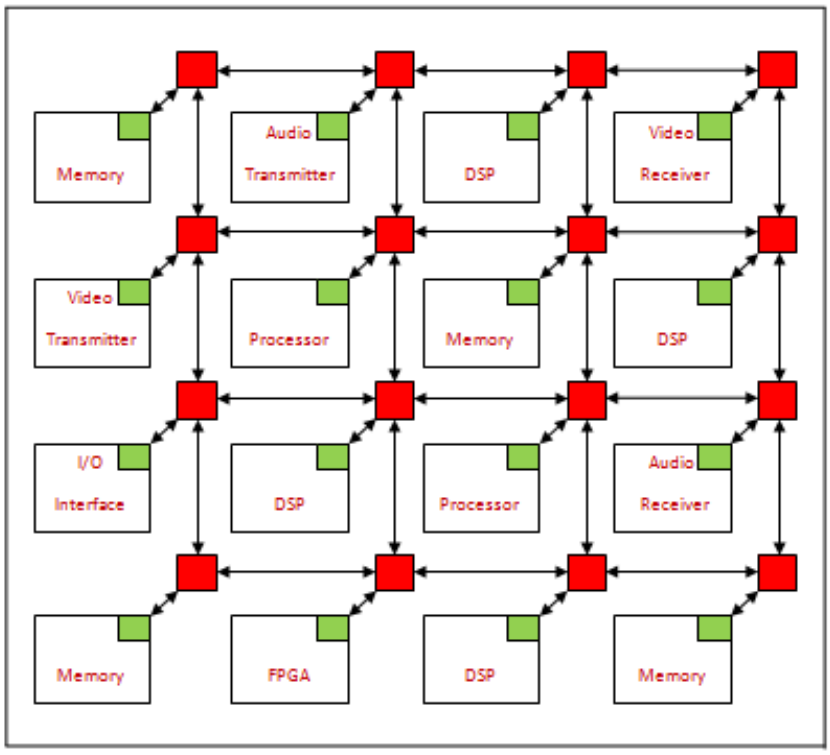


Figure 2-1.NoC-based System on Chip

In the network-based SoC, each resource is connected to a router. Data is transferred from source to destination in the packet form. A packet in the network may not reach to the destination that fast, but many pairs can communicate simultaneously. A message is sent to the router connected to the source core and it is forwarded by other network routers to reach to the destination router and the destination core. A NoC-based system is usually considered scalable, because adding a core needs an extra router and some links depending on the topology.

2.2 Terminology of NoC

This section provides a summary of some network communication concepts that are applicable in NoC field.

2.2.1 Network Architecture

NoC uses layered communication. Network architecture is partitioned into physical layer, data link layer, network layer, transport layer and application layer. Each layer is specifying a particular function and can implement tasks autonomously.

Message represents the data to move between cores and is defined in application layer. Message size can be fixed or variable. A message consists of many packets. A packet is a group of bits for independent transfer in the network. It contains all information that is necessary to reach the destination. Packets can have different sizes. Network layer determines the routing of packets through network routers and it is responsible for performing packetization, packet buffering, congestion control, providing quality of service, etc.

A packet may be partitioned into many flits. A Flow Control Digit (Flit) is a group of bits that is defined in data link layer. The flit size is constant. Data link layer is concerned with reliable node to node communication, error detection and correction, flow control, encoding scheme, etc.

The electrical specifications are defined in physical layer. Phit (PHysical transfer digIT) is transferred as a unit across a channel from one router to the next. The phit size is equal to the number of wires between two routers, and thus can be considered as link width.

2.2.2 Direct and Indirect Networks

Networks can be classified into two categories, namely, Direct and Indirect networks. In a Direct Network, each node is switch and the resource, whereas in an In-direct Network, each node is either a switch or a resource as illustrated in *Figure 2-2*.

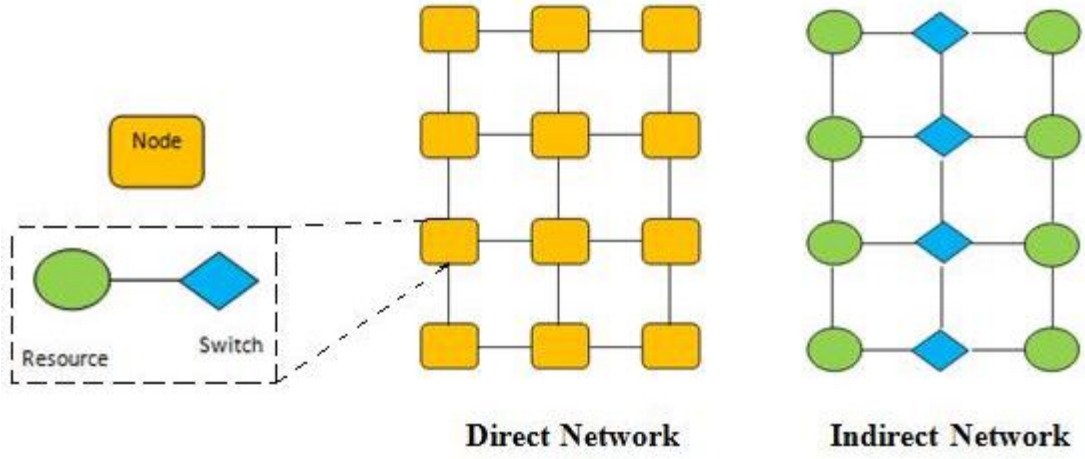


Figure 2-2.Examples of Direct and Indirect networks

2.2.3 Topology

Network topology is defined as the interconnection of various elements (links, nodes, etc.) of a network. Topologies are generally categorized as regular or irregular. Regular topologies have a uniform structure, while irregular topologies can have a heterogeneous structure. *Figure 2-3* shows examples of some topologies proposed for NoC. Two-dimensional mesh topology will be used throughout in this thesis. It is one of the easiest topologies to implement on a silicon die, because of its flat configuration.

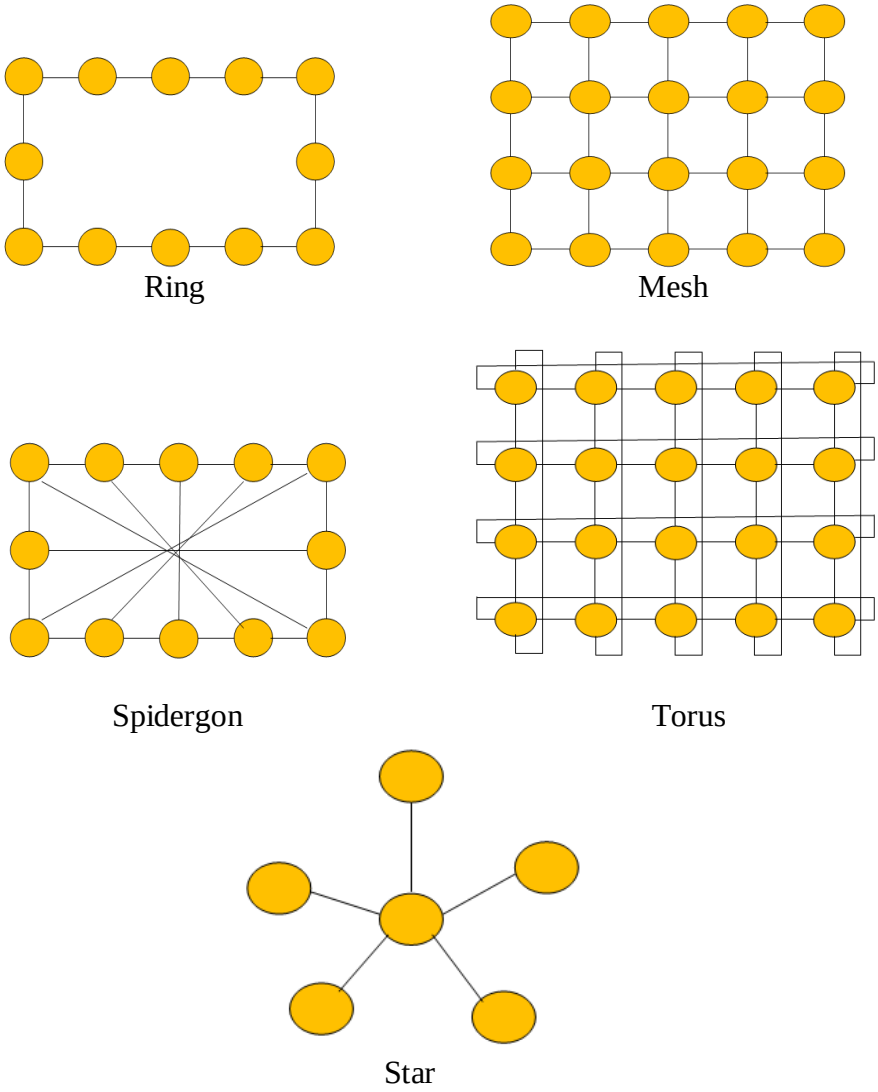


Figure 2-3.Examples of some network topologies

Mesh size given as RxC means the number of node rows is R and the number of node columns is C. Cube and hypercube are also regular topologies similar to mesh.

2.2.4 Network Diameter

Diameter of a network is the maximum of the shortest distance between any pair of nodes in the network. For a MXN Mesh network, the diameter is (M+N-2).

2.2.5 Path

A communication path represents the ordered set of channels between a source and destination node pair and the number of channels specifies the path length. Path diversity describes the number of paths between a pair of nodes. A network with higher path diversity is more fault-tolerant.

2.3 Components of NoC

A NoC consists of three basic building blocks: links (channels), routers (switches) and resource to network interfaces (RNIs). Cores in a NoC-based system on a chip should compete for the shared channels and switches. Resource utilization is one of the challenges in NoC [3].

2.3.1 Router

A router switches an incoming message to an output channel. In distributed routing, the output channel is selected either by looking up a table that is accommodated in the router (table based router) or by running a routing algorithm. In source routing, the path is read from the packet header. Router architecture for mesh topology NoC is illustrated in *Figure 2-4*. The input and output ports are connected through the crossbar which consists of a number of multiplexers and therefore, routing of messages is performed simultaneously when messages are headed for non-conflicting outputs. Arbiter is used if there are several requests for the same output. Commonly packets are buffered before routing. Simple routers are desirable because of their expected lower implementation costs [1][3][5].

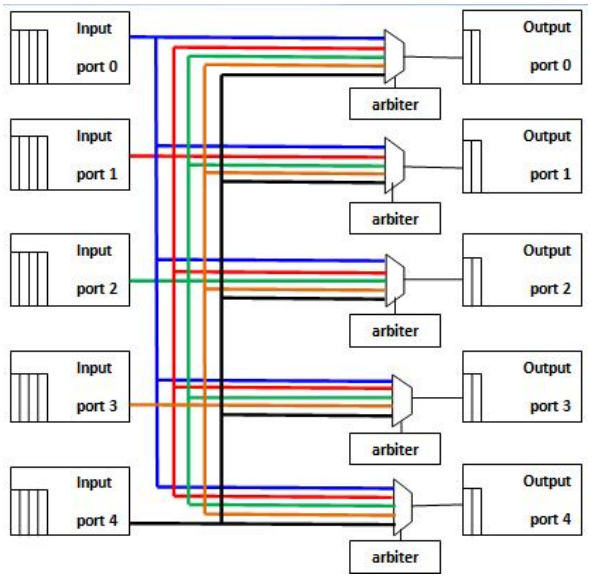


Figure 2-4.Router architecture for mesh topology NoC

2.3.2 Resource Network Interface (RNI)

Each core is connected to a router using a Resource Network Interface (RNI). An RNI and a network card in a PC have the same purpose [8]. An RNI receives messages

(packets) from the source node and performs some services like flitization and adding path information, etc. It also receives data (flits) from a router and performs buffering, deflitization, etc. As illustrated in *Figure 2-5*, an interface is divided into two parts. A resource independent part is reused throughout the network and performs services such as serialization and de-serialization. A resource dependent part depends on I/O, bit-width of data and address bus, control signals, etc. of each resources.

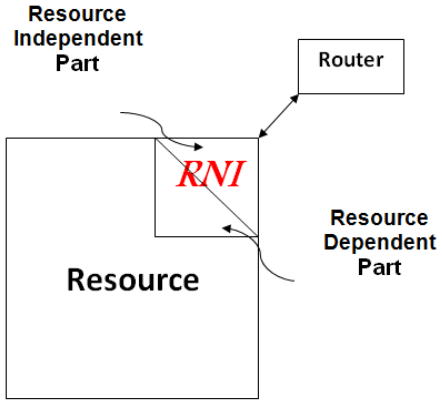


Figure 2-5.Resource-Network Interface

2.4 Switching

The switching method describes how data flows through the NoC. Latency in the network strongly depends on the chosen switching technique [7]. Packet switching and circuit switching are two forms of switching techniques.

2.4.1 Circuit Switching

In circuit switching, an electrical path is set up between a source and a destination for the duration of communication. Therefore, it is not flexible and reactive to traffic. It is helpful for some applications like real time video-processing applications, where dependable exchange of data is needed.

2.4.2 Packet Switching

A packet switch network is a network of switches. The data is exchanged among nodes in packets which consist of header, payload and terminator, as illustrated in *Figure 2-6*.

The packet header holds destination/source address, error detection/correction bits, priority etc. The real data is stored in the packet payload. The packet end is specified using the packet terminator. A packet may flow through many routers before arriving at the destination and network resources are assigned to the packet as it travels towards the destination and thus, routing can be reactive to traffic.

Store and forward, wormhole and cut-through switching are different kinds of packet switching [1][2][3][4].

In store and forward switching, a whole packet is exchanged between switches and the packet is forwarded to the next switch after complete reception. Therefore, delay is high and large buffer is needed.

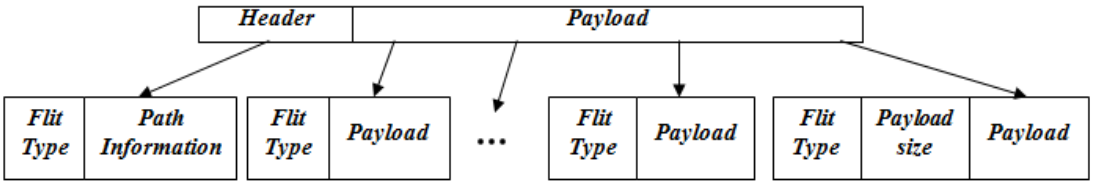


Figure 2-6.Flitization of a packet

In wormhole switching, a packet is partitioned into flits which are transmitted through the network. Therefore, a smaller buffer is needed and the cost and size of a router is decreased. An example of wormhole switching is depicted in *Figure 2-7*. The header flit(s) has the routing information and finds out a path for the packet thus, flits of two packets should not be interleaved at any middle node. The rest of the flits go after the header flit(s) in a pipelined mode and thus, latency is not responsive to the distance between the source and destination. A drawback with wormhole switching is that, if the header flit cannot go further, all the flits are blocked along the path while, they have occupied some of the channels and switches. The other messages are waiting for them and it can cause deadlock [7].

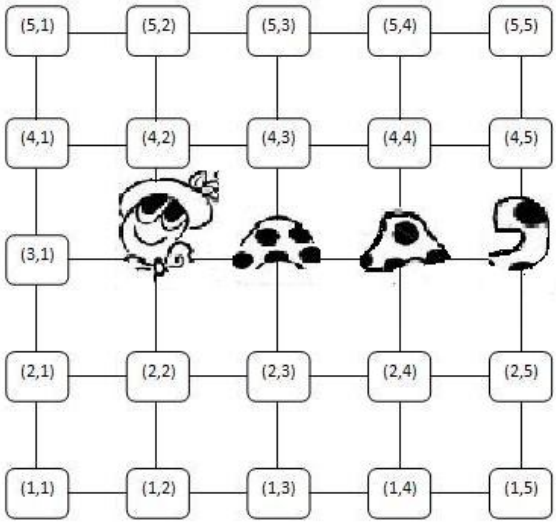


Figure 2-7.An example of wormhole switching

Virtual cut through switching is much like wormhole switching, but each node must be able to store a whole packet. The header flit can go forward and undergo processing while the rest of the flits are still navigating the network and therefore latency and throughput characteristics are close to wormhole switching.

2.5 Buffers and Virtual Channels

Buffers decrease effects of congestion. Input buffers hold received packets (flits) waiting for accessible output ports. Output buffers hold data waiting for accessibility

of next router inputs. Buffers consume lots of energy [5]. FIFO at each router port is a common buffer approach.

A physical channel can be considered as several logically separated channels called virtual channels. Each virtual channel (VC) has its own buffer. For instance, if the packets in Router 1 which are to take a south-ward turn at Router 2 are blocked, then packets which are to take a north turn or go straight can also not move (see *Figure 2-8*). If there were two Virtual Channels, then packets going to north or straight at Router 2 could use the second virtual channel. Therefore latency is reduced and throughput is increased. The most important difficulty is the amount of buffer space that they use. Each virtual channel bandwidth is also reduced [6].

Figure 2-8.Virtual channels

2.6 Routing

Routing is the mechanism that finds out path(s) from a source node to the destination node in the network. The purpose of a routing algorithm is to find these paths. Preferably, a path is chosen such that the overall latency is reduced and the load in the network is balanced thus, the routing algorithm affects the performance of a network. A simple routing algorithm results in minimum circuitry and lower implementation cost of routers [1][2][3][4]. General categorizations of routing algorithms are:

- Source vs. distributed routing
- Deterministic vs. adaptive
- Static vs. dynamic routing
- Minimal vs. non-minimal routing
- Application specific routing

2.6.1 Source vs. Distributed Routing

In source routing, the route information is added to the packet header by source node before sending the packet and cannot be modified after sending the packet. Therefore, switching nodes are simpler but, each packet holds the entire routing information and the packet size gets larger. In distributed routing, a switch determines the output port when a packet arrives. This decision is taken using a routing algorithm or a routing table. Network state is one of the affecting factors of choosing the path.

2.6.2 Deterministic vs. Adaptive Routing

In oblivious routing algorithms, the route from the source to the destination is decided without considering the state of the network traffic. Deterministic routing algorithms

determines a fixed path between the source and the destination and they are oblivious algorithms.

In adaptive routing a number of routes between a source and a destination are specified. One of the routes is chosen by taking into account the state of the network (such as the presence of faulty or congested links). Adaptivity describes the measure of routing flexibility for selecting the paths. In fully adaptive routing, all routes between source and destination are available. In partially adaptive routing, the number of choices is limited at some or all routers. Adaptivity can result in collisions and deadlocks [17].

2.6.3 Static vs. Dynamic Routing

In static routing, the route is not modified after sending a packet. Dynamic routing algorithm determines the paths if path should be altered.

2.6.4 Minimal vs. Non-minimal Routing

A minimal routing algorithm only employs shortest paths. Non-minimal routing algorithm may also use longer distance path and it can often distribute traffic better than a minimal routing algorithm.

2.6.5 Application Specific Routing

A lot of deadlock-free routing algorithms are general purpose. Application specific routing is applied for particular applications or a set of concurrent applications, where we know the set of pairs of cores which exchange data with each other. Application Specific Routing Algorithm (APSRA) is one of them. One method to implement an APSRA is to store a table in every switch which will guide a received flit to an appropriate output channel [10].

2.7 Deadlock-Free Routing Algorithms

2.7.1 Deadlock, Livelock and Starvation

Deadlock is a situation where packets are waiting for each other to free resources (channels and buffers in routers) in a circular chain. Therefore none of them can go towards their destinations.

Livelock is a situation where packets travel in the network without end and never arrive at their destinations. Livelock can be a trouble for non-minimal routing algorithms.

The overhead for resolving deadlocks can be costly, therefore it is usually preferred that routing algorithms are deadlock free, i.e. guarantees that deadlocks cannot happen. There exist several deadlock-free routing algorithms for regular networks. These are quite proficient with regard to cost and performance. *Figure 2-9* illustrate a deadlock situation through an example.

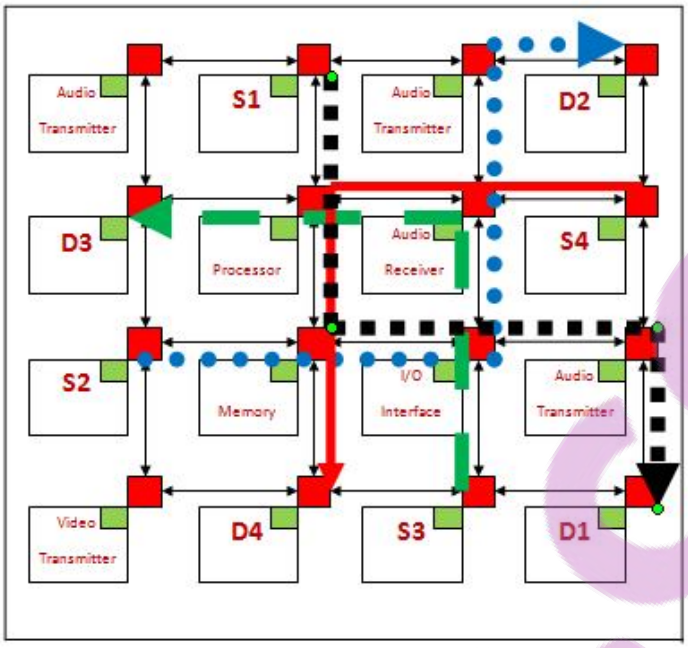


Figure 2-9.An example of deadlock situation

S1 is sending a packet, called packet1, to D1 and packet1 asks for an east turn at the router corresponding to memory block, called Memory node. This packet is stopped by packet 2 that stretches through memory node. Packet 2 asks for a north turn at I/O Interface node but is stopped by packet 3. Packet 3 asks for a west turn but is stopped by packet 4. Packet 4 asks for a south turn at Processor node but is stopped by packet 1.

The packets cannot make progress toward their destinations because of the cyclic dependency among the packets. The network has gone into a condition of deadlock which may be resolved using special methods.

2.7.2 Turn-Model Routing Algorithms

In N-dimensional meshes, deadlock-free routing algorithms can be designed using Turn-model [1][2][3]. As illustrated in *Figure 2-10*, in Turn-model based routing algorithms, some turns are restricted and packets are not allowed to make them in a network. Using channel dependency graphs (CDG) and keeping away from circular communications is a method that is helpful in networks with any kind of topology. [17].

In a 2D mesh network, at least two turns should be banned for a deadlock free routing algorithm [3].

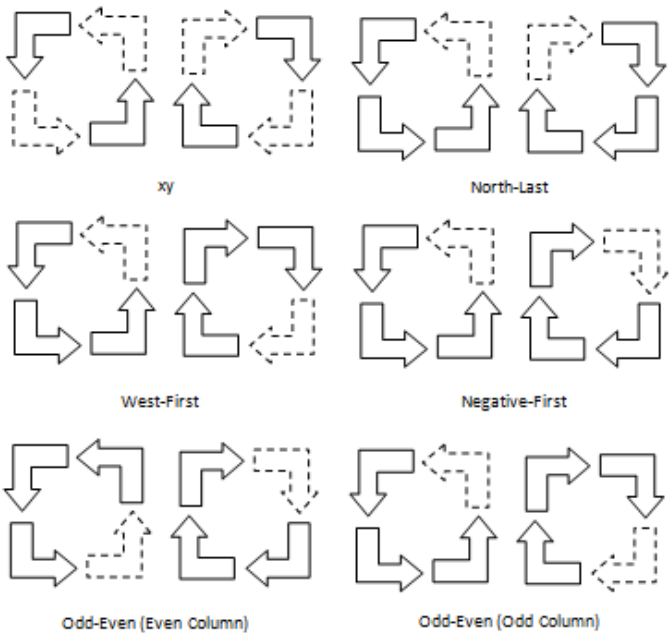


Figure 2-10.Turn model-based routing algorithms for mesh topology NoC

In X-Y routing, if the column of the source and the column of the destination are different, a packet moves along the horizontal axis toward the destination. After that it makes progress to the destination vertically. In *Figure 2-11*, source node (3,1) is communicating with (1,3). The path which is shown using the vector is allowed for sending data from S to D

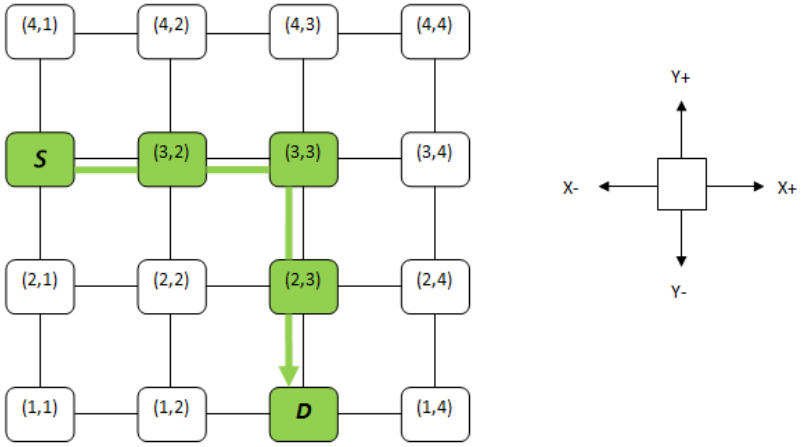


Figure 2-11.Allowed path in XY routing algorithm

West-First is another deadlock free routing algorithm for mesh topology NoC. The West-First routing algorithm is more adaptive in compare with X-Y [3][7]. Therefore several paths are available for the packets to make progress toward their destinations.

Odd-Even routing algorithm is another partially adaptive routing algorithm and has a higher adaptiveness in compared with the other routing algorithms [3][7]. Packets are not allowed to make an East-North or East-South turn at the nodes that are in an even column of a mesh network. North-West or South-West turn is limited at the nodes that are in an odd column (see *Figure 2-10*).

For instance, source node S is sending data to destination node D. Applying Odd-Even routing algorithm, there are three possible paths for sending data from S to D that are depicted in *Figure 2-12.*

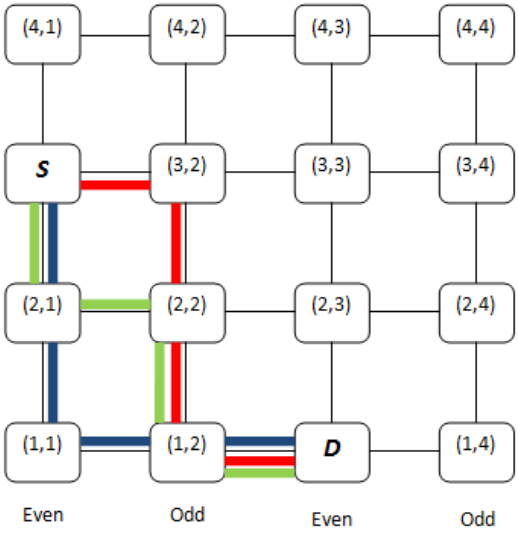


Figure 2-12.Allowed paths in Odd-Even routing algorithm

There are many other deadlock free routing algorithms for mesh topology NoCs [3] [7] [10].

2.8 Evaluation of NoC

2.8.1 Network Simulators

Network simulators are used to model and simulate the behavior of a network and evaluate its performance. Building a hardware prototype for a network is very costly and time consuming. For instance, simulators are used to compare routing algorithms.

Noxim and Network Simulator (NS2) are two simulators commonly used by researchers. Different options for modeling NoCs are SystemC, SDL, C/C++, Java etc. The research group in Jönköping University has developed a specific NoC simulator based on source routing [12]. This simulator will be modified and used in this project. We modeled Junction Based NoC using SDL.

2.8.2 Performance Parameters

Some of the most important parameters that are used in evaluating the performance of NoCs are defined in this sub-section briefly. These parameters are discussed in details in next chapters.

Latency

Network latency presents the required time to transfer n bytes of payload from its source to its destination. Latency consists of routing delay, contention delay, channel occupancy and overhead.

- Routing delay is a function of the distance between a source node and a destination node. It also depends on the routing algorithm that is used.
- A number of bits is required for storing routing information, error detection etc. Channel occupancy depends on these kinds of bits.
- Packets should compete for the shared resources, like channels, in a NoC. There is also some delay due to the waiting time in a switch. Contention delay presents these kinds of delays in a network.
- Packetization at source nodes, de-packetization at destination nodes, synchronization between routers etc. introduces a certain amount of delay in a network, called overhead delay.

Bandwidth

Communication bandwidth is the amount of data that can be moved using a communication link in a unit time period.

Throughput

Throughput is the total number of received packets by the destinations per time unit.

Packet Loss

Packet loss happens when one or more packets do not reach their destination due to the error introduced by the network, the contention for network link or lack of buffer space etc.

Link Load

The offered load is the amount of traffic that is injected by the cores into the network. Network Load is defined as the measure of the real communication traffic in the network, regarding maximum possible traffic. Maximum traffic rate is calculated using the following formula:

*Maximum traffic rate = Number of links in the network * Link bandwidth*

Other load measures used in NoC are actual traffic rate, average load, traffic load etc. Link load is the amount of data flowing through the link in each direction provided the links are considered bidirectional.

Fault Tolerance

Different kinds of faults can occur in the network and fault tolerance describes the capability of a network and a routing algorithm to still route data in this situation.

In-order Packet Delivery

In-order delivery is the delivery of packets in the same order that they were sent. For example, packets following different paths can cause out-of-order delivery.

Power consumed by the routers and their size are important parameters for evaluating routing algorithms. Small and simple routers are desirable.

2.8.3 Traffic Types

For evaluating NoC using a simulator, data is transmitted into the network in different ways and performance values are evaluated regarding the traffic. There are several kinds of traffic used for NoC simulation:

Uniform Random

A source selects the destination arbitrarily. It means that each core has an equal chance of being chosen as a destination for receiving data from a core in the network.

Local Traffic

A source selects the destination that is closer to it. The chance to be selected as a destination is reduced exponentially with increase in distance.

Transpose (Used in mesh topology NoCs)

A source that is located at position (x,y) transmits data to the core that is located at position (y,x).

Address Bit Reversal

A source that is presented by address (bm bm-1 ….b1 b0) sends data to the destination represented by address (b0 b1 ….bm-1 bm).

Application Specific

Some systems perform specific tasks and it is possible to have traffic information (communication pairs and volume). Communicating core pairs, communication density and communication bandwidth are determined by the predefined application.

3 Junction-Based Routing

Source routing has an important disadvantage of overhead for storing the path information in header of each packet sent. This disadvantage becomes worse as the size of the network grows. In this chapter we describe a routing technique, called Junction Based Routing (JBR) to remove this disadvantage. The idea of junction based routing is basically derived from the railway networks. Railway networks generally have a few large stations, called junctions which are connected by fast railways. A long distance journeys from a small town to another small town is achieved by first going to the nearest junction close to the source and from there reaching a junction close to the destination. In this chapter concepts and issues of this new routing technique are discussed.

3.1 An Illustration of Junction-Based Routing

Consider the following 7x7 mesh topology NoC that has the diameter of 13 hops (see *Figure 3-1*).

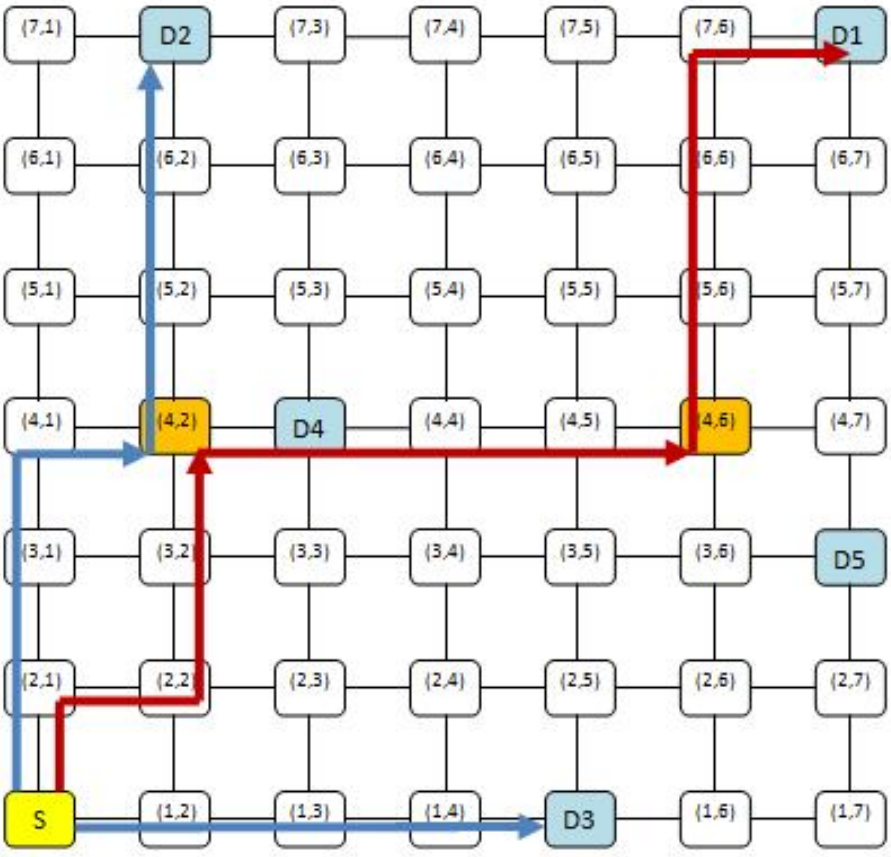


Figure 3-1.An illustration of using junctions in a 7x7 mesh topology NoC

The node that is presented using (x,y) is located at x_{th} row and y_{th} column. Distance between nodes that is located at position (x1,y1) and (x2,y2) is calculated using the formula:

Distance = |y2-y1| + |x2-x1|.

The number of routers used from a source node to a destination node is equal to the number of links used plus one. We define hop count as number of routers on the path from a source to the destination.

Hop Count = Distance + 1.

In distributed routing, the header flit stores the address of the destination and for 49 cores, a field of 6 bits is required (2^6=64 cores can be addressed). In source routing the header flit stores the entire path information and a field of 26 bits is used to store the path information (13 hops*two bits for each hop) [4].

In JBR, the path length is restricted to a constant number of hops, say 6. Therefore, we need a field of only 12 bits to store the path from source to the destination or junction. We give an example to illustrate how JBR will work. Consider that source node (S) is located at position (1,1) and destination node (D1) is located at position (7,7). First junction node (J1) is located at position (4,2) and second junction node (J2) is located at position (4,6). In the next sections, we show that these two junctions are enough to travel in the network with the given hop count limit of six.

S sends the packet to J1 as temporary destination, with the required path information. This junction appends new path information to the packet and forwards it. This new information is necessary to reach the second junction J2. J2 also appends new path information to the packet and forwards it. The information consists of the entire path information from J2 to D1.

To communicate from S to D2 (7,2), J1 is enough and J2 is not used. There is no need to use any junction for sending data to D3 (1,5), because the distance is less than maximum allowed path length (6 hops). A junction can be considered as a normal router for sending packets to next router. This situation happens for communicating between S and D4. Distance between S and D4 is 5 and packet has enough information to reach D4. J1 just forward the packet to the next router.

For sending packets to D5, both of the junctions are used. But length of path increases and becomes 11 hops instead of 9 hops. This issue describes in detail in next sections.

3.2 Analysis of Junction-Based Routing

3.2.1 Packet Format and Path Information

Consider the communicating pair (S,D1) in *Figure 3-1*. Three types of routing algorithms can be applied i.e. distributed routing algorithm, Junction-Based Routing algorithm (JBR) and source routing. The packet format for each of them is shown in *Figure 3-2*. As can be seen the packet header for JBR is so small in compare with simple source routing, but larger than distributed routing algorithm.

As can be seen, a large distance can be covered by going through intermediate temporary destinations (called Junctions) such that each sub-path (from source to a junction, junction to another junction, and junction to the destination) is smaller than or equal to a maximum hop count.

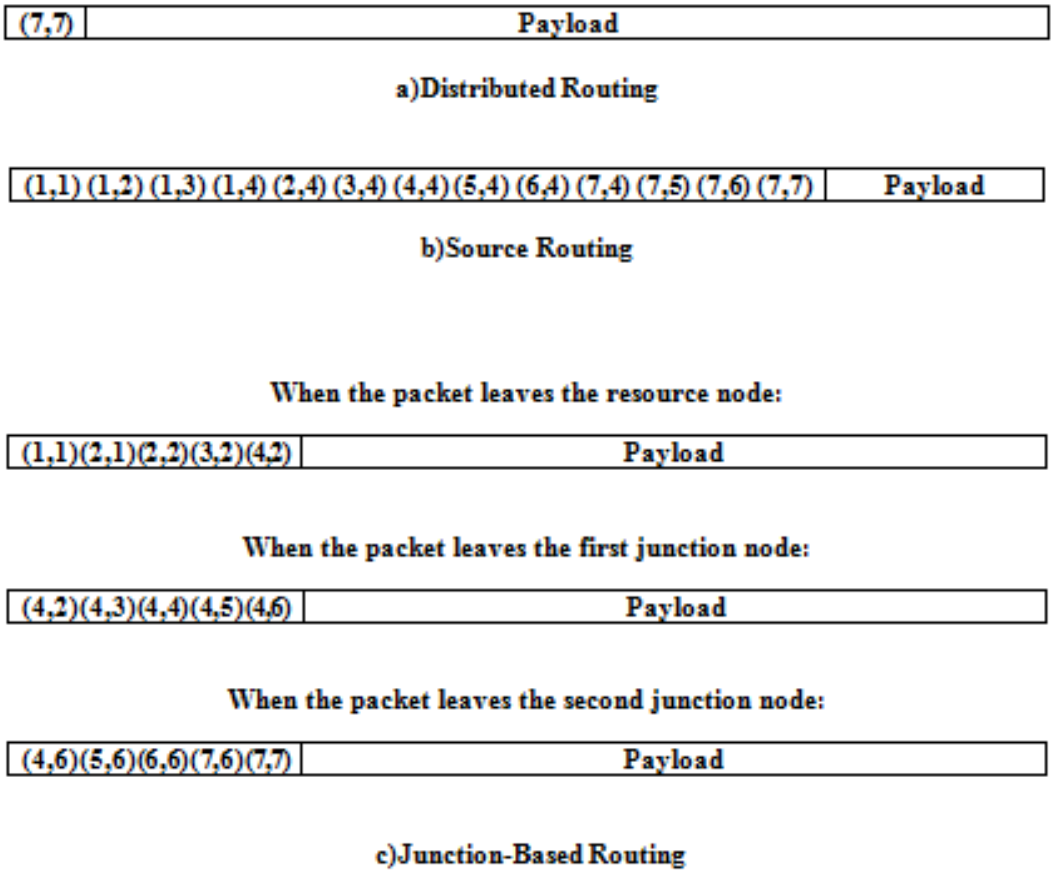


Figure 3-2.Packet format in distributed routing, source routing and JBR

For the communicating cores with large distance (larger than allowed path length), the source node appends path information from source to a junction. Since the packet entering this junction does not have any information about the output port in its header field, the junction adds the information of path from the junction to the destination or another intermediate junction and forwards it.

The required path information is stored at cores and junctions such that it can be easily used to fill up the required fields in the packet header. The paths can be also computed dynamically. Junctions and resources can also use different mechanism, i.e. resources can use memory tables and junctions can compute the required path using a routing algorithm dynamically. This increases the fault tolerance in the network because junctions can consider the situation of network, like congested links and broken links. But simple source routing has a small fault tolerance. In fact, JBR tries to use the advantages of both source and distributed routing by adding path information in each junction that can consider network situations, like link loads. In distributed routing, each router adds path information for one hop and forwards the packet but in JBR, each junction can add path information for more than one hop.

Obviously the architecture of junctions is different from simple routers and delay increases due to replacing path information in each junction. We try some methods to decrease this delay in next chapters.

The idea of JBR is general and will be applicable to all topologies- regular or irregular, but we apply this idea to mesh topology NoC.

3.2.2 Header Overhead in JBR

The first purpose in using JBR is to decrease the header overhead. Junctions append the path information using the destination address and packets need to carry the destination address in its header. The following table compares the overhead among distributed routing, source routing and JBR for various network sizes and segment length of 4 hops.

As can be seen from the *Table 3-1*, the overhead in JBR grows very slowly and therefore it is more scalable. Obviously, there will be a price paid, in terms of routing complications and increased latency in using this technique. In next chapters, we will deal with these important issues.

Table 3-1.The overhead among distributed routing, source routing and JBR for various network sizes

Mesh Size	Distributed Routing	Source Routing	JBR
5x5	6 bits	18 bits	8+6 = 14bits
6x6	6 bits	22 bits	14 bits
7x7	6 bits	26 bits	14 bits
8x8	6 bits	30 bits	14 bits
10x10	8 bits	38 bits	16 bits
16x16	8 bits	62bits	16 bits

For a given hop count limit (4), we need 8 bits to store the required path information, two bits for each hop [4]. The destination address needs 6 bits for a 7x7 network. An extra bit in the header is needed to indicate whether the path information is enough to reach the destination or not. In second case, junction will add the path information. In next chapter we describe this completely.

3.3 Challenges in JBR

Source routing is very suitable for small NoC platforms and has many advantages over distributed routing algorithm [4]. Since the packet entering a router has information about the output port in its header field, it simplifies router design and router delay is relatively smaller. Only recently researchers have started considering source routing as a routing candidate in NoCs [4]. Although, the authors in [4] give an analysis of the overhead due to source routing, they have not proposed any solution to reduce/handle the overhead with the size of the network. Some researchers have proposed hierarchical organization of networks and proposed hierarchical routing for large on-chip communication networks [16]. The proposed technique in this paper can be considered as an alternative to their approach.

In JBR, path information for only a few hops is stored in the packet header. With this information, either the packet reaches the destination, or reaches a junction from

where the path information for on-ward path is picked up. If a packet needs to go through a junction (or many junctions) the source just appends path information from source to the first junction. On reaching the junction, the packet picks up path information to reach the destination (or another junction) from this junction. There are many interesting issues related to this approach.

3.3.1 Number and Position of Junctions

Given the limit on the allowed number of bits available in the header flit for storing path information, a minimum number of junctions will be required to be placed in the network. In a 7x7 network and a given hop count limit of 7, it is easy to see that one junction at the position (4,4) is enough. *Figure 3-3* shows this situation.

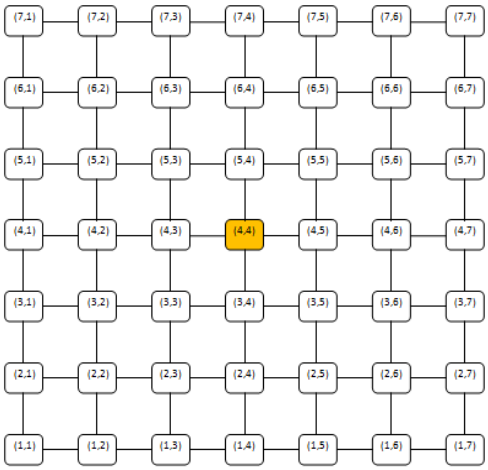


Figure 3-3 A 7x7 mesh topology NoC with one junction in the middle of network

We developed a MATLAB program that computes the minimum number of junctions and their positions for a given network size and a hop count limit such that the communicating pairs with a hop count larger than a given hop count limit can progress in the network through these junctions.

The pseudo code of the algorithm for calculating the minimum number of junctions and their positions is as following:

N is the network size and *H* is hop count limit. *N* and *H* are input variables for this function. We generate a graph in which every junction is a node and a pair of junctions has an edge between them if and only if the path length between them is less than the path length limit. This graph must be connected in the sense that there exists a path between any pair of junctions. This condition is necessary for reaching any junction from every other junction.

```
ALGORITHM Number _Postion_Junctions ( N,H )
{
Num_Junctions := 0;
IF (H<2*N-1)
{
Num_Configurations := 0;
WHILE (Num_Configurations == 0) DO
{
Num_Junctions := Num_Junctions + 1;
CALL Jun_Procedure;
}
}
ELSE
PRINT 'Need No Junction'
}


PROCEDURE Jun_Procedure (Num_Junctions)
{
FOR all possible combinations of Num_Junctions node(s)
DO
{
Assume the combination as one of the possible configurations for the junctions;
IF (There is a path from every node to at least one of this (these) junction(s)
with path length less than or equal to H)
THEN
{
Create a graph of these junctions such that there is a link between two nodes
of this graph (junctions) iff the path length <= H;
IF (This graph is fully connected)
THEN
{
Jun_Configuration:= Jun_Configuration + 1;
Store the selected combination; // As one of the possibilities for placing
junctions   in the network.
}
}
}
RETURN Num_Configurations;
}
```

3.3.2 A Case Study: Number and Position of Junctions

We use 7x7 mesh topology NoC to illustrate these issues. If path length limit is more than 7 and less than 13 (diameter of the network) then at least one junction is required. *Figure 3-3* shows the position of a junction in the centre of the network, which allows the use of hop count limit of 7. For hop count limit 6, we require at least two junctions. *Figure 3-1* shows the positions of two junctions in the network meeting the requirements listed above.

There are 40 feasible placements of two junction nodes in a 7x7 mesh NoC such that hop count limit of 6 is sufficient.

Table 3-2 shows some of different configurations for a 7x7 mesh NoC and a given H of 5. For instance, one of the possible configurations consists of the nodes that are located at positions (1,3), (3,4) and (6,4).

Table 3-2.Results: Some possible configurations of three junctions which are required for a 7x7 network and a 5 Hop Count Limit

Configuartion No.	Positions of Junctions
1	(1,3), (3,4), (6,4)
2	(1,3), (5,3), (4,6)
3	(1,3), (5,3), (5,6)
4	(1,3), (5,3), (5,7)
5	(1,4), (3,4), (6,4)
6	(1,4), (4,4), (6,4)
7	(1,4), (4,4), (7,4)
8	(3,4), (1,5),(6,4)
9	(4,2), (1,5),(5,5)

In *Figure 3-4*, some of the possible configurations for a 7x7 mesh topology NoC and a given H of 5 are depicted.

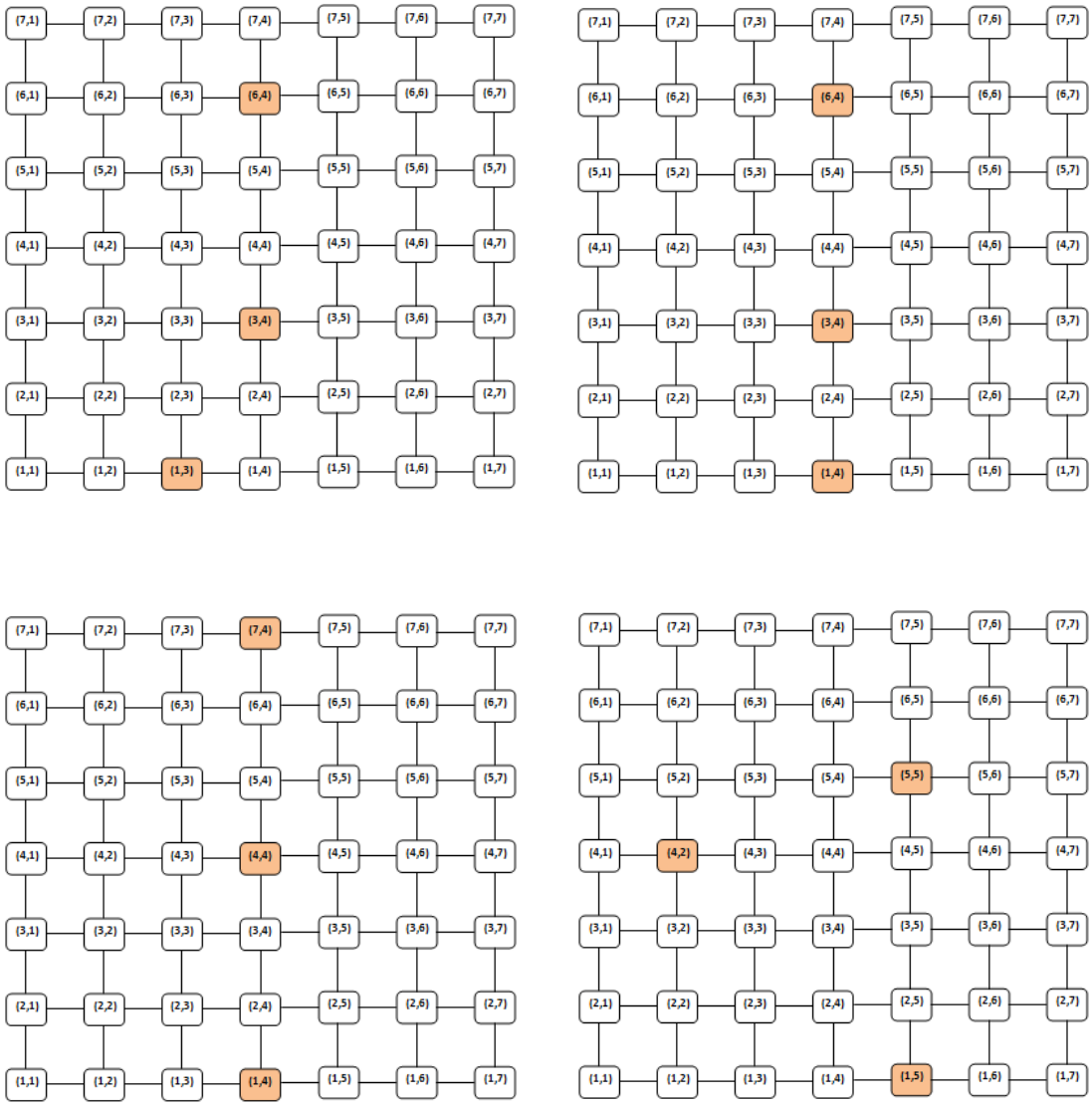


Figure 3-4. Some of different configurations of three junctions which are required for a 7x7 mesh topology NoC and a given H of 5.

Table 3-3 gives the minimum number of junctions required in a 7x7 network for a given hop count limit. It also gives the number of possible placement of junctions for various cases.

As we see the number of junction(s) is not comparable with the number of all nodes in the network and the number of junctions grows slowly with decreasing the hop count limit.

NJ/NN = (Number of Junction(s))/ (Number of Nodes).

The required number of bits for describing path in a header flit (it is supposed to use worm-hole switching) is calculated as following:

H*2+ (One bit for indicating the completeness of path) + number of bits for addressing the cores.

For a 7x7 network, 6 bits is enough to carry the destination address.

Table 3-3.Results: Minimum number of junctions and number of possible configurations for a 7x7 network with different Hop Count Limit

Hop Count Limit (H)	Number of Junctions (NJ)	Number of Configurations	NJ/NN	Number of Bits for Path Header
13	0	1	0	33
12	1	45	0.02	31
11	1	37	0.02	29
10	1	25	0.02	27
9	1	13	0.02	25
8	1	1	0.02	23
7	1	1	1/49=0.02	7*2+6+1=21
6	2	40	2/49=0.04	6*2+6+1=19
5	3	80	0.061	17
4	5	691	0.102	15
3	9	1	0.183	13
2	49	1	1	11

Table 3-4 shows the minimum number of junctions that are required for a mesh topology NoC with different sizes and a given hop count limit (H=6). The number of junctions grows very slowly. For instance, in a 7x7 network (49 cores) the number of junctions is 2 (2/49=0.0408) and in a 10by10 network, the number of junctions is 4 (4/100=0.04).

Table 3-4. Results: Minimum number of junctions for mesh topology NoC with various sizes and a given hop count of 6

Mesh Size	Minimum Number of Junctions (H=6)
7x7	2
8x8	3
9x9	3
10x10	4

Considering each row of *Table 3-5*, we observe that the number of junctions grows very slowly with the decrease in hop count limit. For instance, the minimum number of junctions is 3 for a 9x9 network and a given H of 6 (3/81=0.037). Number of junctions is 4 for a given H of 5 (4/81=0.049).

Table 3-5. Results: Minimum number of junctions for mesh topology NoC with various sizes and given hop counts of 6 and 5

Mesh Size	Minimum Number of Junctions (H=6)	Minimum Number of Junctions (H=5)
7x7	2	3
8x8	3	4
9x9	3	4

Having multiple configuration of junctions for a given path length can be useful for satisfaction of some other criteria like layout uniformity or optimization of performance in the context of application specific communication [10]. As we showed in first section of this chapter, the use of junction based routing can lead to increase in the hop count between some pairs. This average increase in hop count per packet is dependent on the position of junctions as well as on the amount of communications between pairs in the network.

3.4 Increase in Path Length by Using Junctions

In the first section, we observed that path length between S (1,1) and D5(3,7) is increased by going through junctions. In pure source routing, distance between source and destination nodes is 8 but using JBR the distance is 10.

The distance between S (source node) and J1 (first junction) is 4 and the distance between J1 and J2 (second junction) is also 4. The distance between J2 and D4 (destination node) is 2 and therefore, distance between source node and destination node become 10.

Increase in path length is unavoidable while the minimum number of junctions is used. One of the future works in JBR will be to find the minimum number of junctions such that path length does not increase.

3.4.1 Calculating Extra Overhead of Increase in Path Length

Here we describe how to calculate extra overhead of increase in path length in a junction-based network using the following formula:

$$\textbf{Extra overhead}= \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} V_{ij}(JD_{ij} - D_{ij})}{\sum_{i=1}^{M}\sum_{j=1}^{M} V_{ij} D_{ij}}$$

Where,

JD_{ij} =Distance between node *i* and node *j* using Junction based routing

D_{ij} =Distance between node *i* and node *j* using source routing,

V_{ij} = Communication volume between node *i* and node *j*,

M is the total number of nodes in the network and in a 7x7 NoC, *M* = 49.

For a given junction configuration, the following procedure finds the increase in average hop count for a set of communications C. Assume the communication $c_i \in C$ has a volume v_i.

```
PROCEDURE Hop_Count_Increase (C)
   {
   Total_Overhead := 0;
   FOR each communication c_i in C  DO
   {
   IF Shortest_path_length > H  // H is a given Hop count limit
       THEN
       {
       Find the shortest path through junction(s);
       Overhead := v_i*(L - Shortest_path_length);
// vi is communication volume. For finding shortest path through junctions, we
create a graph of junctions and as we know this graph is connected (section
3.3.1). Then a given communicating pair finds the shortest path using finding
shortest path in a connected graph that is a common and solved problem.
        };
   Total_Overhead :=  Total_Overhead + Overhead;
   };
   Av_hop_increase := Total_Overhead / Σc_iv_i
```

We developed a MATLAB function for computing total overhead for all possible configurations for a given Junction-Based mesh network. Input variables are all possible configurations and traffic type or application specific communication matrix.

3.4.2 Results of Computing Extra Overhead Using the Developed Tool

We have computed the average increase in hop count for uniform random traffic and application specific traffic favoring locality. We make the following assumptions/choices for our computations.

For modeling a realistic communication traffic we assume that each core communicates with at least one and at most N (Network Size) cores in the network. Communication volume for each pair is a random number in range 1 to 10. Obviously an acknowledgement signal is also taken into account.

For modeling a local traffic we divide the cores into three categories as illustrated in *Figure 3-5*. The first category consists of the nodes that are located at the corners of the network. Second one is the nodes that are located at the boundary of the network and third one includes the other nodes.

The probabilities used for choosing destinations for the nodes that are located at the corners of the network as sources, are as following:

Probability of destination at 1 hop is 15 %, probability of destination at 2 hops is 20 %, probability of destination at 3 hops is 25 % and probability of destination at more than 3 hops is 40 %.

The number of nodes that are located at one hop away from a node that is located at the corners of the network is two. For instance, the source node is located at position (1,1) and it is supposed to choose a destination for this source. Nodes are located at positions (1,2) and (2,1) have 15% chance of being selected as a destination for receiving data from node that is located at position (1,1).

The probabilities used for choosing destinations for the nodes that are located at the boundary of the network as source, are as following:

Probability of destination at 1 hop is 40 %, probability of destination at 2 hops is 30 %, probability of destination at 3 hops is 15 %, and probability of destination at more than 3 hops is 15 %.

The probabilities used for choosing destinations for the other nodes as sources, are as following:

Probability of destination at 1 hop is 30 %, probability of destination at 2 hops is 40 %, probability of destination at 3 hops is15 %, probability of destination at more than 3 hops is 15 %.

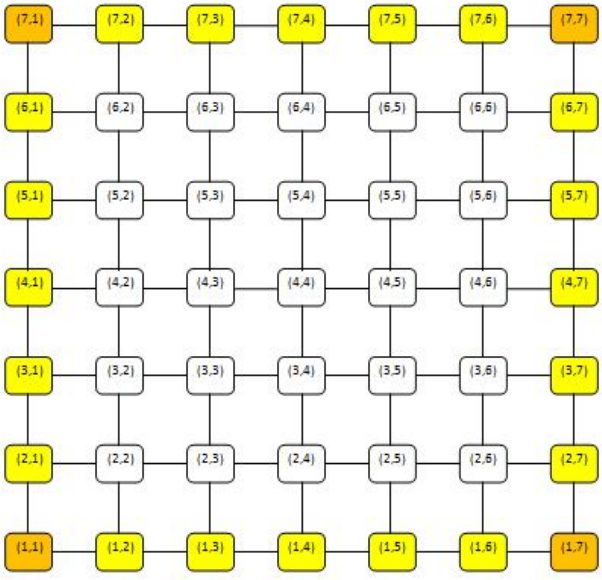


Figure 3-5.Three different types of nodes that are categorized depending upon their positions in a 7x7 mesh topology NoC.

Based on our experiments and computations, for a 7x7 mesh NoC with path length limit of 6 hops:

a) For uniform random traffic, the average increase in hop count for different configurations vary between 0.05% to 3%

b) For application specific traffic favoring locality, the average hop count for different configurations vary between 0.01% to 0.09%

It is possible to find the best possible configuration of junctions for a given communication traffic of the application to minimize the increase in extra hop count.

Two of the best configurations of junctions in a 7x7 mesh NoC with a given hop count limit of 5 are shown in *Figure 3-6*. One of the worst configurations is illustrated in *Figure 3-7*. *Figure 3-8* and *3-9* present some of the best configurations and the worst configurations in local traffic respectively.

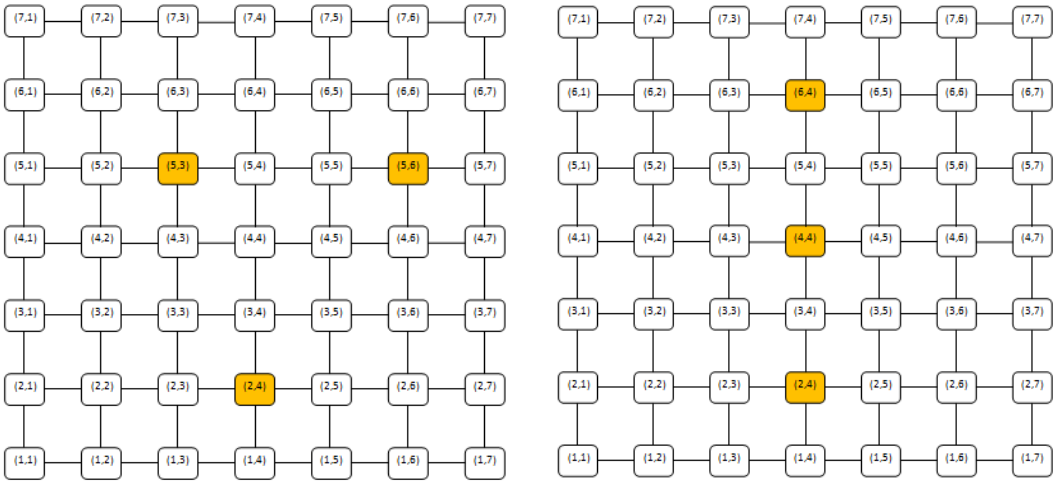


Figure 3-6.Two of the best configurations of junctions in a 7x7 NoC and a given H of 5 and for a random traffic

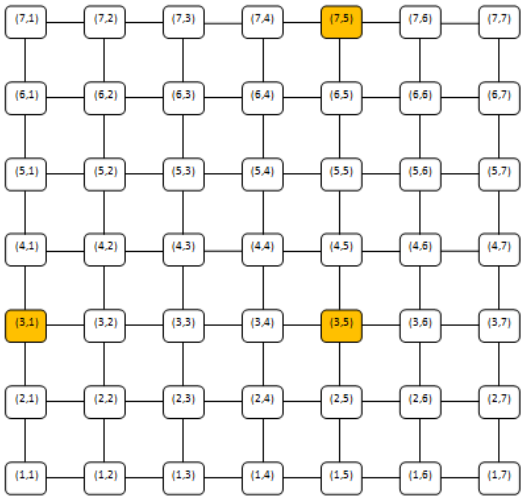


Figure 3-7.One of the worst configurations of junctions in a 7x7 NoC and a given H of 5 and for a random traffic

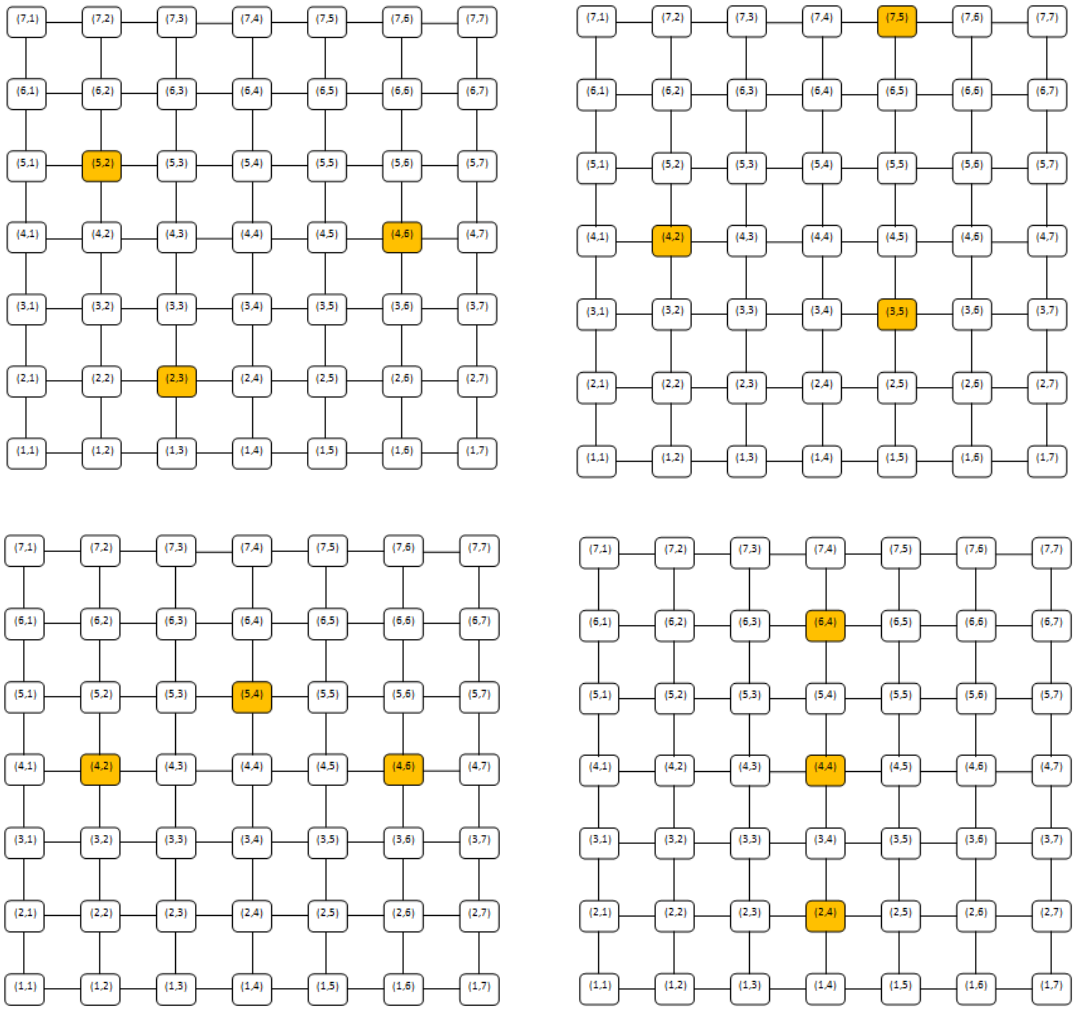


Figure 3-8.Some of the best configurations of junctions in a 7x7 NoC and a given H of 5 and for a local traffic

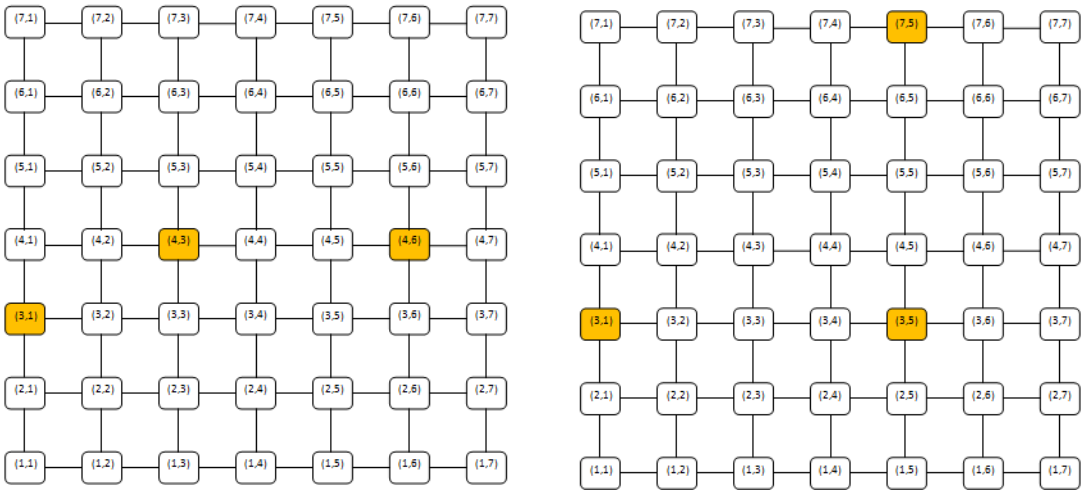


Figure 3-9.Two of the worst configurations of junctions in a 7x7 NoC and a given H of 5 and for a local traffic

3.5 Junctions and Deadlock-Free Routing

Presence of junctions makes the network non-homogeneous in the sense that all routers are not identical. A junction router has functionality of a router plus some other functionality. If one is not careful in computing the paths, it can lead to a deadlock.

Suppose we want to use negative first routing algorithm for a 7x7 mesh NoC and a given H of 7 (see *Figure 3-10*). Consider the communicating pair is S (1,7) and D(7,1). Packets have to go through a junction since distance between nodes S1 and D1 is more than 7 hops. The only path between S1 and D1 is shown in the figure. A junction has to be at the position (1,1), because the junction should be close enough to the source and destination nodes and the only possibility is (1,1).

Suppose S is sending data to the node that is located at the position (7,2). Distance between these two nodes is more than 7 hops but the only path between these two nodes does not go through J and J cannot be used for getting the path information. Then another junction is needed for this communicating pair and one junction is not enough.

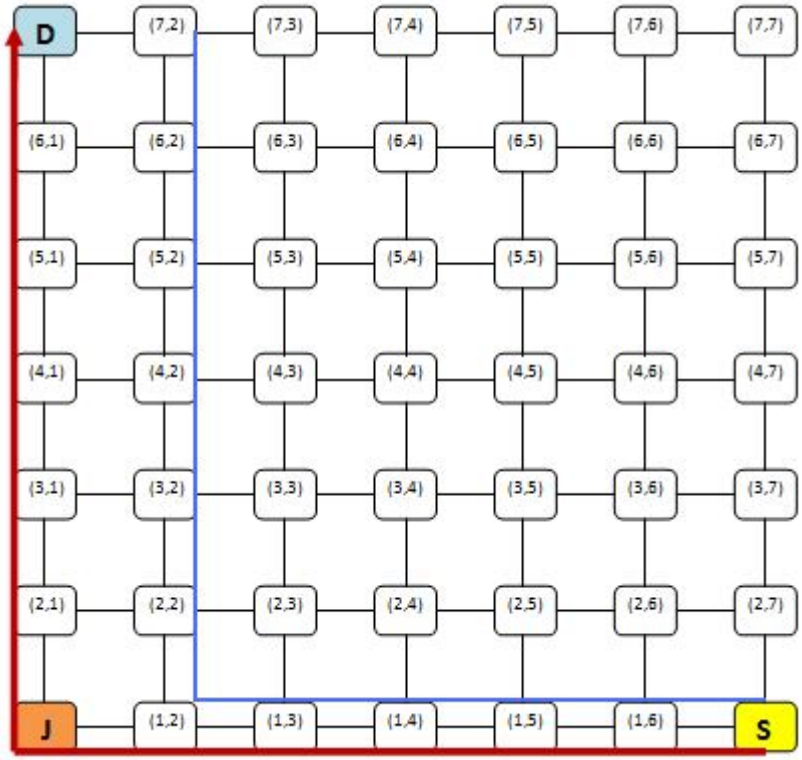


Figure 3-10.S is sending data to D and it must use node (1,1) as the only possible location for getting the path information from J to D.

By investigating, we find out that, 6 junctions are needed at least for communication between various nodes using Negative-First routing algorithm. In next chapter, we use a developed MATLAB function that proves that 6 junctions are necessary and sufficient. This is done by exhaustive search of possible solutions. In *Figure 3-11* each vector shows the route that is the only possible path for corresponding communicating pair. For instance, for sending packets from (1,7) to (7,6) the only possible route is:

{(1,7) (1,6) (2,6) (3,6) (4,6) (5,6) (7,6)}.

The path length is more than path length limit and using a junction in one of the following nodes is necessary:

{(1,6) (2,6) (3,6) (4,6) (5,6)}.

There are similar situations for the following communicating pairs: (S1,D1), (S2,D2), (S13,D3), (S4,D4) and (S5,D5). The next chapter focuses on deadlock free routing algorithm for a junction based network. Different MATLAB functions are used for finding the minimum number of junctions and their positions for different routing algorithms and also computing paths and finding best paths for different traffic types. Application specific communication is also taken into account.

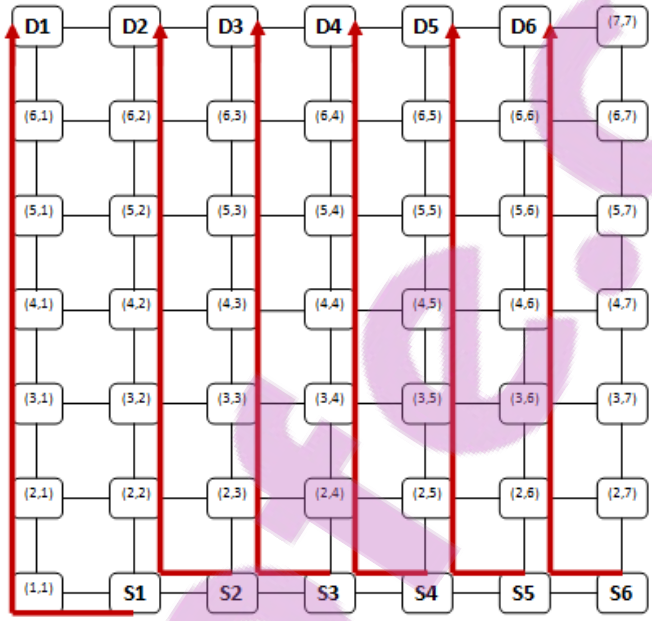


Figure 3-11. The only possible paths for some of the communicating pairs using Negative-First routing algorithm.

4 Path Computations for Mesh Topology NoC with Junctions

In last chapter, we showed that it is not possible to use Turn-Model based deadlock-free routing algorithms for NoC using the minimum number of junctions. In this chapter, the required number of junctions and the positions of these junctions are computed for a mesh NoC which uses Turn-Model based deadlock-free routing algorithms. Using a tool developed in MATLAB, all possible paths are given and the adaptivity is compared with adaptivity of paths in source routing. Link load distribution is also analyzed and one path for each communicating pair is selected for different types of communications and traffic patterns. Finally, a method is suggested for encoding the paths and junction architecture is analyzed to some extent.

4.1 A Tool for Computing Paths for JBR

A tool is developed in MATLAB that computes the required number of junctions and all possible junction configurations for different Turn-Model routing algorithms. It also computes paths for source routing and JBR. This tool uses minimal routing and does not allow 180 degree turns. As mentioned before, Turn-Model routing algorithms are used due to its simplicity.

The user gives the size of mesh network and hop count limit and selects a routing algorithm from XY, Odd-Even, West-First, North-Last and Negative-First. These algorithms were described in *Chapter 2*. The task is to compute the paths to be used between various pairs of nodes which communicate. In a general situation we need to compute paths for every pair of nodes, but in an application specific case we need to store paths only for communicating pairs. The computed paths should ensure deadlock freedom as well as low latency through avoidance of congestion. Since every packet carries the path information in its header, the path must be efficiently encoded. The required path information needs to be stored at normal nodes and junction nodes such that it can be easily used to fill up the required fields in the packet header.

The tool computes all the paths for all the pairs in the network using source routing and JBR. These paths are stored in a number of tables. Adaptivity and other parameters are also computed that can be useful for analyzing different configurations of the junctions.

N is the network size and H is the hop count limit. The pseudo-code of the algorithm which is used by the mentioned tool is described below:

```
ALGORITHM Number _Postion_of_Junctions_and_Paths (N,H,
SelectedRoutingAlgorithm)
 {
 Num_Junctions := 0;
 IF (H<2*N-1)
  {
  Num_Configurations:=0;
  WHILE (Num_Configurations==0) DO
   {
   Num_Junctions := Num_Junctions +1;
   CALL Path_Procedure;
   }
  }
 ELSE
  PRINT 'Need No Junction'
 }


PROCEDURE Path_Procedure (Num_Junctions, SelectedRoutingAlgorithm)
  {
  FOR all possible combinations of Num_Junctions node(s) DO
    {
    Assume the combination as one of the possible configurations for the
junctions;
     IF (There is at least one path from every node to all the other nodes using
the Selected Routing Algorithm and with path length less than or equal to H)
     THEN
       {
        Jun_configuration:= Jun_configuration+1;
        Store the selected combination; //as one the possibilities for placing
junctions in the network
        }
    }
RETURN Num_Configurations;
   }
```

4.2 Analysis of Junction-Based Networks Using Different Turn-Model Routing Algorithms

This section compares different configurations of junctions for different routing algorithms. We also analyze them and compare their parameters with corresponding ones in normal source routing.

4.2.1 Junction Configurations for North-Last Routing Algorithm

In the last section, we showed that it is not possible to apply Turn-Model based routing algorithms when the minimum number of junctions is used. The mentioned developed tool gives the required number of junctions and their positions while north-last routing algorithm is applied (see *Figure 4-1*). All possible paths are also computed and used for analysis of this case.

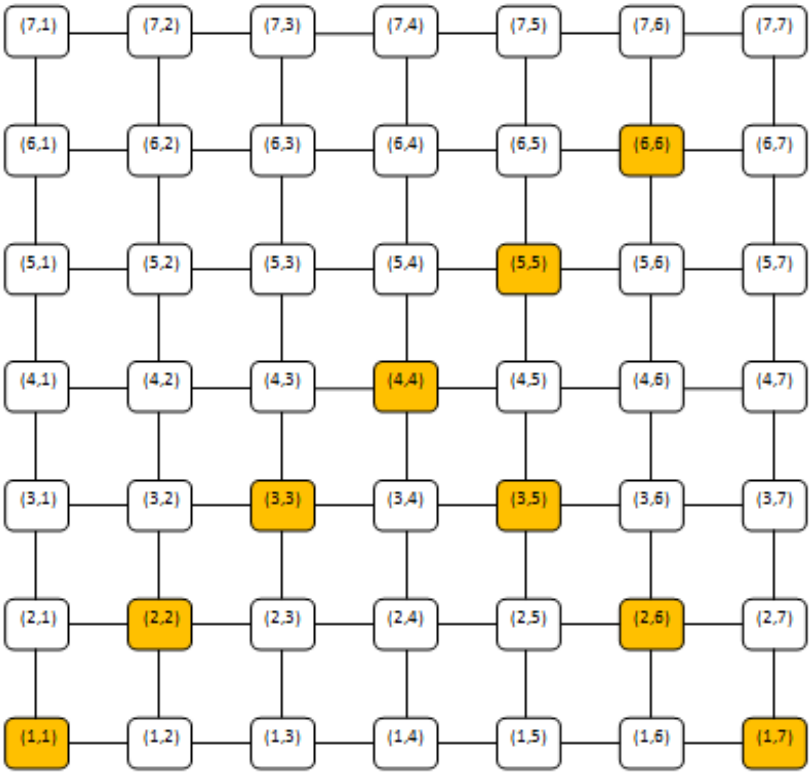


Figure 4-1.7x7 Junction-Based network using North-Last routing algorithm with hop count limit of 7

Applying North-Last routing algorithm, minimum number of junctions is 9. The number of junctions is not comparable with the total number of nodes. We define junction ratio as follows:

NJ/NN=Number of Junctions/Number of Nodes=9/49=0.18.

Figure 4-1, shows the only possible configuration for this case. The numbers of available paths are generally reduced when using JBR. For example, in the above 7x7 network, the possible paths for all communicating pairs are equal to 24577. If the path length is not restricted to a constant number of hops then in a 7x7 network the possible paths for all communicating pairs using North-Last routing algorithm are equal to 26443. We define relative path adaptivity of JBR with respect to source routing as follows:

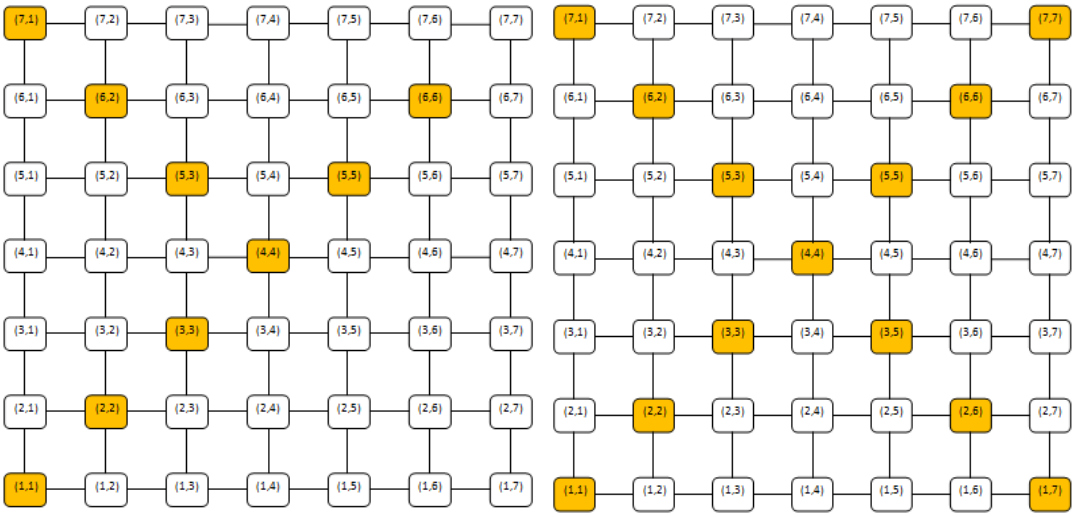
PJBR/PSR=Number of Paths in JBR/Number of Paths in Source Routing

=24577/26443=0.93.

A high value shows that JBR retains high path adaptivity.

4.2.2 Junction Configurations for Other Kinds of Routing Algorithms

Figure 4-2 shows the required junctions while West-First, XY and Negative-first routing algorithms are applied.



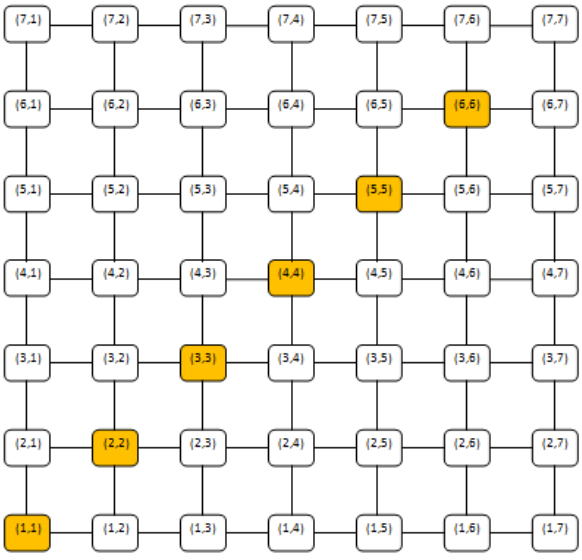


Figure 4-2.7x7 Junction-Based networks using various routing algorithms with hop count limit of 7

For instance, applying XY routing algorithm, number of required junctions is 12. *Figure 4-2* shows the only possible configurations for these routing algorithms used.

Applying Odd-Even routing algorithm, we need 6 junctions in a 7x7 network. There are different possible configurations for these junctions. Five of them are depicted in *Figure 4-3*. First possible configuration is the nodes that are showed at this set of nodes:

{(2,1),(2,2),(3,3),(4,4),(5,5),(6,6)}, PJBR =10999.

This configuration is considered in *Table 4-1* for Odd-Even routing algorithm.

The other possible configurations are:

{(3,1),(2,2),(3,3),(4,4),(5,5),(6,6)}, PJBR =11117

{(4,1),(2,2),(3,3),(4,4),(5,5),(6,6)}, PJBR =11149

{(5,1),(2,2),(3,3),(4,4),(5,5),(6,6)}, PJBR =11185

{(6,1),(2,2),(3,3),(4,4),(5,5),(6,6)}, PJBR =11073

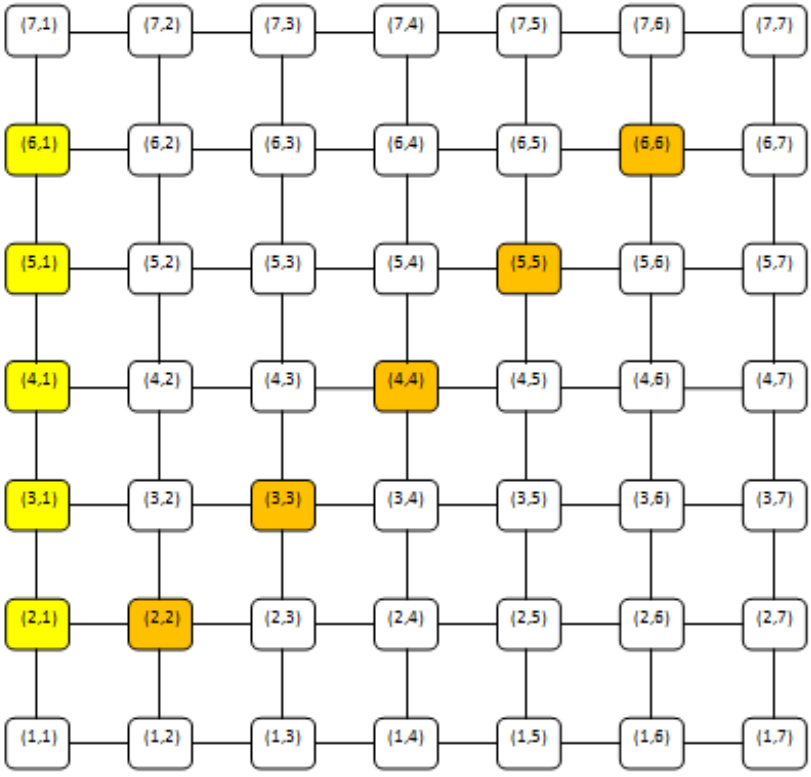


Figure 4-3.A 7x7 Junction-Based network using Odd-Even routing algorithm

4.2.3 Comparison of Different Routing Algorithms Used

As can be observed in *Table 4-1,* the number of junctions that are needed for a mesh topology NoC is not comparable with the total number of nodes. Applying Negative-First routing algorithm the number of junctions is 6.

Table 4-1.Results: Number of junctions and number of paths in JBR for different routing algorithm.

Routing Algorithm	Junction Ratio	Relative Path Adaptivity
Negative-First	6/49=0.12	19063/26443=0.72
Odd-Even	6/49=0.12	10999/12481=0.89
West-First	9/49=0.18	24577/26443=0.93
North-Last	9/49=0.18	24577/26443=0.93

4.3 Path Selection

In this section, we select a path for each communicating pair considering link load in a junction-based network.

4.3.1 Link Load Distribution

In the last section, we showed that a developed tool computes all admissible paths for each communicating pair. We can select one path for each pair randomly. But it is possible to choose them more cleverly if we know the traffic pattern. Link load distribution can be taken into account for this selection. In this situation link load in the NoC can be balanced to some extent during the path selection process.

Assume all the cores which are communicating with each other for a specific application are known. We order the communications according to locality, bandwidth and adaptivity i.e. their cost depends upon the distance between source and destination, the required bandwidth for the communication and the number of possible paths between the source and the destination nodes. Hence,

*Communication Cost = (Communication Bandwidth * Distance) / Path Adaptivity*

The communications are ordered in descending order with respect to the cost of communication. Paths for the communications with higher Communication Bandwidth and larger Distance are selected first. Communicating pairs with less Path Adaptivity are considered first as well. The reason for selecting paths with smaller cost of communications in the end is to have more control to distribute the load among links. A developed MATLAB function computes the standard deviation of links load (SD) and the maximum and minimum and mean value of them. The pseudo-code of the algorithm is described below:

Input parameters are network size, hop count limit, routing algorithm and traffic pattern or application specific communication. Output parameters are one path for all communicating pairs and SD, Mean, Max and Min.

Pseudo-Code of the Algorithm

1: Generate_All_Paths();

{Find all possible paths for each pair for a given network size and a hop count limit and desired routing algorithm}

2: Generate_All_Communications();

{Find the communicating cores for specific application}

3: Select_Adaptivity_One();

{Find the communicating cores with path adaptivity==1 and store them in the final paths table}

4: For (All_Other_Communications)

5: Compute_Communication_Cost();

{Compute cost for all communications using the formula suggested in *Section 4.3.1*}

6: End for

7: Order_All_Communications();

{Creating ordered list of communications starting with the largest and ending up with the smallest communication cost}

8: While (End_of_Communication_List)

9: Select_Best_Path();

{Select and store a path that results in better link load distribution using adaptivity of routing algorithm}

10: End while

4.3.2 An Example of Selecting the Best Path for Each Communicating Pair

In *Figure 4-4*, link load distribution is presented for local traffic when Negative-first routing algorithm is used. Local traffic pattern used sends data to at least one and at most seven nodes from each node. Each node also chooses a random value in the range 1 to 10 for communication volume applied.

Maximum link bandwidth, mean and standard deviation of the link load distribution are equal to 41, 17.488 and 8.295 respectively.

In this case, the number of communicating pairs that use junctions is 532 and the number of all communicating pairs is 2401. The percentage of pairs using junctions is,

532/2401=0.22.

Therefore, 22% of all communicating pairs use junctions for getting the path information.

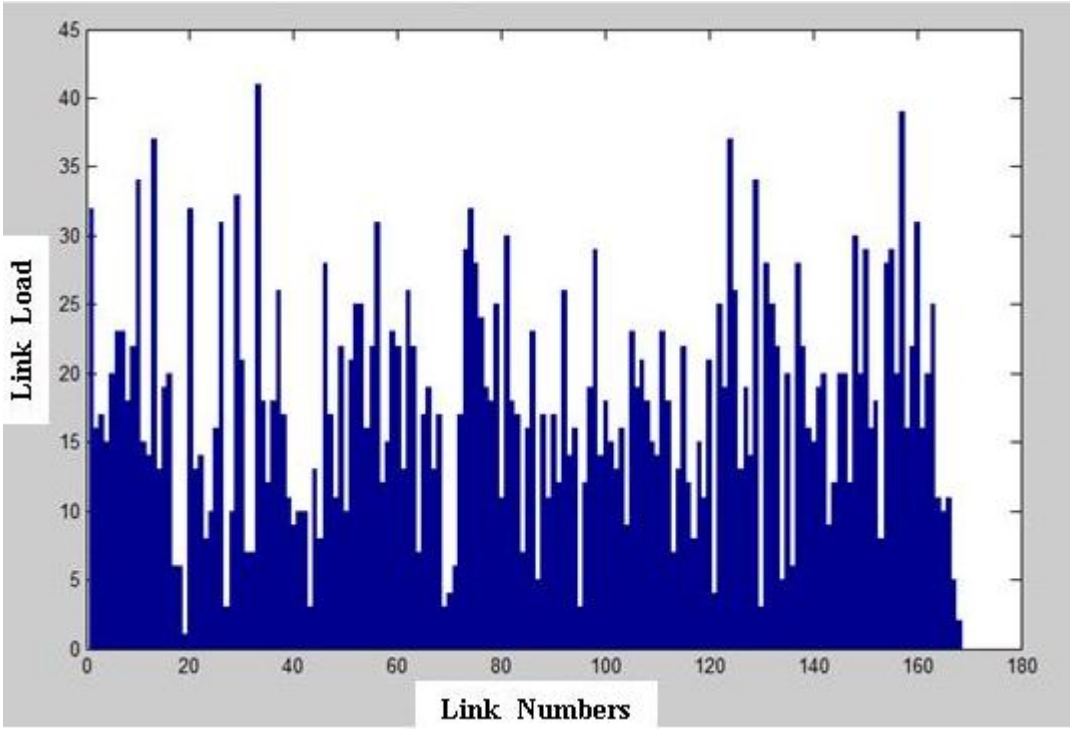


Figure 4-4.Link load distribution in a 7x7 NoC with junctions for local traffic using Negative-First routing algorithm

Figure 4-5 shows standard deviation of the link load distribution during the selection of one path for each communicating pair by the developed tool. As described in the last sub-section, for choosing one path for each communicating pair, the tool starts from the communications with zero cost value then SD is zero in the beginning and increases suddenly for choosing a path for the first communicating pair considered. The tool tries to balance the network load among the links and SD is decreased slowly and at the end the best possible paths for all the communicating pairs are selected such that SD is as small as possible.

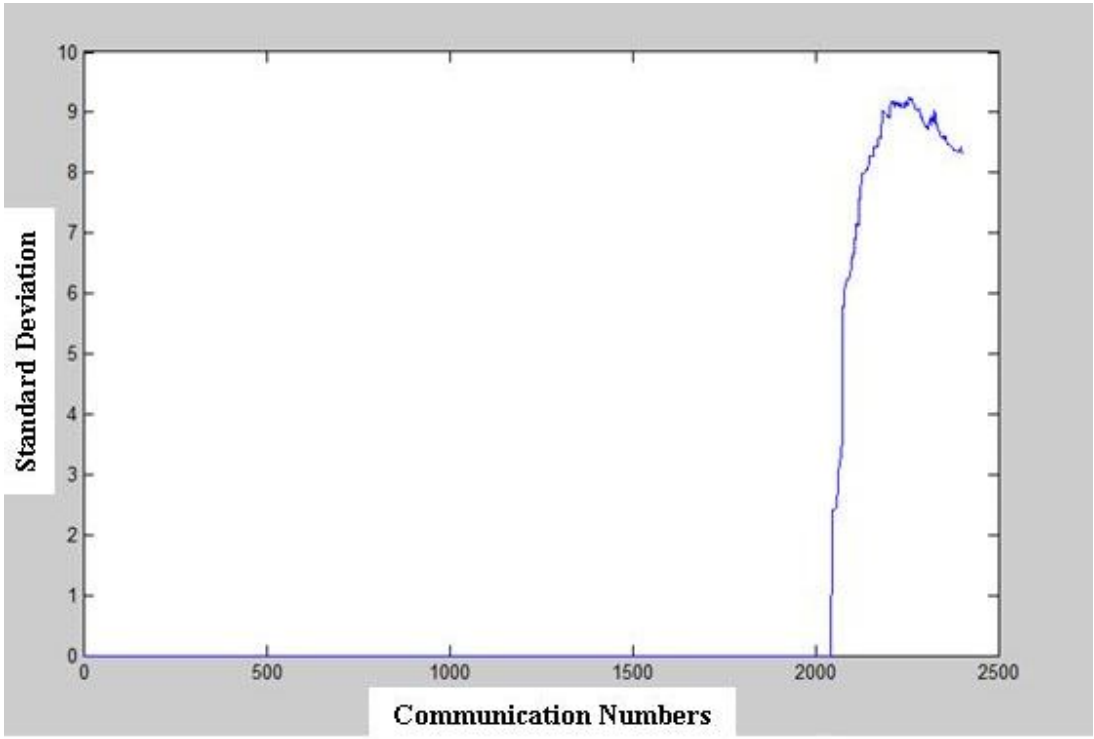


Figure 4-5.Standard deviation of link load distribution during the selecting one route for each communicating pair

Consider these two formulas used for computing communication cost:

*Communication Cost = (Communication Bandwidth * Distance)*

*Communication Cost = (Communication Bandwidth * Distance) / Path Adaptivity*

In the first formula path adaptivity is not considered [4]. Adding path adaptivity to the formula used for computing communication cost results in a better link load distribution. For example, link load distribution in a 7x7 NoC in local traffic for Negative-First routing algorithm using these two different formulas are computed and compared. Average improvement of link load distribution is shown in *Table 4-2.*

Table 4-2.Improvement in link load distribution using the formula with path adaptivity factor for Negative-First routing algorithm in local traffic

Average Improvement of Standard Deviation of the Link Load Distribution	10.5%
Average Improvement of Mean of the Link Load Distribution	15%
Average Improvement of Maximum Link Bandwidth	16.5%

4.4 Packet Format in JBR

We showed the packet format and flit format in *Chapter 2*. In this section, flit format is described in more details.

4.4.1 Flit Types

Flit type determines the type of a flit and is encoded using 2 bits. Flit type is encoded to 00 when the type of the flit is *HEAD* and the flit carries the route information. Other flit types are *BODY* and *END*. In *Table 4-3*, we present the way they are encoded. The purpose for the encoding of END flit is described in *Section 4.4.4*.

Table 4-3.Encoded flit types

Flit Types	Flit Type Bits
BODY Flit	01
END Flit with Full Payload	10
END Flit with Payload Less than 4 Bytes	11

4.4.2 HEAD Flit

Structure of a header flit is depicted in *Figure 4-6*. As can be seen two different formats can be used. In the second format, flit does not consist of source address.

Flit Type	Source Address	Destination Address	IndBit	Route Information	Payload

a) First format: header flit carrying the source and destination addresses

Flit Type	Destination Address	IndBit	Routing Information	Payload

b) Second format: header flit carrying just the destination address

Figure 4-6.Format of HEAD flit in JBR routing

Each field in the header flit is described below:

Source Address

Source address can be used to send acknowledgement signal back from a destination resource to the source resource. If acknowledgement is not required, then source address may not be sent. The address of source resource is stored using six bits in a 7x7 network.

Destination Address

Destination address is used to find the path from a junction to another junction or the destination when arriving at a junction and route information is not enough to reach to the destination. Then destination address shall be carried in each header flit. The address of destination resource is stored using six bits in a 7x7 network.

IndBit

In JBR some of the packets need to get the information about rest of their route from the junctions. Then a junction looks at a bit, called IndBit, to recognize that it should either act as a simple router or as a junction. When a junction acts as a simple router, it just forwards the packet to the next router in the network. When a junction acts as a real junction, it must append the information about the rest of the path to the destination or to the other junction.

A source core sends a head flit that consists of routing information to the destination or to a junction. If the distance between the source core and the destination core is larger than allowed path length then IndBit is equal to "0", otherwise that IndBit is equal to "1". When a head flit arrives at a junction, the junction looks at IndBit to determine whether the head flit consists of required path information to reach the destination or not. If yes, it just forwards the packet to the next router in the network otherwise that, the junction looks at the route information to determine whether the output port has been determined or not. If yes, the junction just forwards the flit to the next router otherwise that, the junction loads the path information from its memory and forwards the packet to the next router. We describe path encoding in detail in next section.

IndBit represents Indicator Bit that determines whether the path information is enough to reach the destination or not. One bit is enough for this purpose.

Routing Information

Routing information consists of the information of the path between a source node and a destination node or a junction node. For a 7x7 mesh network with a hop count limit of 7, at most 14 bits are needed to store the routing information using 2-bit clockwise router port address encoding scheme [4].

Payload

A header flit in JBR consists of payload field and can carry payload of data due to reduced path information. In *Sub-section 4.4.5*, size of the payload is described completely.

4.4.3 BODY Flit

In *Figure 4-7*, structure of a body flit is shown. Size of the payload is 30 or 32 depending on the size of the flit. First two bits show the flit type and will always be equal to "01" for a BODY flit. All the subsequent bits will carry the payload.

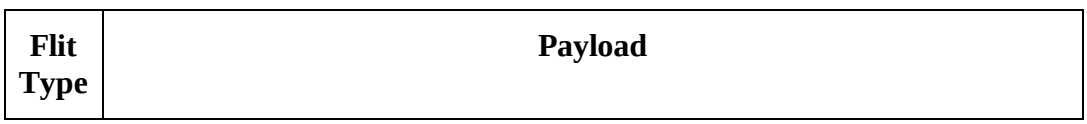


Figure 4-7.Format of BODY flit in JBR routing

4.4.4 END Flit

Structure of an END flit is presented in *Figure 4-8*. As can be seen, an END flit consists of 2-bit "Payload Size" field as well as Flit Type. Payload Size indicates the number of bytes of data that are carried by this flit if Flit Type is equal to "11" [4]. If Flit Type is equal to "10", then these two bits are part of payload.

Figure 4-8.Format of END flit in JBR routing

4.4.5 Comparison of Two Different Header Flit Formats

Consider a 7x7 network. Since 34 bit flit size is not a standard size, 32 bit flit size is also considered and analyzed. In *Table 4-4*, payload size (in bits) is presented for different header flit format. For instance, consider the second row and fifth column of the table. The size of the header flit is 34 bits and the header flit does not carry the source destination address. Hop count limit is 4 and 8 bits is needed to store routing information, therefore 17 bits (2 byte) is free for data payload:

2 (flit type) + 6 (destination address) + 1 (IndBit) +8 (routing information) + 17 = 34

Table 4-4.Results: comparison of different header flit formats in JBR in terms of number of bits of data that can be carried by the header flit

Format Type	H=7	H=6	H=5	H=4	H=3
First format with 34 bits header flit	5	7	9	11	13
Second format with 34 bits header flit	11	13	15	17	19
First format with 32 bits header flit	3	5	7	9	11
Second format with 32 bits header flit	9	11	13	15	17

4.5 Paths Encoding

Each resource stores the path information and uses these pre-computed paths for sending packets. For instance, in a 7x7 mesh network with an H of 7, the size of the memory used in every resource is almost half of the size of the memory that is needed in source routing.

Each junction just stores the pre-computed path information needed for each junction. We preferred to use a table-based routing. By using table-based routing, the junctions can work faster and their functionality is simpler. It is possible to generate the paths using a run-time routing algorithm. By using run-time routing algorithm for generating paths in a junction, the junction can react to different network situations appropriately. This is one of the future works for improving the fault-tolerance and load distribution and quality of service.

2-bit clockwise router port address encoding scheme is used for encoding routing information [4].

Each junction is allowed to contain more than one table. In this case, routing is done faster but it is costlier. For instance, it is allowed to use 5 tables (one for each direction).

Finding the best architecture for a junction and an efficient way for storing information in sources and junctions tables are some of the future works in JBR.

An Example of Paths Encoding in a 7x7 Network

Consider a 7x7 network with a hop count limit of 7. Negative-First routing algorithm is applied and the positions of junctions are as shown in *Figure 4-9*. For sending data from the first node to the 28th node, 25th node is used as the junction. Path between the source node and the destination node is shown using an arrow in the figure below.

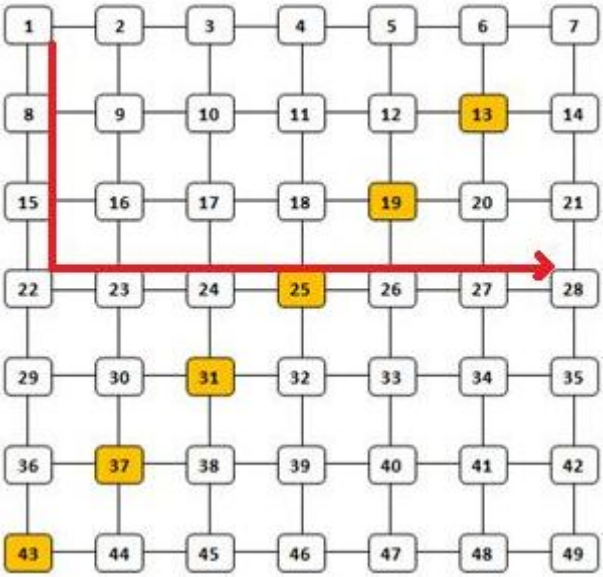


Figure 4-9.A 7x7 Junction-Based network using Negative-First routing algorithm

In *Figure 4-10*, we present routing information in the form of node numbers.

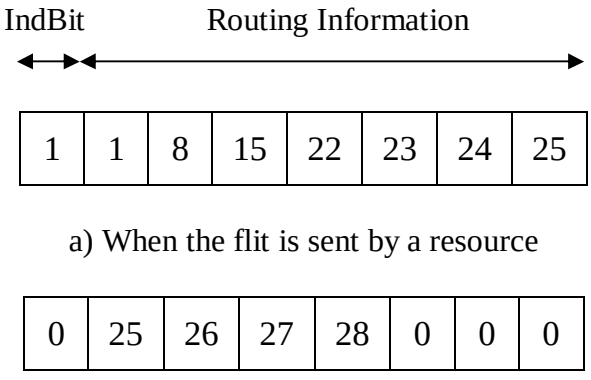


Figure 4-10.Routing information is presented in the form of node numbers

In *Figure 4-11(a)* and *4-11(b)*, routing information is represented in the form of directions defined below:

- N or 1 represents North Direction
- S or 2 represents South Direction
- E or 4 represents East Direction
- W or 3 represents West Direction
- R or 5 represents Resource Direction (in the case that destination is reached, otherwise that junction table should be accessed.)

1	S	S	S	E	E	E	R
0	E	E	E	R	X	X	X

a) Representing the route information in the form of the directions defined

1	2	2	2	4	4	4	5
0	4	4	4	5	0	0	0

b) Assigning numbers to the directions defined

1	11	01	01	00	01	01	11
0	10	01	01	11	X	X	X

c) Routing information is encoded using 2-bit clockwise

Figure 4-11.Routing information is presented in the form of node numbers

Path is encoded in the header flit. It is desirable to encode the path in such a way that the overhead of path information is minimized and the path decoding is as easy as possible (see *Figure 4-11(c)*). 2-bit clockwise router port address encoding scheme is used for encoding routing information [4].

4.6 Architecture of a Junction-Based Router

A router switches an incoming message to an output channel. In junction-based routing, the output channel is read from the packet header or it is selected by looking up a table that is accommodated in the junction (the pre-decided path information is available in the junction table). Router architecture for junction-based mesh topology

NoC is illustrated in *Figure 4-12*. Simple junction-based routers are desirable because of their expected lower implementation costs. Router design for source routing is much simpler than the router design to handle a junction-based routing algorithm [4]. Simple routers do not need to modify routing information to select the output port for an incoming packet. A junction-based router consists of a number of components such as "I/O buffers", "cross bar", "arbitration and control unit" etc. In the following, we describe these components briefly.

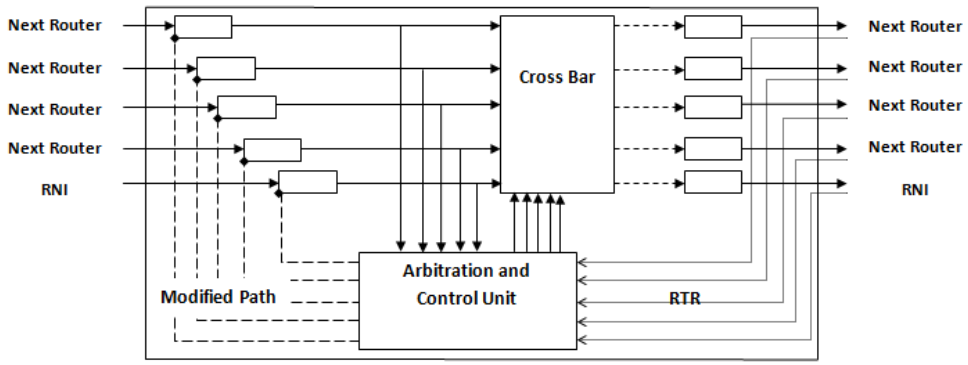


Figure 4-12.Block diagram of a junction-based router

4.6.1 Main Blocks of a Junction-Based Router

I/O Buffers

Flits received from RNI and other routers are stored (buffered) in the input buffers before routing. When a flit is forwarded from a buffer, a signal is sent back to the previous router and another flit can be accepted by the input buffer.

Flits are latched in output buffers before transferring to the next router or to the RNI. As soon as a flit is moved out from an output buffer, the buffer sends a signal to the "Arbitration and control unit" indicating it is ready to receive (RTR) the next flit.

Cross Bar

The input and output ports are connected through the crossbar which consists of five multiplexers and therefore, routing of messages is performed simultaneously when messages are headed for non-conflicting outputs. Cross Bar locks the path for BODY flits and END flits. Cross Bar is controlled by "Arbitration and control unit".

Arbitration and Control Unit

"Arbitration and control unit" determines the destined port for every HEAD flit. BODY and END flits follow the head flit then cross bar locks the path for these flits. Arbitration and control unit is described in more details in the next sub-section.

4.6.2 Arbitration and Control Unit

Arbitration and control unit consists of several blocks which are depicted in *Figure 4-13.* These blocks are described in this sub-section.

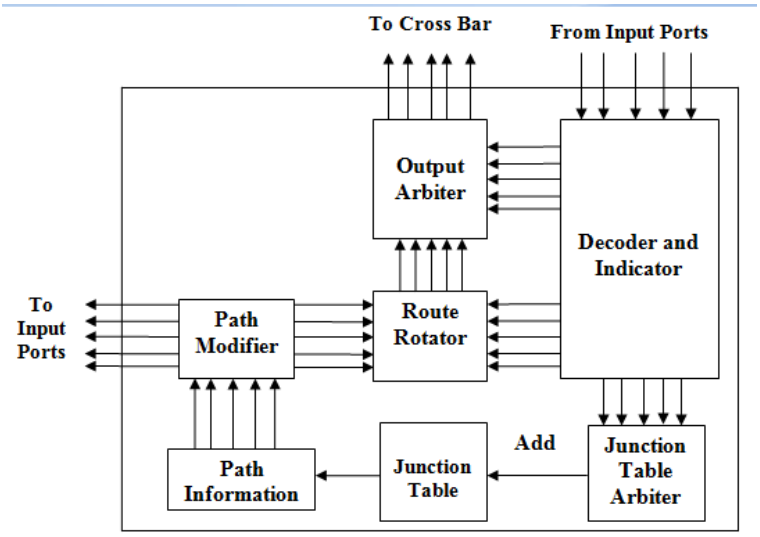


Figure 4-13.Arbitration and control unit in a junction-based router

Decoder and Indicator

This block decodes the "Flit Type" field and the 2-bit clockwise port address routing information of an incoming flit and sends the flit to the output arbiter or route rotator or junction table arbiter block. When a BODY or END flit is received, it is transferred to output arbiter block because the path was already locked by the HEAD flit. When "Flit Type" field is decoded and it is found that the flit is a HEAD flit then IndBit is decoded to determine whether the head flit consists of required path information to reach the destination or not. If yes, it just transfers the flit to route rotator block otherwise that, the first two bits of the route is decoded to determine whether the output port has been determined or not. When IndBit is equal to "1" and the first two bits of the route determines that the output port is RNI then routing information needs to be modified in this junction and required information shall be taken from the memory block. In this condition, the flit is transferred to junction table arbiter block.

Output Arbiter

Arbiter is used if there are several requests for the same output. Arbiter knows the status of all the ports. When a HAED Flit is transferred to the next router, arbiter locks the path for the body flits and the end flit of the packet. Arbiter also unlocks the path when the END flit has been transferred.

Junction Table

Junction table stores the pre-computed path information to give the information about rest of the route for the flits which does not have enough routing information to reach the destination or another junction.

Junction Table Arbiter

Junction table arbiter resolves contention problem for accessing the memory (junction table) block.

Path Modifier

Routing information and IndBit field of a HEAD flit is modified in this block after getting the information from junction table block. The head flit stored in the input buffer shall be updated before transferring to cross bar. When the path information is updated in the input buffer, path modifier transfers the head flit to route rotator block.

Route Rotator

Routing information in HEAD flit is rotated before sending HEAD flit to output arbiter block because the destined output port address shall be reside in a pre-specified places in the header flit [4].

4.7 Advantages and Disadvantages of JBR

Advantages and disadvantages of JBR are discussed in the following subsections. JBR has many advantages over source routing [4].

- The path information is limited into specified number of bits and is less than that in source routing but is more than that in distributed routing.
- Header flit in JBR can carry some payload due to less number of bits required for path information and it results in better performance in comparison with source routing.
- JBR is a good candidate for large networks due to relatively small path information.
- Specified number of bits for storing the routing information increases scalability. For different mesh sizes, it just needs to increase the number of junctions used. The path fits in the flit and the number of junctions is not comparable with the number of total nodes in the network. There are different configurations for junctions which can be useful in some aspects.
- In this thesis we store one path for each communicating pair in sources and junctions nodes and therefore in-order packet delivery of packets is guaranteed. This advantage is derived from source routing.
- Junction architecture is more complex than a simple router and junction needs a memory to store paths information and therefore junction is more costly than a simple router. The junction memory increases the size of a router as well.
- Delay in a junction is likely to be more than in a simple router.
- Sources nodes need to store the paths information in a memory that is costly. Size of the memory is also another issue that should be considered.
- JBR that we are using in this thesis is static and does not care about the network situation like the traffic pattern and broken links. But it is possible to store more than one path for each communicating pair in the sources and junctions nodes and also to apply some of techniques to increase fault tolerance of JBR.

5 Performance Evaluation of JBR

In this chapter, JBR for mesh topology NoC is modeled using SDL and packet delay for JBR is analyzed. The simulator is a modified version of an existing simulator for source and distributed routing [12]. JBR is evaluated using different routing algorithms and different types of traffics and the performance of JBR and source routing are compared.

5.1 Packet Delay Model for JBR

In previous chapters packet delay or latency has been defined but in this section packet delay is described in details and is analyzed for JBR specially. Packet delay or latency (T_{Packet}) is formulated by the following equation [4].

$$T_{Packet} = T_F + T_{QRNIR} + (n_1)\,T_{RL} + (n_2)\,T_{JL1} + (n_3)\,T_{JL2} + (k-1)\,T_L + [n + (k-1)]\,T_{RIO} + [(n+1) + (k-1)]\,T_{Link} + [n + (k-1)]\,T_{CO} + T_{HRRNI} + T_{QRRNI} + T_{DF}$$

Where,

n_1: Number of Simple Routers traversed by packet from Source to Destination

n_2: Number of Junctions traversed by packet from Source to Destination which are used for getting route information

n_3: Number of Junctions traversed by packet from Source to Destination which are not used for getting route information

k: Number of Flits in the packet

n: Total number of routers

$n = n_1 + n_2 + n_3$

T_L: The amount of time that is needed to transfer a flit from the input buffer of a router to the output buffer of the router. This router may be a simple router or a junction. The junction may act as a simple router or as a real junction using extra functionality.

$$T_L = [(n_1)\,T_{RL} + (n_2)\,T_{JL1} + (n_3)\,T_{JL2}] / n.$$

T_F, T_{DF}, T_{HRRNI}, T_{QRNIR} and T_{QRRNI}

Data is transferred in a packet format in a packet switched network and converted into flits while using worm-hole switching. Normally packets are flitized in an RNI. The resource sends a packet to corresponding RNI when transmit buffer of the RNI is completely free. T_{QRNIR} describes the delay due to the mentioned blocking communication. Flitization process also results in a certain amount of latency (T_F).

Flits are converted back into a packet format after receiving from the network. Deflitization is performed in RNI and results in a certain amount of latency (T_{DF}). But deflitization process performs when all the flits of a packet are received in the input buffer of RNI from the network then another kind of delay is introduced called T_{QRRNI}. When deflitization process is finished and the packet is delivered to the destination, an RTR (Ready To Receive) signal is sent to corresponding router by RNI. The router does not send flits of another packet until it receives this signal. T_{HRRNI} describes the delay due to the mentioned blocking communication.

T_{Link}

Speed of links between routers and between a core and corresponding router are much higher than routers speed but propagating a flit through a link introduces a delay called link delay.

T_{RIO}

A flit is transferred from the input buffer to the output buffer within a router when there is an empty place for one flit in output buffer.

T_{CO}

Flits are forwarded to the next router when there is a free space in the input buffers of the next router. Channel occupation results in TCO delay.

T_{RL}, T_{JL1} and T_{JL2}

A certain amount of time (**T_L**) is needed to transfer a flit from the input buffer of a router to the output buffer of the router. This delay happens due to decoding flit type and some control information, selection and switching activity inside the router and transfer of flit from input buffer to output buffer.

A router may be a simple router or a junction. The junction may act as a simple router or as a real junction. A simple router just read the path information that has been already stored in a header flit to decide which output port should the packet is sent. It does not take so much time (**T_{RL}**). In JBR some of the routers are special junctions rather than simple routers in normal source routing. When a flit arrives at a junction, the junction checks one or two bits to determine whether the header flit has enough information to be transmitted to one of the output port or not. If the path information is not complete, the junction uses its memory to fill up the header flit. Checking those mentioned bits and loading the information from the junction memory and updating the header flit introduces a delay that is larger than a delay in a simple router (**T_{JL1}**). Then in JBR, a junction determines the path for one or more than one hop. **T_{JL2}** presents the delay in a junction when it does not need to use its memory.

T_L is multiplied by the number of routers traversed by the flit while travelling from source node to destination node. The delay for a packet is calculated using the following formula:

$$[n+ (k-1)]\ T_L = [(n_1+ n_2+ n_3) + (k-1)]\ T_L = (n_1+ n_2+ n_3)\ T_L + (k-1)\ T_L =$$

$$(n_1)\ T_{RL} + (n_2)\ T_{JL1}+ (n_3)\ T_{JL2} + (k-1)\ T_L$$

Application-specific NoC offers the opportunity to decrease the delay in a junction by storing the path information for communicating pairs rather than all pairs of nodes in a network. Obviously size of the memory and overall cost of the junction are decreased. There are also some methods to compress memory tables in routers [15]. Junction architecture plays a central role in the performance of a junction based on-chip network.

5.2 Language Used for Modeling

Specification and Description Language (SDL) is used for modeling of JBR. Existing simulator that has been developed by Jönköping University Research Group is used for evaluating source and distributed routing and it uses SDL as well. JBR simulator is modified version of the simulator used in [12].

NoC simulator uses the graphical editor of SDL but textual Phrase Representation (SDL/PR) can also be provided by SDL

In SDL a system is specified as a set of blocks which communicate with each other using channels. A block consists of a number of processes which are extensions of finite state machines (FSM). A message (signal) is used for communicating between these processes.

A process can consist of a number of procedures and it can also call a procedure in another process. Therefore SDL can provide structure, communication, behavior, data, and inheritance aspects properly.

"Telelogic SDL and TTCN Suite 6.1" programming tool is used for modeling JBR. This tool fully supports SDL-92 [11].

5.3 Simulation of JBR

This section presents the main components in a JBR model. JBR is modeled using four main blocks. In the following the functionality of each block is explained and the assumptions made for simulation are mentioned. Input and output parameters are also described.

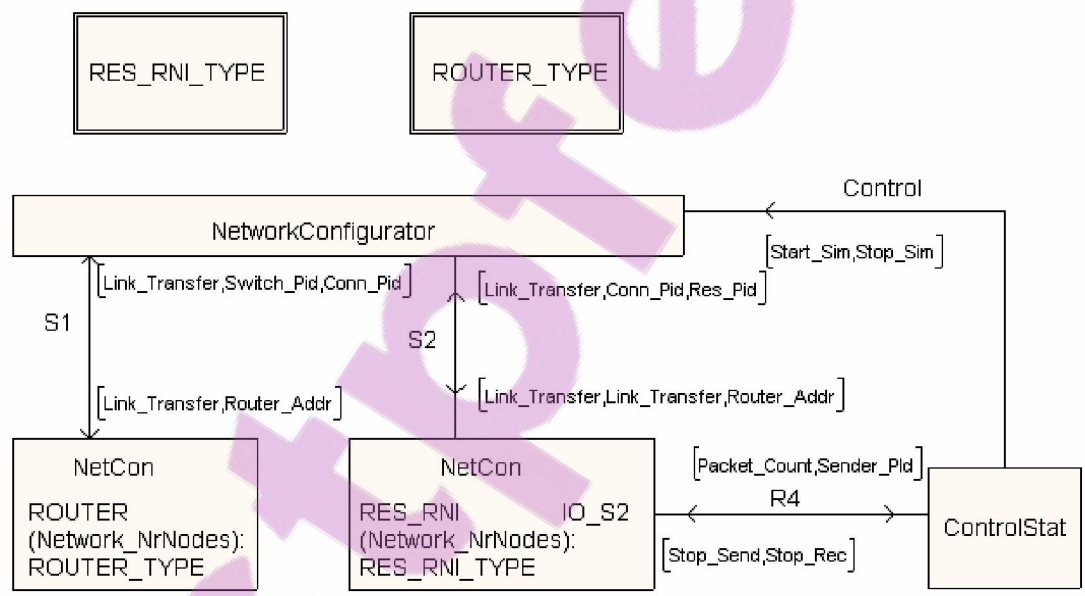


Figure 5-1.Top level system description of simulator in SDL

5.3.1 RES_RNI_TYPE

RES_RNI_TYPE models a core and corresponding RNI. There are 49 blocks of this kind for a 7x7 mesh NoC.

Cores are modeled as flit generators and consumers. All the flits of a packet are sent in a burst mode and time gap between two packet generations (BurstGap) can be set by users. Number of flits in each packet is fixed. This value can be set by users but the default value is 10. A path for each communicating pair is stored in every resource.

Different types of traffic are also modeled in this block. Traffic Type can also be selected by the user. The address of a resource is determined by NetworkConfigurator block. The simulation will run until 20,000 packets are received by all destination resources with warm up number of packets adjusted.

Resources send data in flit format and RNIs just send them to corresponding routers. Flits are converted back into a packet after receiving from network. A receiver core receives data in flit format. Deflitization is also done in resources in our simulator. Then RNI are modeled simply.

Packet Injection Rate (PIR)

The number of packets injected in the network per unit time per resource is presented as the load of the network. The user can also store different values of PIR in a file and use them for simulating.

Default PIR = 0.1

5.3.2 ROUTER_TYPE

ROUTER_TYPE models a router. This router can be a simple router or a junction. The addresses of routers are assigned by NetworkConfigurator block. Number and position of junctions shall be given by users.

Router Buffers Size

Sizes of router buffers are set by users.

Default Router Input Buffer Size=2 (flits)

Default Router Output Buffer Size=1 (flit)

The output buffer size is smaller than input buffer size. Normally link speed is much higher than the router speed but a flit cannot be sent to the next router until the input buffer of next router has a free space. Then output buffer is used to avoiding flit corruption.

5.3.3 NetworkConfigurator

First of all we assume the links are reliable and there is no need to retransmit a packet. There is also no loss and error in the network. For instance, network never contains broken links. No virtual channel is used.

NetworkConfigurator block configures the network. By receiving Start_Sim from ControlStat block, NetworkConfigurator block initializes the files that are used for reading and writing data needed in simulation and then starts keeping the process IDs of all the I/O buffers of routers, RNI and resources.

5.3.4 ControlStat

The network statistics are recorded by the ControlStat block. Total number of packets and flits received, latency, throughput, total simulation time etc. are recorded by the simulator. The simulation is started by sending Start_Sim from ControlStat block and

the statistics are collected after a certain amount of packets has been received by the destinations. This value is presented as Warm-up Number and can be set by user.

Default Warm-up Number of Packets = 2000

By receiving specified number of packets (TotalNrPackets) by the destinations, simulation is stopped by sending Stop_Sim signal to NetworkConfigurator block. TotalNrPackets can be set by users.

Default TotalNrPackets = 20,000

This block also sends Stop_Send signal and Stop_Rec signal to RES_RNI block as well. By arriving these two signals, resources stop sending and receiving the packets accordingly. ControlStat block writes the network statistics in some text files and the simulator displays the network statistics.

Global Average Flit Delay (Cycles)

The overall average time taken from the generation of flits at the sender resource to the reception of the flits by the receiver resource

Global Average Packet Delay (Cycles)

Average time taken by a packet to travel from source core to destination core

Global Average Throughput (Flits/Cycle)

Overall average number of flits received per cycle

Throughput (Flits/Cycle/IP)

Number of flits received in each cycle per core

5.4 Simulation Results

This section discusses simulation results for JBR and source routing. Various routing algorithms are analyzed using random traffic and local traffic with different Packet Injection Rate (PIR). Hop count limit is assumed to be 7.

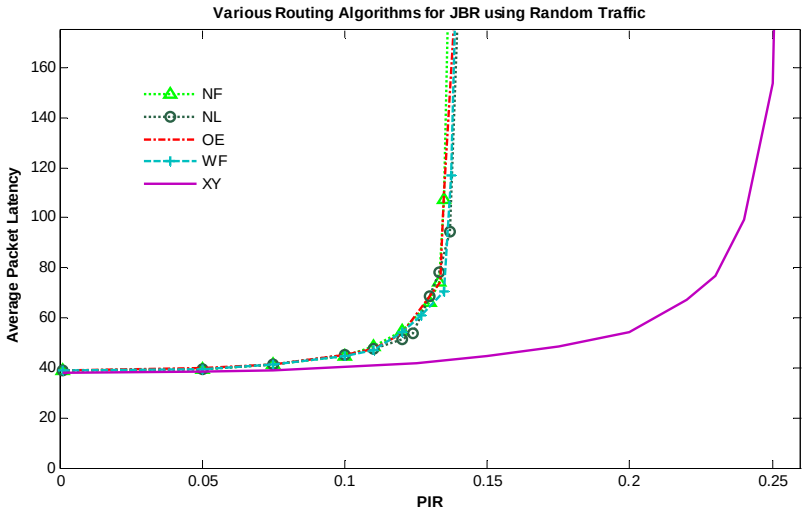
5.4.1 Evaluation of Performance of JBR

This section presents and analyzes the simulation results for JBR using various routing algorithms in random and local traffic.

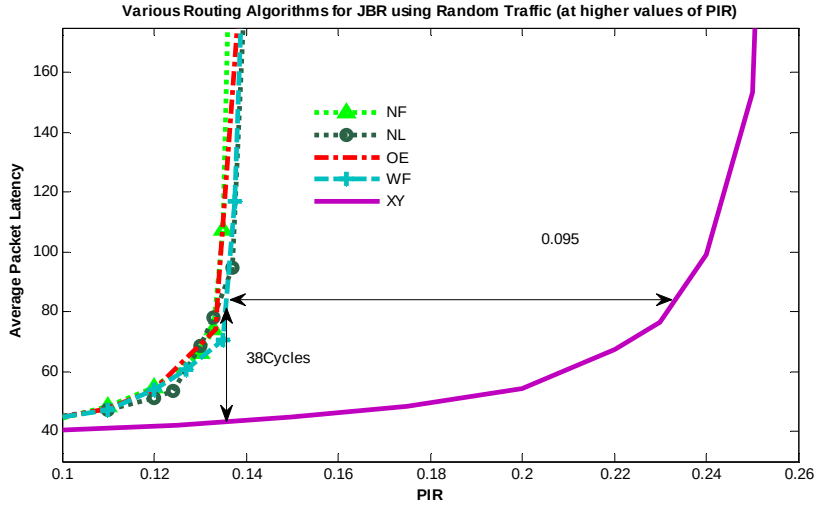
Random Traffic

As can be seen in *Figure 5-2*, average packet latency using XY routing algorithm is lower than four other routing algorithms. These results were predictable because XY routing algorithm performs the best in random traffic for source routing [4].

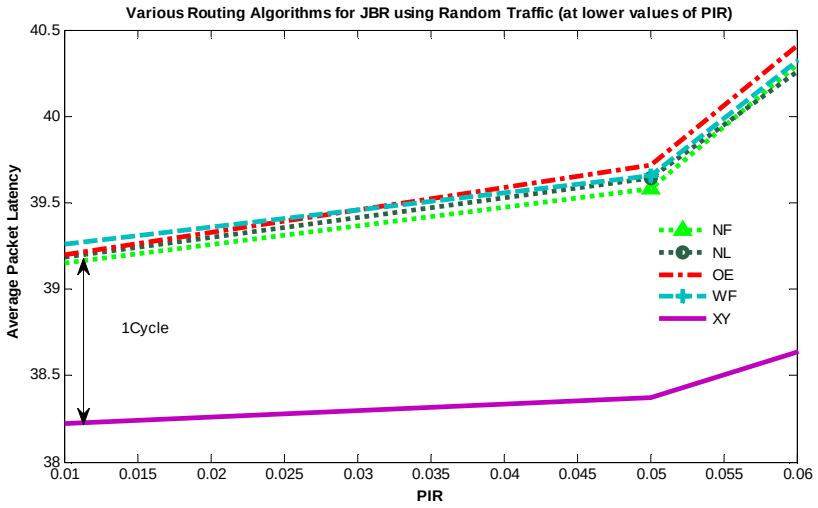
It is obvious from the figure that at lower network load, average packet latency using XY routing algorithm is about 1 cycle lower than that of other ones. Actually at high loads, the difference in latency given by XY and other routing algorithms becomes very high. In the case of West-First routing algorithm, when PIR value increases beyond 0.13, the latency increases very quickly. Using XY routing algorithm, the latency remains low until PIR reaches 0.23 when the network starts to saturate.



a) For different values of PIR



b) For higher values of PIR



c) For lower values of PIR

Figure 5-2.Average packet latency plotted against PIR in random traffic for JBR using various routing algorithms in a 7x7 mesh NoC

As can be seen in *Figure 5-3*, at lower values of PIR, throughput increases linearly and is almost equal for all types of routing algorithms. Using OE routing algorithm, when PIR is increased beyond 0.144, throughput stars to level off. Using XY routing algorithm throughput keeps on increasing linearly and starts to saturate beyond PIR equal to 0.26. Network starts to saturate at PIR equal to 0.145, 0.147 and 0.15 using West-First, Negative-First and North-Last routing algorithms, respectively.

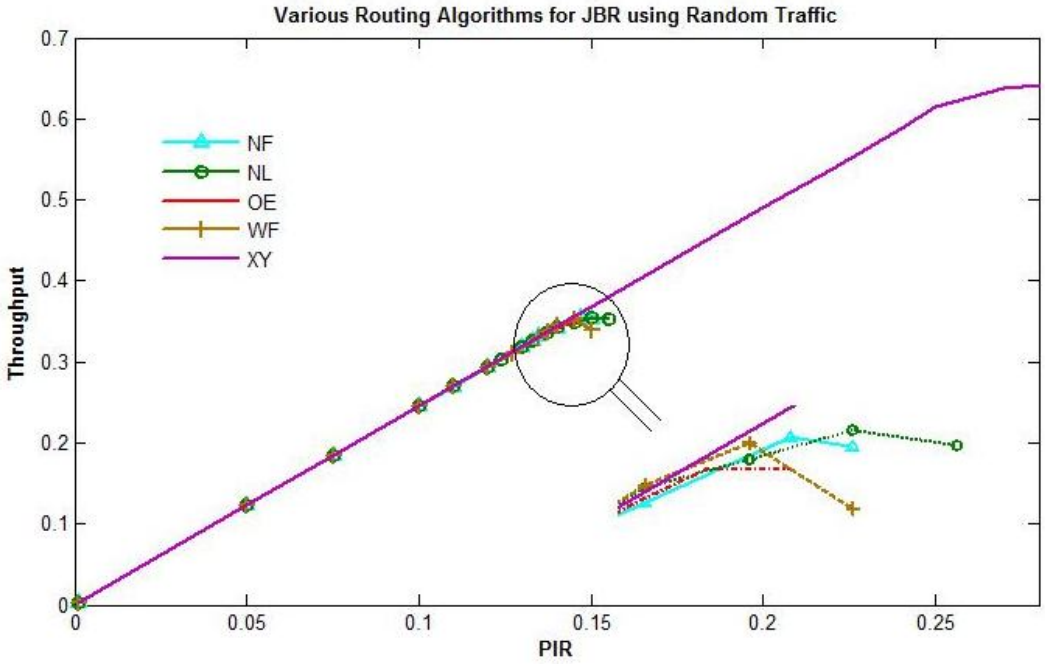


Figure 5-3.Throughput plotted against PIR in random traffic for JBR using various routing algorithms in a 7x7 mesh NoC

The network starts to saturate at relatively higher value of PIR using XY routing algorithm. Therefore XY routing algorithm gives much lower latency and provides higher throughput compared to that of four other ones and performs better than other routing algorithms at relatively higher load. At lower values of PIR, all types of routing algorithms perform almost the same and the advantage of XY is not visible at lower PIR.

In conclusion, XY routing algorithm performs the best and Odd-Even performs the worst in random traffic.

Local Traffic

As can be seen in *Figures 5-4* and *5-5*, XY routing algorithm performs the best in local traffic and Odd-Even and Negative-First routing algorithms perform the worst.

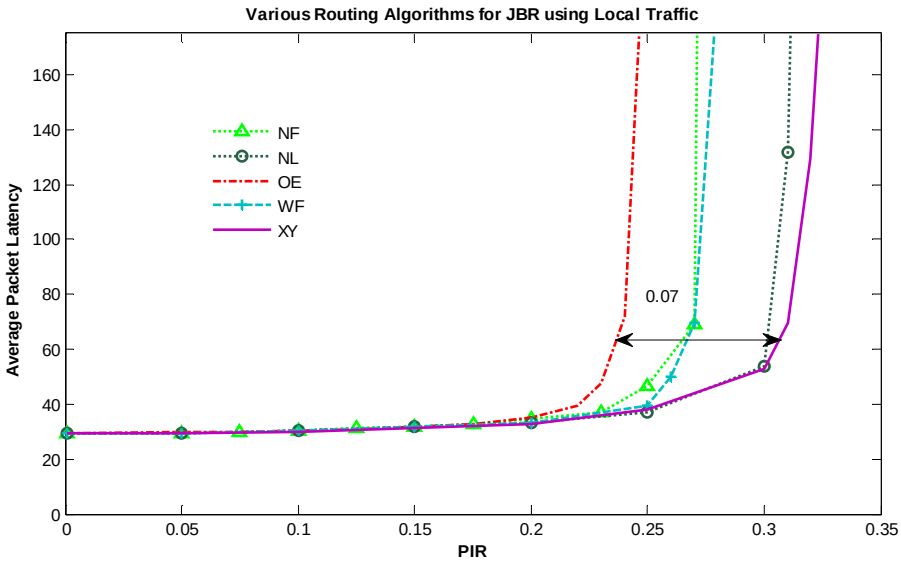


Figure 5-4.Average packet latency plotted against PIR in local traffic for JBR using various routing algorithms in a 7x7 mesh NoC

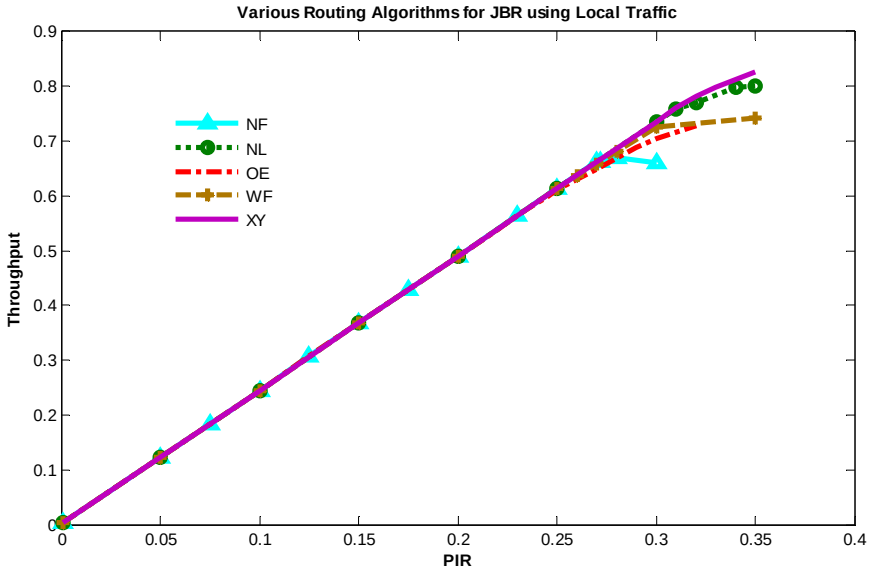


Figure 5-5.Throughput plotted against PIR in local traffic for JBR using various routing algorithms in a 7x7 mesh NoC

Based on our computations, using XY routing algorithm the number of junctions used in an experiment is less than that of using Negative-First routing algorithm. In local traffic number of junctions used in an experiment is less than that of in random traffic. Then in local traffic using junctions affect less compared to that of random traffic and the gap between latency graphs becomes less.

5.4.2 Performance Evaluation of Source Routing

This section briefly evaluates source routing for different routing algorithms in random and local traffic [4].

Simulation results in *Figure 5-6*, show that the average packet latency using XY routing algorithm is lower than other routing algorithms.

It is obvious from the figure that at high loads, the difference in latency given by XY and other routing algorithms becomes very high. For instance, in the case of West-First routing algorithm, when PIR value increases above 0.14, the latency increases rapidly. Using XY routing algorithm, the latency remains low until PIR reaches 0.24 when the network starts to saturate.

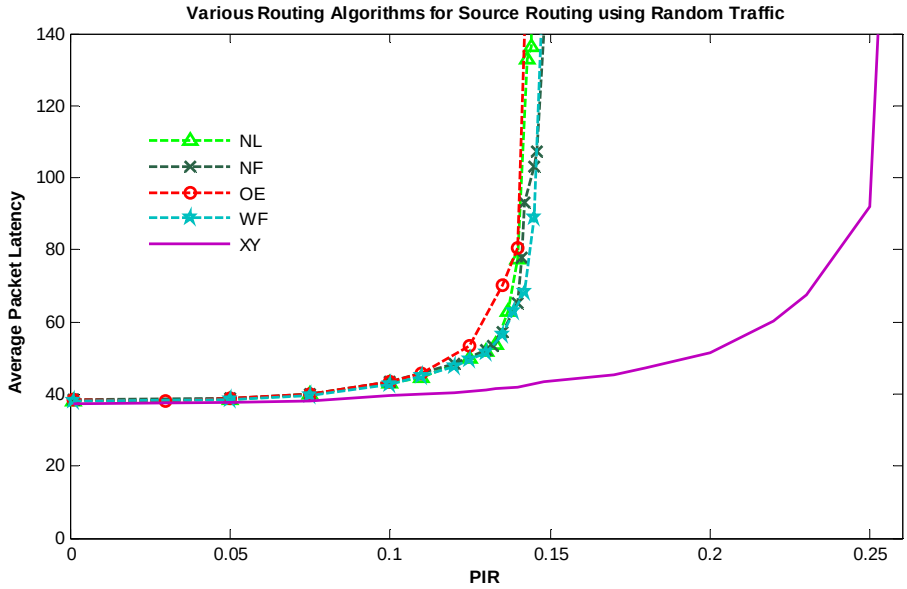


Figure 5-6.Average packet latency plotted against PIR in random traffic for source routing using various routing algorithms in a 7x7 mesh NoC

Simulation results in *Figure 5-7* show that at lower values of PIR, throughput grows linearly and is almost equal for all types of routing algorithms. But for example, when PIR is increased above 0.15, throughput using Negative-First routing algorithm starts to level off.

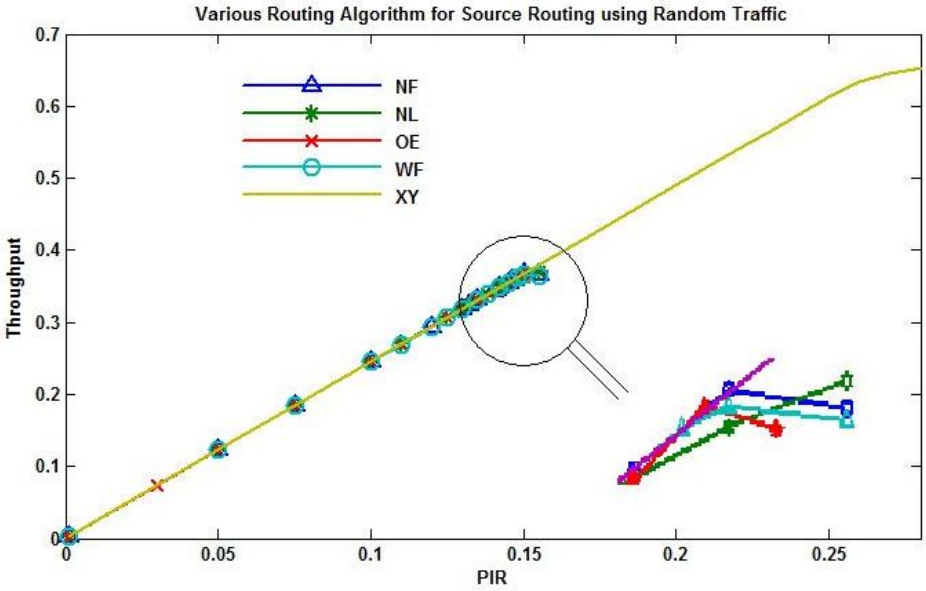


Figure 5-7.Throughput plotted against PIR in random traffic for source routing using various routing algorithms in a 7x7 mesh NoC

Using XY routing algorithm throughput keeps on increasing linearly and starts to saturate beyond PIR equal to 0.27.

The network starts to saturate at relatively higher value of PIR using XY routing algorithm. Therefore the best results with respect to latency and throughput are produced by XY routing algorithm. At lower values of PIR, all types of routing algorithms perform almost the same and the advantage of XY is not visible at lower PIR. Then XY routing algorithm can be used at higher network load while the NoCs using other routing algorithms cannot almost continue to work.

Local Traffic

As can be seen in the *Figures 5-8 and 5-9*, XY routing algorithm performs the best in local traffic and Odd-Even and Negative-First routing algorithms perform the worst.

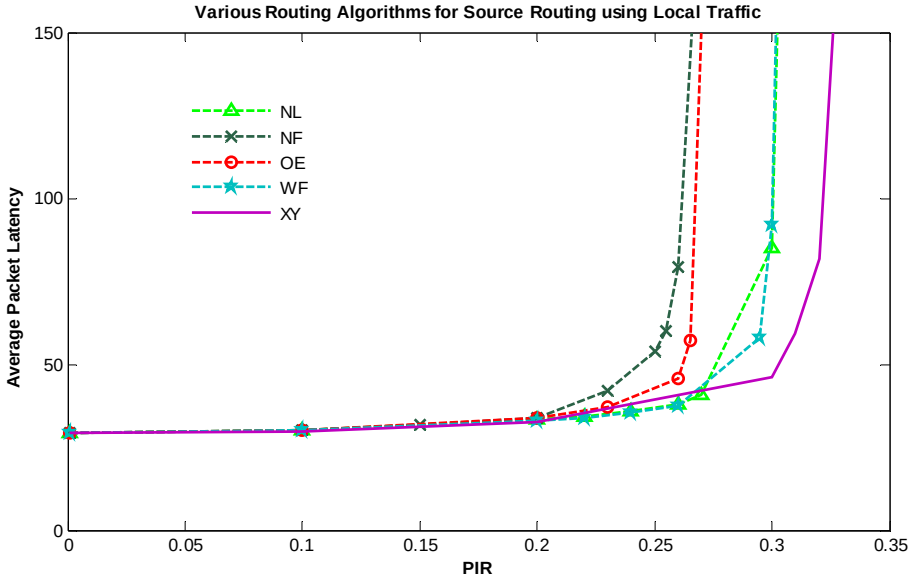


Figure 5-8.Average packet latency plotted against PIR in local traffic for source routing using various routing algorithms in a 7x7 mesh NoC

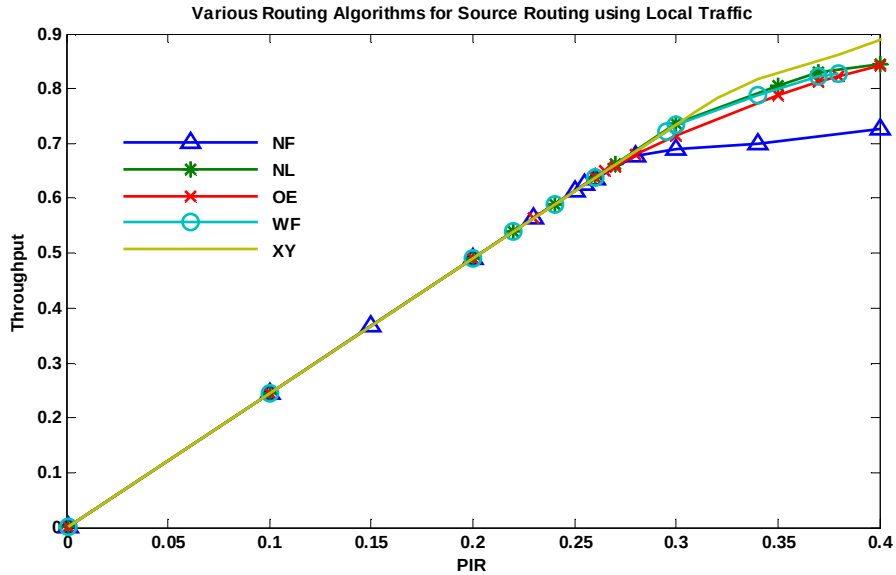


Figure 5-9.Throughput plotted against PIR in local traffic for source routing using various routing algorithms in a 7x7 mesh NoC

5.4.3 Comparison of JBR and Source Routing

This section discusses and compares simulation results for JBR and source routing. Various routing algorithms are analyzed using random traffic with different Packet Injection Rate (PIR).

XY Routing Algorithm

In this section, comparison of JBR and source routing in terms of average packet latency and throughput using XY routing algorithm in random traffic is presented. These two parameters are plotted against different values of PIR in *Figure 5-10* and *5-11* respectively.

Average packet latency of source routing (dashed curve) is lower than that of JBR (dotted curve). Latency in a junction is more than a simple router and that adds a delay to overall delay in a network.

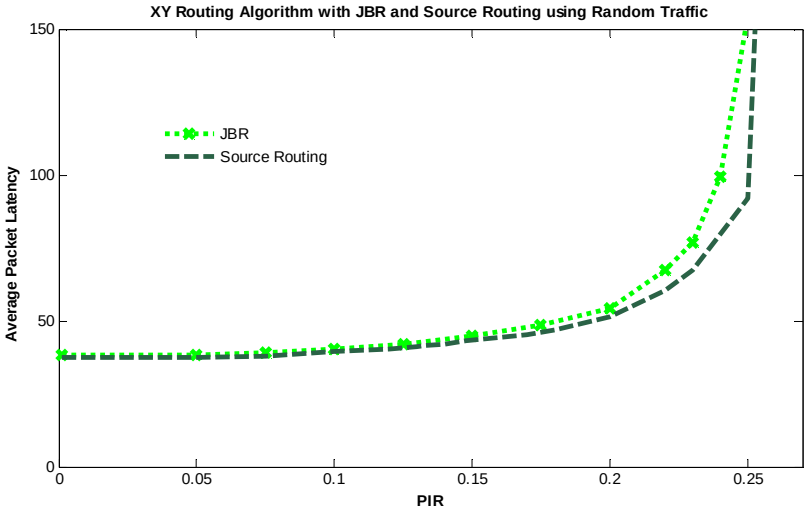
As can be seen in *Figure 5-10*, average packet latency in source routing is about 1 cycle lower than that of JBR at lower values of PIR. In case of JBR, when PIR value increases beyond 0.22, the latency increases very quickly. In case of source routing, the latency remains low until PIR reaches 0.23 when the network starts to saturate.

The difference in latency given by JBR and source routing is not high. Actually at high loads, the difference in latency given by JBR and source routing becomes about 8 cycles (the difference in latency for the load when the network using JBR starts to saturate).

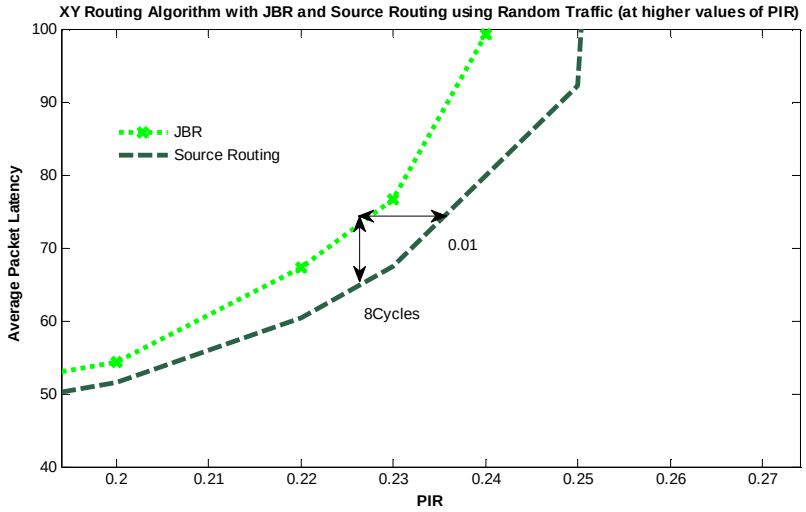
Throughput is plotted against different values of PIR for both JBR and source routing and is shown in *Figure 5-11*. At lower values of PIR, throughput increases linearly and is almost equal for both types of routing. When PIR is increased beyond 0.27, throughput in case of JBR stars to level off. In case of source routing throughput keeps on increasing linearly and starts to saturate beyond PIR equal to 0.28.

Then the network using XY routing algorithm, gives almost the same latency and throughput in case of JBR and source routing.

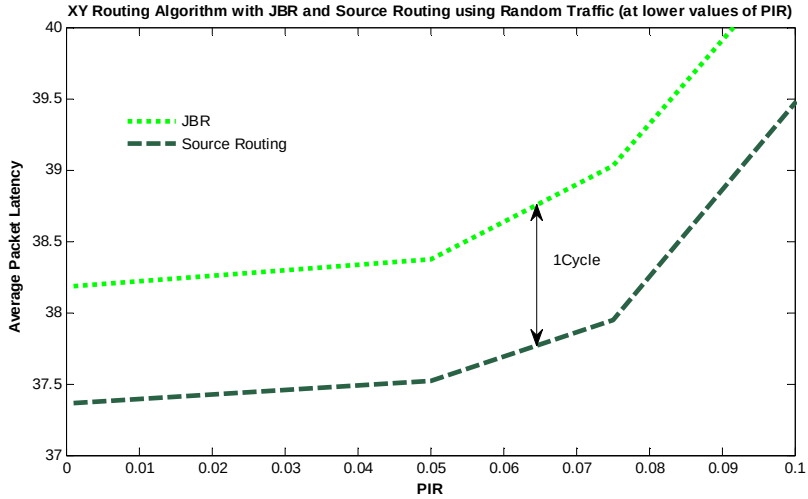
In average 21% of communicating pairs use one junction (504/2401= 0.21).



a) For different values of PIR



b) For higher values of PIR



c) For lower values of PIR

Figure 5-10.Average packet latency plotted against packet injection rate for a 7x7 NoC for JBR and source routing *using XY routing algorithm*

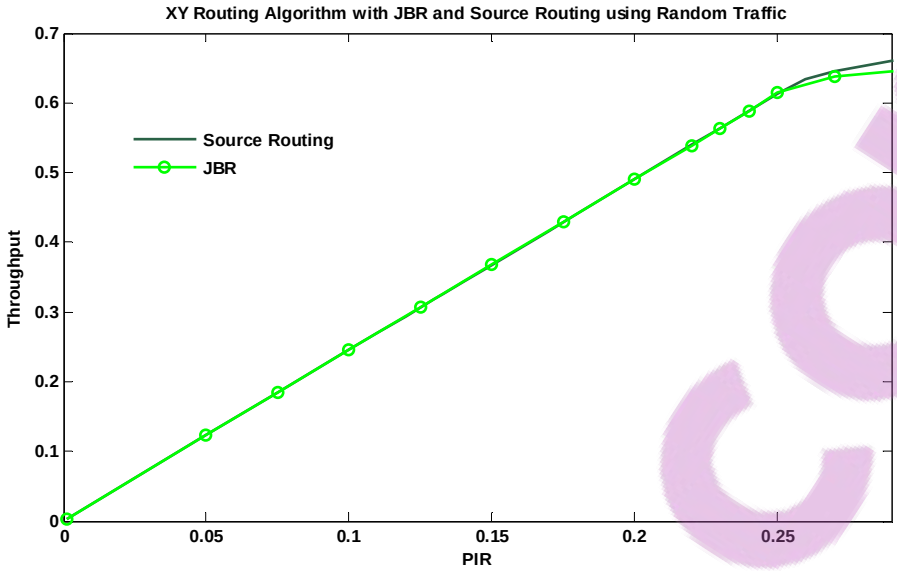


Figure 5-11.Throughput plotted against packet injection rate for a 7x7 NoC for JBR and source routing using XY routing algorithm

Negative-First Routing Algorithm

In this section, comparison of JBR and source routing in terms of average packet latency and throughput using Negative-First routing algorithm in random traffic is presented. Average packet latency of source routing is lower than that of JBR as can be seen in *Figure 5-12*.

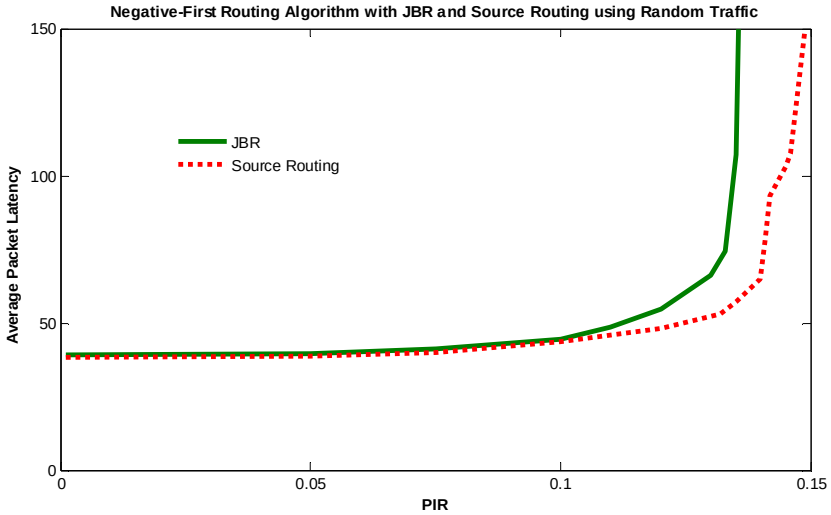
In the case of JBR, when PIR value increases beyond 0.13, the latency increases very quickly. In the case of source routing, the latency remains low until PIR reaches 0.14 when the network starts to saturate.

Actually at high loads, the difference in latency given by JBR and source routing becomes about 17 cycles (the difference in latency for the load when the network using JBR starts to saturate).

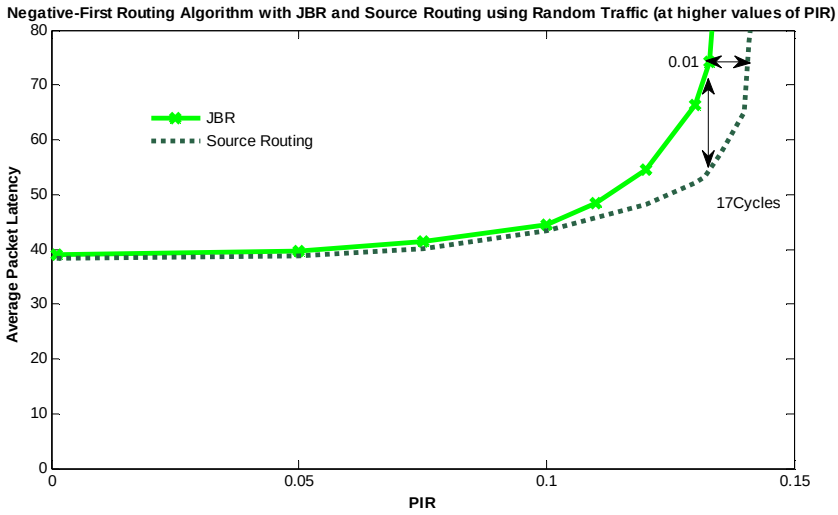
Then the network using Negative-First routing algorithm, gives a little worse performance in case of JBR in compare with source routing.

In this case 6 junctions are used. In average 20% of communicating pairs use one junction (493/2401=0.2) and 0.4% of communicating pairs use two junctions (11/2401=0.004).

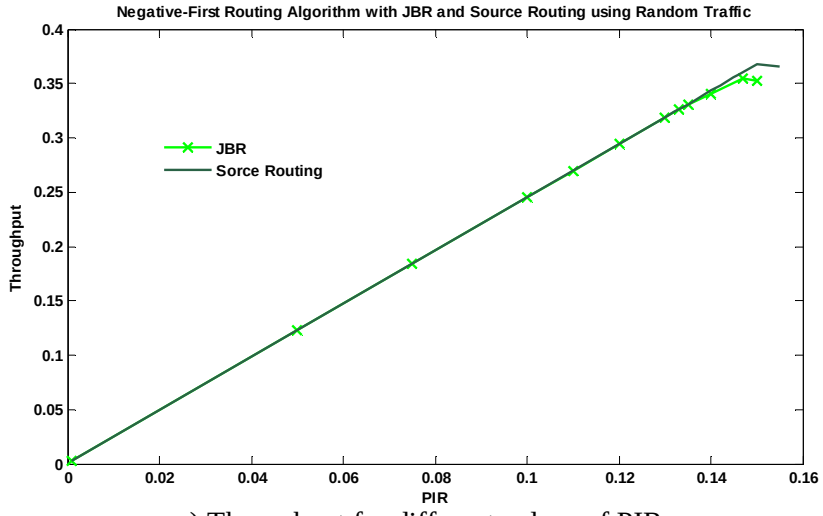
Throughput is plotted against packet injection rate for JBR and source routing as illustrated in *Figure 5-12(c)*.



a) Average packet latency for different values of PIR



b) Average packet latency for higher values of PIR

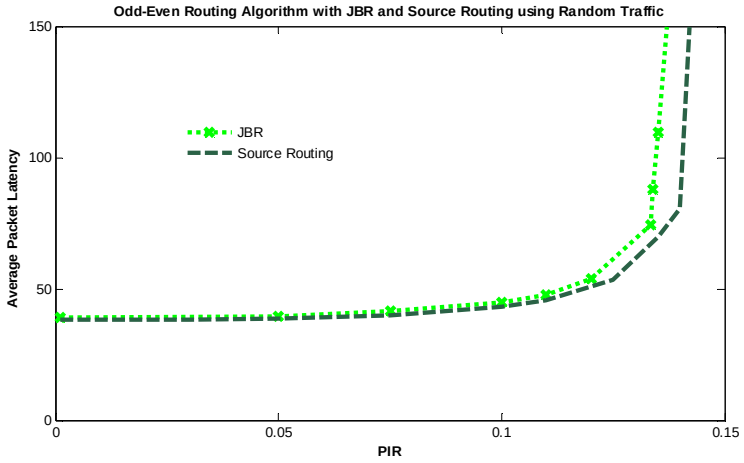


c) Throughput for different values of PIR

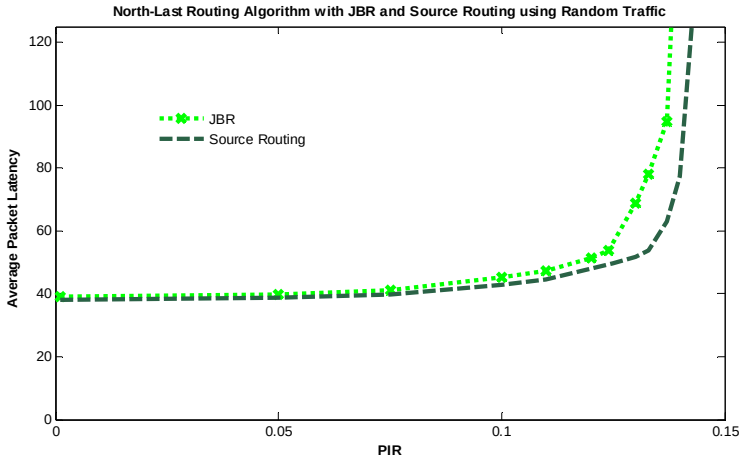
Figure 5-12.Average packet latency and throughput plotted against packet injection rate for a 7x7 NoC for JBR and source routing using Negative-First routing algorithm

Other Routing Algorithms

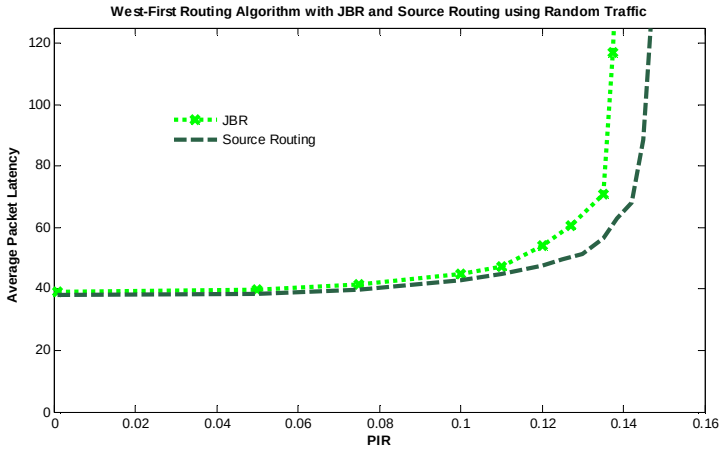
As can be seen in *Figure 5-13* and *5-14*, the difference in latency and throughput given by JBR and source routing is not high.



a) Odd-Even routing algorithm

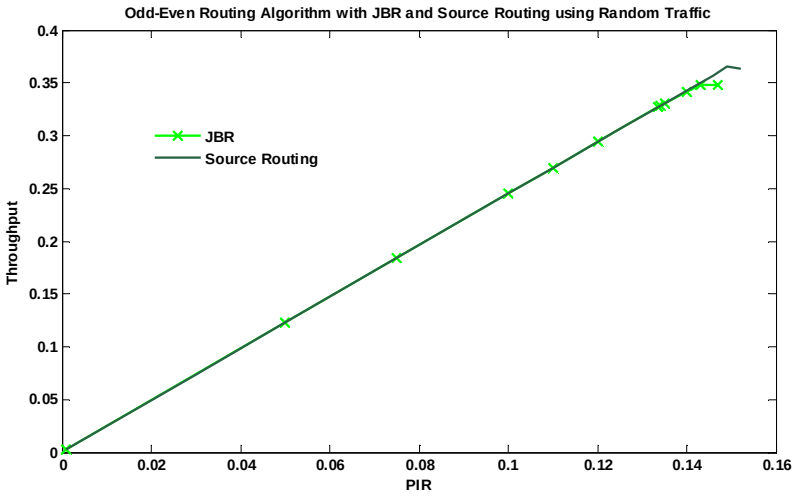


b) North-Last routing algorithm

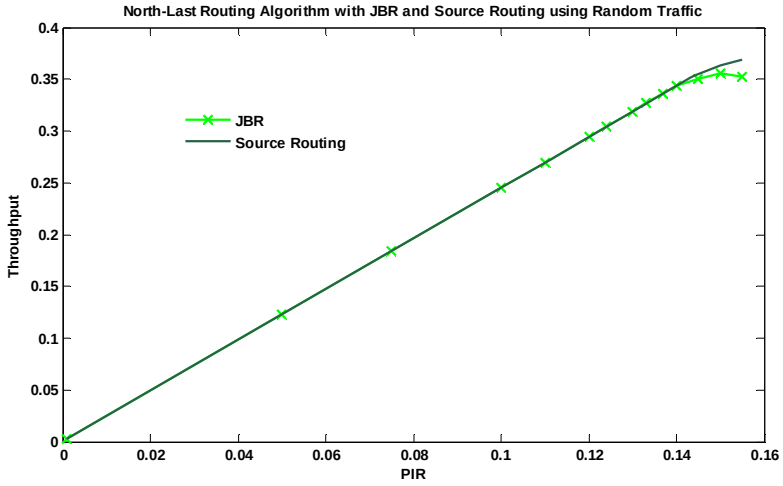


c) West-first routing algorithm

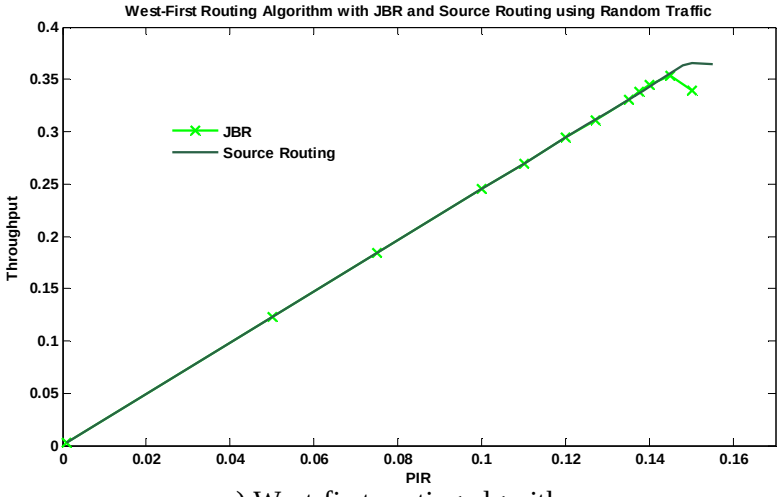
Figure 5-13.Average packet latency plotted against PIR for JBR and source routing using various routing algorithms in a 7X7 mesh NoC



a) Odd-Even routing algorithm



b) North-Last routing algorithm



c) West-first routing algorithm

Figure 5-14.Throughput plotted against PIR for JBR and source routing using various routing algorithms in a 7X7 mesh NoC

5.4.4 Importance of Smaller Routing Information

Consider a 7x7 network with a hop count limit of 7. If we consider a packet consists of 10 flits and end flit holds 3 bytes of payload, then maximum amount of data (payload) that is carried by each packet is 280 bits using source routing and 291 bits using JBR.

The following formula is used for calculating maximum amount of data (payload) that is carried by each packet.

Maximum amount of data (payload) = Number of bits of data in Head flit + Number of bits of data in Body flit + Number of bits of data in End flit.

Then, maximum payload is

Using source routing:

3*8+8*4*8=280.

Using JBR:

3*8+8*4*8+11=291.

Therefore using JBR, larger payload is possible than when using source routing.

We define two parameters which are used to compare JBR and source routing from another point of view. These two parameters are described as follows:

Global Average Bit Latency (Cycles)

Average time taken by a bit to travel from source core to destination core

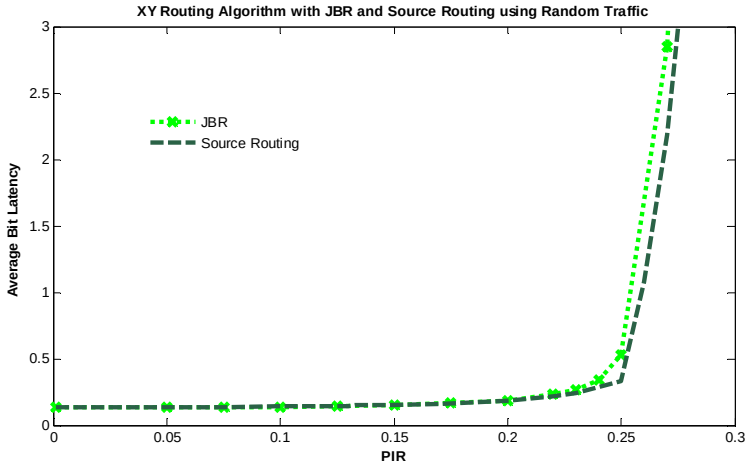
Throughput (Bits/Cycle/IP)

Number of bits received in each cycle per core

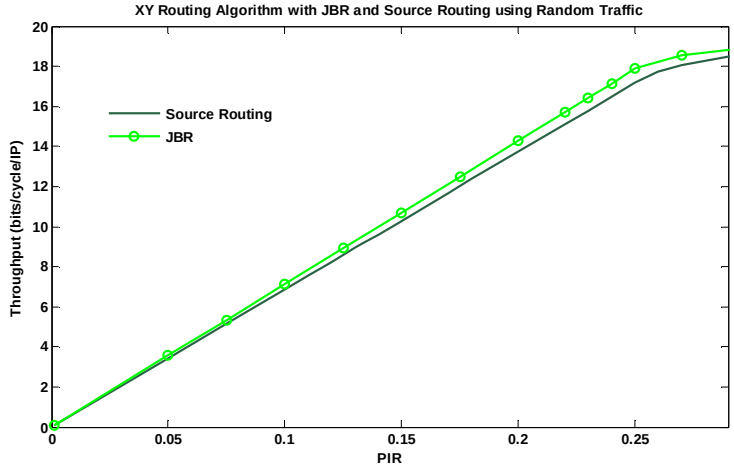
In these definitions “Bits” represents bits of actual data (payload).

Figure 5-15 shows graphs in which latency and throughput are plotted against different values of packet injection rate using XY routing algorithm for JBR and source routing in random traffic.

As can be seen in *Figure 5-15*, source routing provides lower latency (Cycle) compared to that of JBR but the difference in latency given by source routing and JBR is not high. JBR provides higher throughput (Bits/Cycle/IP) compared to that of source routing because a head flit can carry payload of data in JBR.



a) Average bit latency



b) Throughput

Figure 5-15.Average bit latency and throughput (bits/cycle/IP) plotted against packet injection rate for a 7x7 NoC for JBR and source routing using XY routing algorithm

In JBR header flit can carry payload and senders does not need to send one flit more for a few bits more data, which happens in source routing. For instance, header flit can carry 5 to 11 bits payload of data in a 7x7 network and a hop count limit of 7. In source routing, for sending this amount of data, one more flit shall be sent by senders. In this section, XY routing algorithm is analyzed using local traffic with different Packet Injection Rate (PIR) for JBR and source routing. The packet size is equal to 1 flit for JBR and is equal to 2 flits for source routing.

Average packet latency and throughput are plotted against different values of PIR in *Figure 5-16* and *5-17* respectively.

Average packet latency of JBR (dotted curve) is lower than that of source routing JBR (dashed curve). Packets in source routing consist of one flit more than that in JBR as mentioned before. Therefore performance improves in JBR.

As can be seen in *Figure 5-16*, average packet latency in JBR is lower than that of source routing at lower values of PIR. At high loads, the difference in latency given by JBR and source routing becomes more (the difference in latency for the load when

the network using JBR starts to saturate). The network using JBR starts to saturate at higher value of PIR as well.

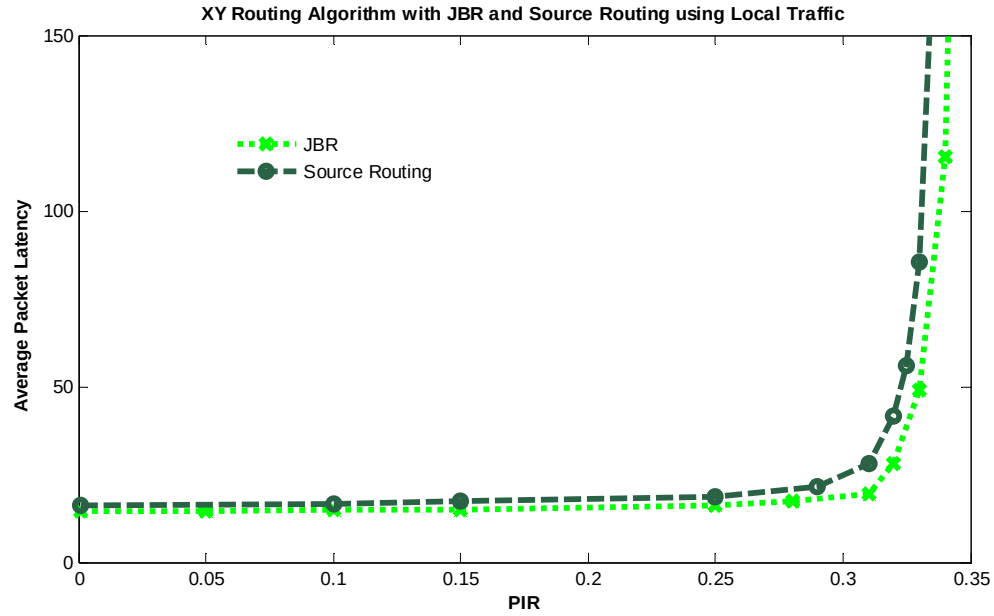


Figure 5-16.Average packet latency plotted against packet injection rate for a 7x7 NoC for JBR and source routing using XY routing algorithm

Throughput is plotted against different values of PIR for both JBR and source routing and is shown in *Figure 5-17*. Throughput is much better in JBR in comparison with source routing and throughput in case of JBR stars to level off in higher value of PIR as well. In case of source routing throughput starts to saturate at lower value of PIR.

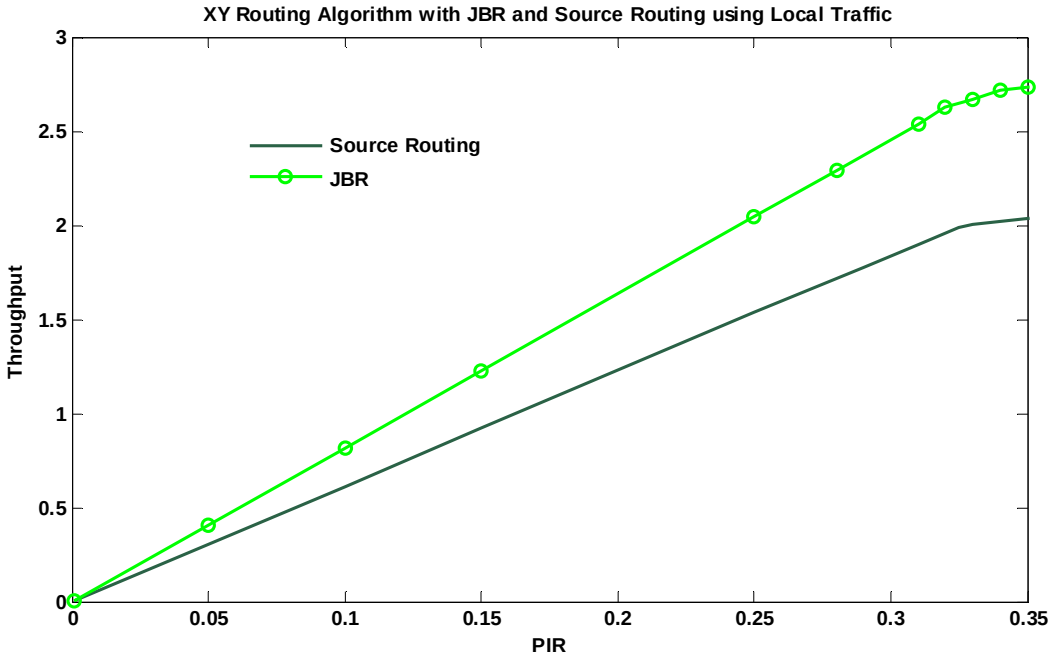


Figure 5-17.Throughput plotted against packet injection rate for a 7x7 NoC for JBR and source routing using XY routing algorithm

Then the network using XY routing algorithm, gives better latency and throughput in case of JBR. This result supports the main advantage of JBR over source routing.

6 Conclusions

Source routing, with many advantages (including lower router cost and higher communication speed) over distributed routing, is a good candidate for mesh topology NoC platforms [4]. But source routing is not considered scalable and efficient for large networks since the overhead of appending path information in the packet header increases with network size. In this thesis we have developed a new technique which will make source routing in large NoCs systematic, scalable and efficient.

6.1 Contributions and Results

We proposed a new technique called Junction-Based Routing (JBR) and completed the theory regarding the idea of this new technique. The idea of JBR is general and will be applicable to all topologies: regular or irregular, but we applied this idea to mesh topology NoC.

6.1.1 Junction-Based Routing

Concepts and issues of JBR were discussed and Junction-Based Routing was analyzed in terms of packet format, path information, header overhead etc. Using JBR, a large distance can be covered by going through intermediate temporary destinations (called Junctions) such that sub-paths (from source to a junction, junction to another junction and junction to the destination) are always smaller than or equal to a maximum hop count. We showed that the overhead in JBR grows very slowly and therefore it is more scalable. We also compared the overhead and the number of bits needed for routing in the case of source routing, distributed routing and JBR.

6.1.2 Number and Position of Junctions

Given the limit on the allowed number of bits in the header flit for storing path information, a minimum number of junctions will be required to be placed in the network. Minimum number of junctions and their positions were analyzed manually and also using a function developed in MATLAB. We presented some possible configurations of these junctions which are required for a 7x7 network and a given hop count limit. Using the function, we analyzed different configurations for various network sizes and different hop count limits. We showed that the number of junction(s) is not comparable with the number of nodes in the network and the number of junctions grows slowly with the decrease in hop count limit.

We also showed that having multiple configuration of junctions for a given path length can be useful for satisfaction of some other criteria like layout uniformity and optimization of performance in the context of application specific communication [10].

6.1.3 Increase in Path Length by Using Junctions

We discussed the increase in path length while the minimum number of junctions is used. For better analysis of increase in path length, we developed another function in

MATLAB. The function computes total overhead for all possible configurations for a given junction-based mesh network. We analyzed the average increase in hop count for uniform random traffic and application specific traffic favoring locality. Based on our experiments and computations, the average increase in hop count for different configurations is negligible. It is possible to find the best possible configuration of junctions for a given communication traffic of the application to minimize the increase in extra hop count. We showed some of the best configurations of junctions in a 7x7 NoC using different hop count limits for different traffic types.

6.1.4 Path Computations for Mesh Topology Network on Chip with Junctions

We showed that Turn-model routing algorithms cannot be applied in the networks while using minimum number of junctions. We investigated different methods for deadlock-free routing in junction-based networks and we decided to increase the number of junctions and use Turn-model routing algorithms. We also analyzed this new situation in terms of number and position of junctions. We showed that using turn-model routing algorithms and minimal paths, extra overhead is zero.

The required number of junctions and the position of these junctions are computed using a tool developed in MATLAB for different Turn-Model routing algorithms and a given network size and a given hop count limit. This tool also computes all the paths for all the pairs in the network using JBR.

6.1.5 Analysis of Junction-Based Networks Using Different Turn-Model Routing Algorithms

We analyzed different configurations of junctions for each routing algorithm and showed some of the results for each routing algorithm. We compared different routing algorithms in junction-based networks in terms of number of junctions and other parameters. Generally the number of junctions is much smaller than the total number of nodes. We showed that the number of admissible paths for the communicating pairs in a junction-based network is not much smaller than that in a network using normal source routing.

6.1.6 Link Load Distribution

Link load distribution was analyzed and one path for each communicating pair was selected for different types of communications and traffic patterns using a tool developed in MATLAB. Input parameters are network size, hop count limit, routing algorithm and traffic pattern or application specific communication. Output parameters are one path for all communicating pairs and standard deviation of links load, mean value of links load, maximum and minimum value. The method used for analyzing links load was compared with the other methods used in [4]. We showed that this method distributes the network load better. We also showed that the number of routes that uses the junction for getting routing information is not high.

6.1.7 Packet Format in JBR and Paths Encoding

Packet format in JBR was presented and a method was suggested for encoding paths information. Number of bits used for encoding is one more than that in the case of source routing [4]. Architecture of a junction in JBR was discussed a little. Focus should be on decreasing delay in a junction as well as keeping it simple and cheap as much as possible.

6.1.8 Simulation of JBR

Packet delay model in JBR was analyzed and an existing simulator in SDL [12] was modified for modeling JBR. Format of the stored paths which are output of the developed tool are not suitable for using as inputs to the simulator and they were converted into the proper format. Number and positions of junctions should be given as input to the simulator as well as some other parameters that were mentioned in chapter five.

6.1.9 Simulation Results

Performance of JBR was evaluated for various routing algorithms in random and local traffics and we found out the best routing algorithm for each traffic type. Some of the results were shown as the graphs that plot average packet latency and throughput against different values of packet injection rates. JBR was compared with source routing and the simulation-based results showed that latency does not increase so much using junctions. Throughput also does not level off significantly. We showed that JBR performs better than source routing if the header flit can carry some bytes of data (payload). Then JBR can be a good candidate for routing in NoCs with large size.

6.2 Limitations

Delay in a junction is more than a simple router and a junction is costlier than a simple router. Junction architecture is more complex than a simple router and junction needs a memory to store paths information. The junction memory increases the size of a router as well. A good design for junction architecture can compensate this disadvantage but it is not addressed in this thesis.

We could not estimate the delay in a junction accurately because we did not make a prototype of a junction. There are also some limitations related to performance evaluation of JBR in this thesis.

6.3 Future Work

There are many interesting work related to JBR that we plan to do in future. Here, we mention some of them briefly.

- **Zero Path Length Overhead**

Increase in path length is unavoidable while the minimum number of junctions is used. One of the future work in JBR is, finding the required number of junctions such that path length does not increase using junctions.

- **Using Virtual Channels**

In this thesis, we did not use virtual channels in junction-base networks. Using virtual channels for JBR can result in better performance and is one of our future work.

- **Using Non-minimal Routing**

In this thesis, we did not apply non-minimal routes. Using non-minimal routes can result in better link load distribution in a network.

- **Providing Multiple Paths in JBR**

It will be interesting as a future work to keep more than one path for each communicating pair in the sources and junctions that paths are selected considering the current network condition. This could also provide some tolerance to link or router faults.

- **Prototyping of Routers for JBR**

We are planning to design and implement a junction to support our JBR ideas. The prototype will be carried out in FPGA.

There are many practical decisions, regarding design and implementation of JBR, which must be taken. These include finding the best packet format for JBR and the best packet and flit size, finding a proper hop count limit for a given network and applying junction-based technique to other types of network topologies are other ideas for the future work.

7 References

[1] Dally W.J. and Towles B. *Principles and Practices of Interconnection Networks.* Morgan Kaufmann Publishers an Imprint of Elsevier Inc. ISBN: 0-12-200751-4. 2004.

[2] Dally W.J. and Towles B. *Route Packets, Not Wires: On-chip Interconnection Networks.* In Design Automation Conference, Las Vegas, Nevada, USA, 2001. Pages 684-689.

[3] Holsmark R. *Deadlock Free Routing in Mesh Networks on Chip with Regions.* PHD Thesis, Department of Computer and Information Science, School of Engineering, Linköping University, Sweden, 2009.

[4] Mubeen S. *Evaluation of Source Routing for Mesh Topology Network on Chip Platforms.* Master of Science Thesis, Högskolan I Jönköping, 2009.

[5] Hangsheng W., Peh L. S. and Malik S. *Power-driven design of router microarchitectures in on-chip networks.* In Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2003. Pages 105-116.

[6] Bertels K., Cardoso J. and Vassiliadis S. *Reconfigurable Computing: Architectures and applications.* Second International Workshop, ARC, Delft, The Netherlands, 2006. Pages 299 and 689.

[7] Lionel M. Ni and McKinley. *A Survey of Wormhole Routing Techniques in Direct Networks.* IEEE Journal of Computer Science. February 1992.

[8] Holsmark R., Johansson A. and Kumar S. *On Connecting Cores to Packet Switched On-Chip Networks: A Case Study with Microblaze Processor Cores.* In IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, Slovakia, April 18-21, 2004.

[9] T. Bjerregaard and S. Mahadevan, *A survey of research and practices of Network-on-chip.* ACM Computing Surveys, vol. 38, 2006.

[10] Palesi M., Holsmark R., Kumar S. and Catania V. *A Methodology for Design of Application Specific Deadlock-free Routing Algorithms for NoC Systems.* In International Conference on Hardware-Software Co-design and System Synthesis, Seoul, Korea, October 22-25, 2006.

[11] Telelogic SDL Suite: Avaialable at: http://www.1.ibm.com/software/awdtools/sdlsuite/. [Accessed on August 1, 2010]

[12] Holsmark R. and Mubeen S. *NoC SDL Simulator: Description and User Interface Version 2009-08-20 0.11.* Department of Electronics, School of Engineering, Jönköping University, Sweden. 2009.

[13] FLI7540 Digital TV SoC and FLI2520 Video-Enhancement IC. Available at: http://www.st.com/stonline/. [Accessed on November 6, 2010]

[14] Network-on-Chip with High-speed Serial Links. Available at: http://ssl.kaist.ac.kr/2007/. [Accessed on May 1, 2011]

[15] Palesi M., Kumar S. and Holsmark R. *A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architectures*: SAMOS VI:

Embedded Computer Systems: Architectures, Modeling, and Simulation. Samos, Greece, July 17-20, 2006.

[16] Holsmark R., Kumar S., Palesi M. and Mejia A. *HiRA: A methodology for deadlock free routing in hierarchical networks on chip.* In Proceedings of the 2009 3rd IEEE/ACM International Symposium on Networks-on-Chip, 2009.

[17] Dally W. and Seitz C. *Deadlock-Free Message Routing in Multiprocessor Interconnection Networks.* Computers, IEEE Transactions, 1987. Pages 547-553.

8 Appendix: Extra Results Related to JBR

8.1 Number and Position of Junctions for Different Network Sizes and Different Hop Count Limits

In this section, we present some of results of number and position of junctions for a given network size and a hop count limit. In *Table 8-1*, we present all possible configurations of two junctions which are required for a 7x7 network and a 6 Hop Count Limit. As can be seen, number of junctions is 2 and number of different configurations of these junctions is equal to 40.

Table 8-1.Results: All possible configurations of two junctions which are required for a 7x7 network and a 6 Hop Count Limit

Configuration No.	Positions of Junctions	Configuration No.	Positions of Junctions
1	(1,3) and (5,4)	21	(3,4) and (7,3)
2	(1,4) and (5,4)	22	(3,4) and (7,4)
3	(1,4) and (6,4)	23	(3,4) and (7,5)
4	(1,5) and (5,4)	24	(4,2) and (3,6)
5	(2,3) and (5,4)	25	(4,3) and (3,6)
6	(2,3) and (6,4)	26	(4,3) and (3,7)
7	(2,4) and (5,4)	27	(4,1) and (4,5)
8	(2,4) and (6,3)	28	(4,1) and (4,6)
9	(2,4) and (6,4)	29	(4,2) and (4,5)
10	(2,4) and (6,5)	30	(4,2) and (4,6)
11	(2,4) and (7,4)	31	(4,2) and (4,7)
12	(2,5) and (5,4)	32	(4,2) and (5,6)
13	(2,5) and (6,4)	33	(4,3) and (4,5)
14	(3,1) and (4,5)	34	(4,3) and (4,6)
15	(3,2) and (4,5)	35	(4,3) and (4,7)
16	(3,2) and (4,6)	36	(4,3) and (5,6)
17	(3,4) and (5,4)	37	(4,3) and (5,7)
18	(3,4) and (6,3)	38	(5,1) and (4,5)
19	(3,4) and (6,4)	39	(5,2) and (4,5)
20	(3,4) and (6,5)	40	(5,2) and (4,6)

In *Figure 8-1*, we present the positions of junctions for a 7x7 mesh topology NoC and a given H of 3. As can be seen in the figure, number of required junctions is 9.

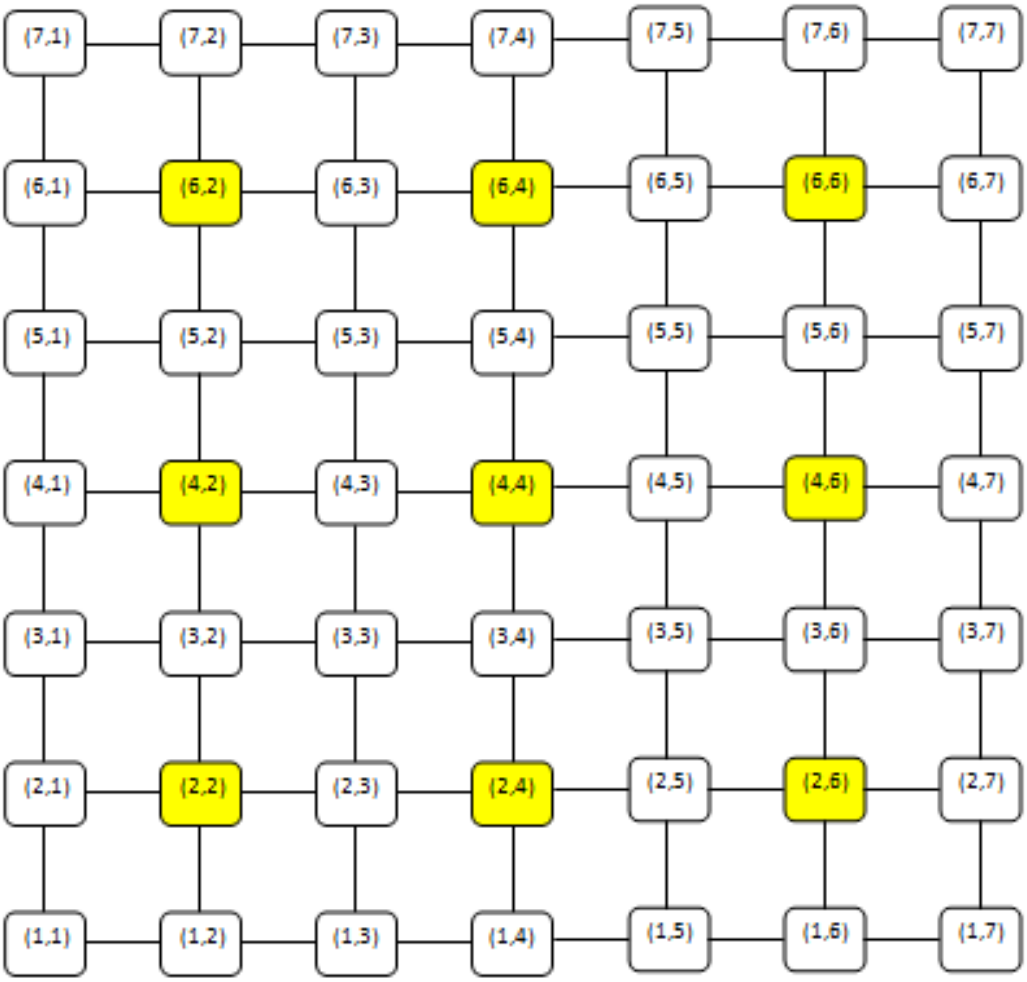


Figure 8-1 .A configuration of 9 junctions which are required for a 7x7 mesh topology NoC and a given H of 3.

In *Figure 8-2*, we present the positions of junctions for a 8x8 mesh topology NoC and a given H of 6. As can be seen in the figure, number of required junctions is 3.

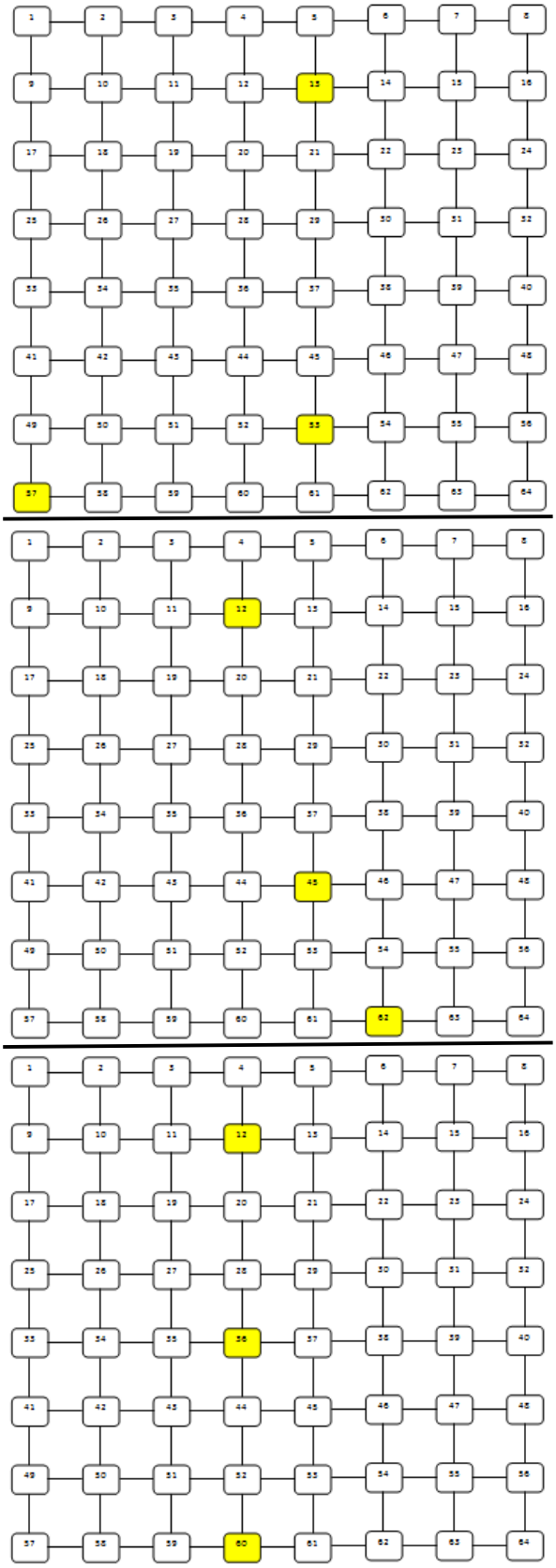


Figure 8-2 .Some of different configurations of 3 junctions which are required for a 8x8 mesh topology NoC and a given H of 6.

8.2 Junction Configurations for a 7x7 Mesh Network and a Given Hop Count Limit of 5

Figure 8-3 shows a possible configuration of junctions which are used in a 7x7 junction-based network with hop count limit of 5. The mesh NoC uses Negative-First routing algorithm and the number of junctions is 13.

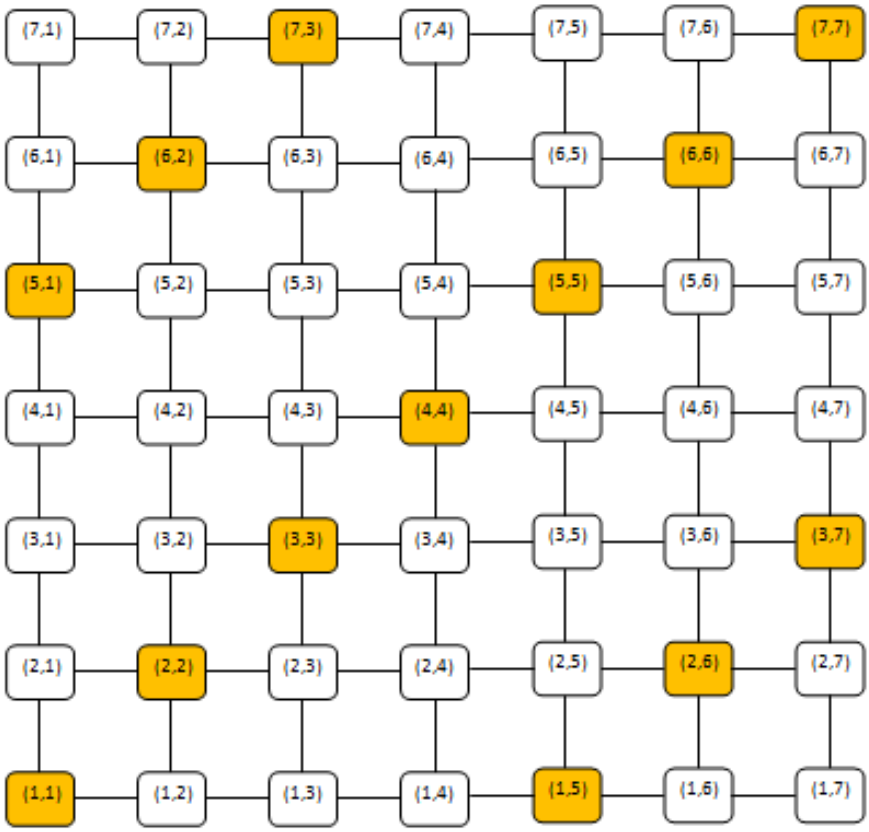
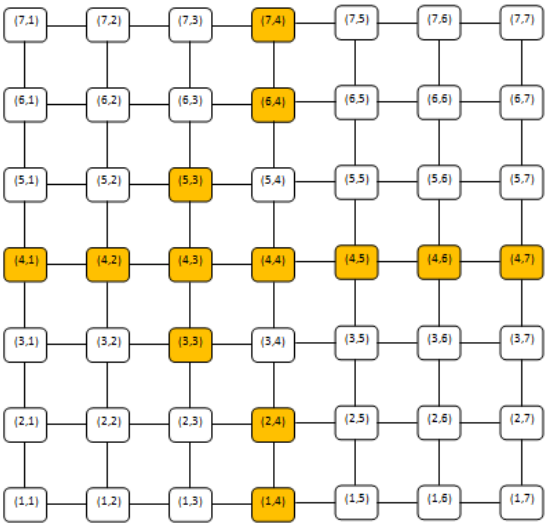


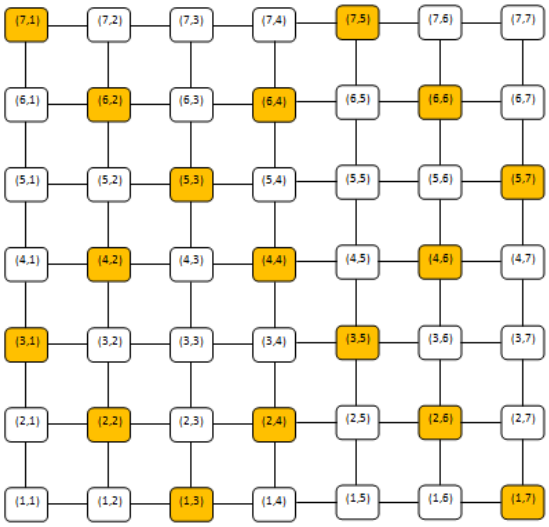
Figure 8-3.7x7 Junction-Based networks using Negative-First routing algorithms with hop count limit of 5

Figure 8-4 shows some possible configurations of junctions which are used in a 7x7 junction-based network with hop count limit of 5 while Odd-Even, XY, North-Last and West-First routing algorithms are applied. As can be observed, the number of junctions is not comparable with the total number of nodes.



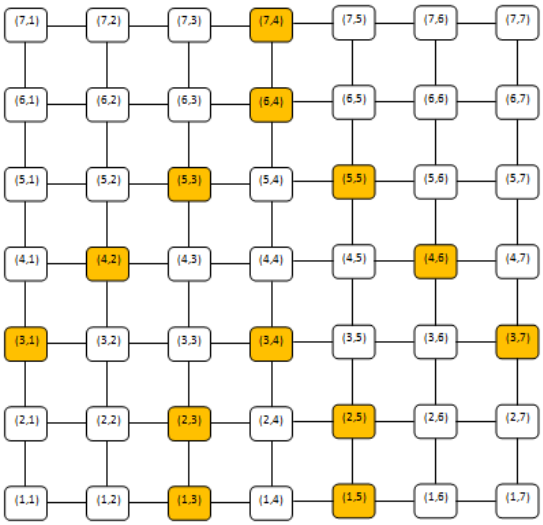
Odd-Even Routing Algorithm

(a)



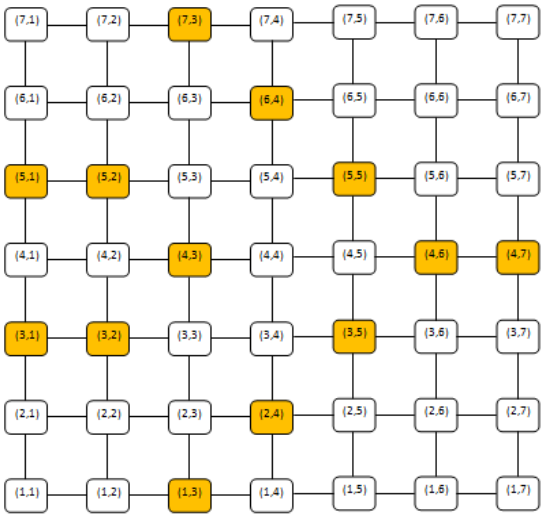
XY Routing Algorithm

(b)



North-Last Routing Algorithm

(c)



West-First Routing Algorithm

(d)

Figure 8-4.7x7 Junction-Based networks using various routing algorithms with hop count limit of 5

8.3 Comparison of JBR and Source Routing for Various Routing Algorithms in Local Traffic

8.3.1 Latency Graphs

In this section, the graphs which we plotted for average packet latency against different values of packet injection rate in local traffic using various routing algorithms for JBR and source routing are presented.

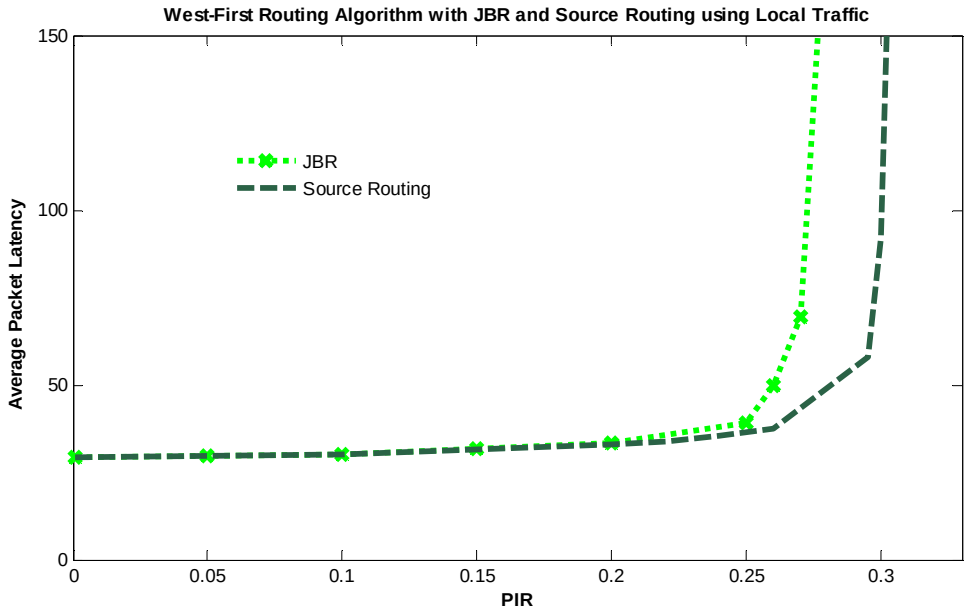


Figure 8-5.Average packet latency plotted against PIR in local traffic for JBR and source routing using West-First routing algorithm in a 7x7 mesh NoC

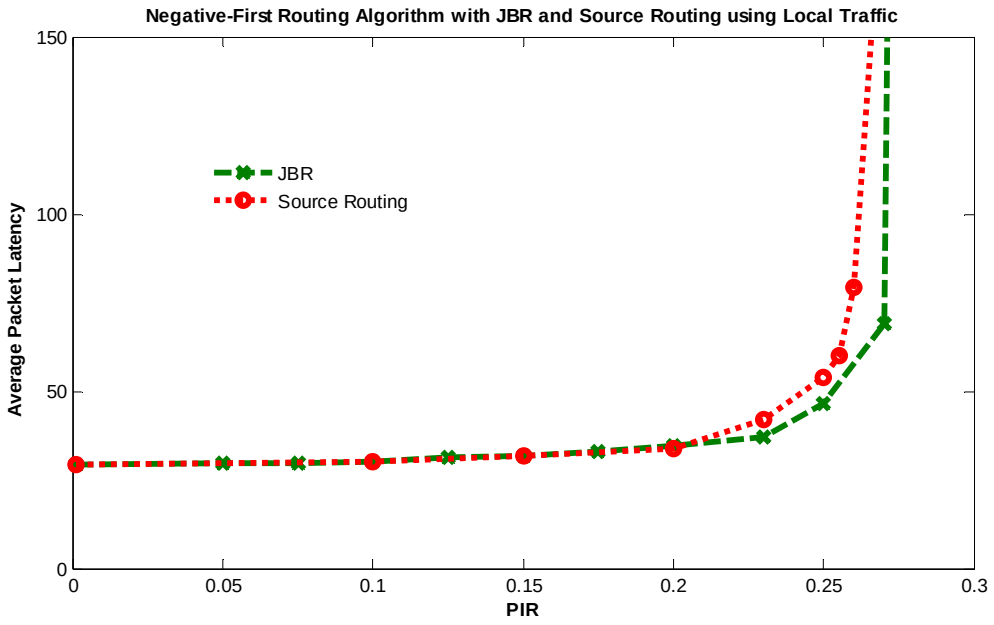


Figure 8-6.Average packet latency plotted against PIR in local traffic for JBR and source routing using Negative-First routing algorithm in a 7x7 mesh NoC

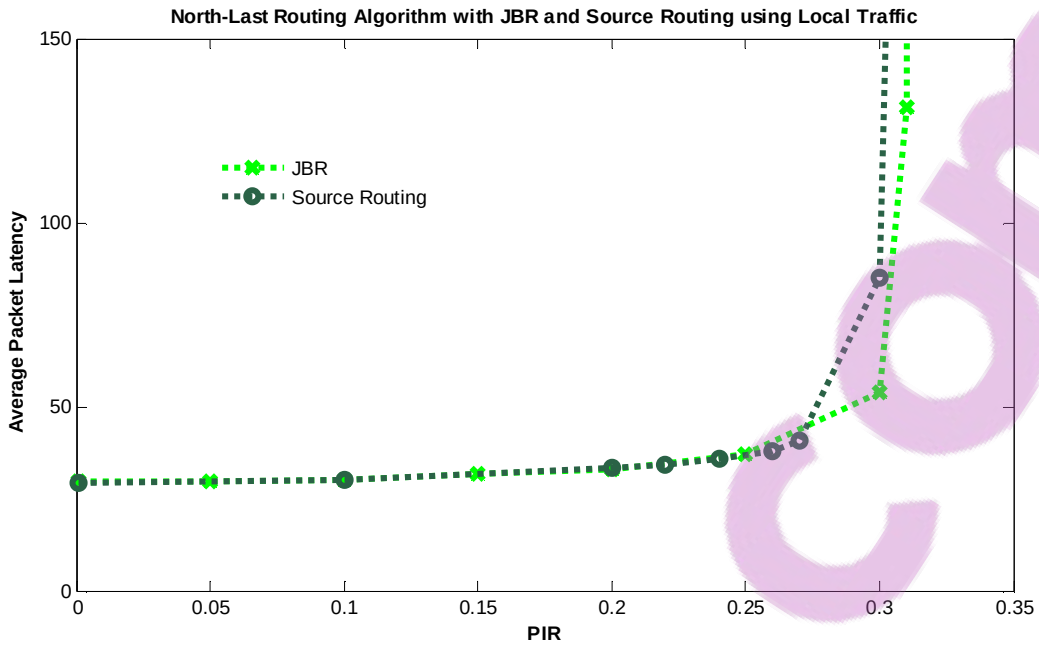


Figure 8-7.Average packet latency plotted against PIR in local traffic for JBR and source routing using North-Last routing algorithm in a 7x7 mesh NoC

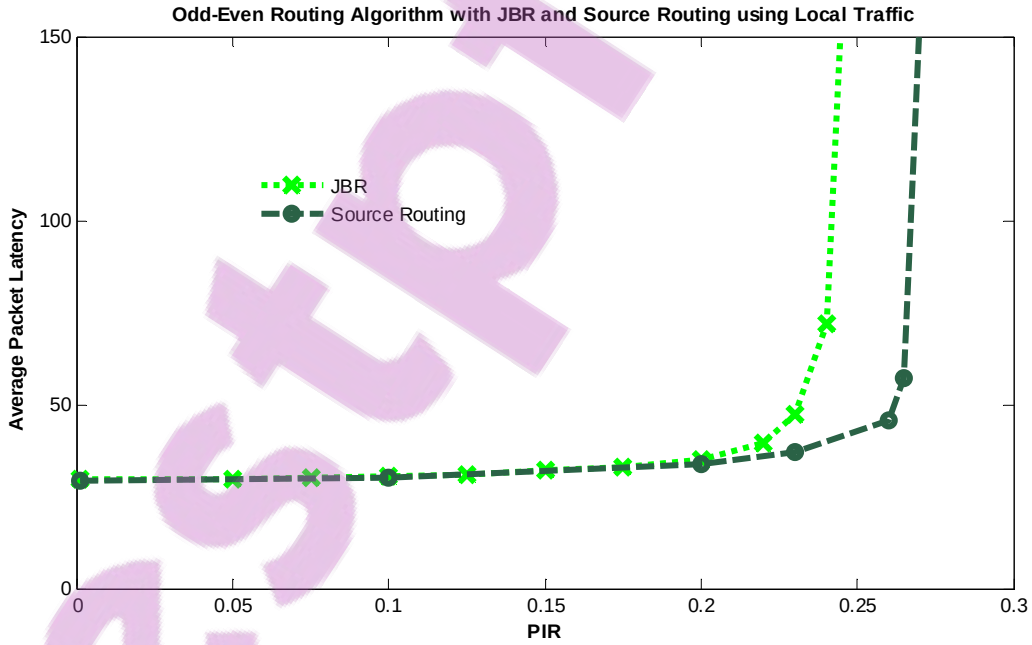


Figure 8-8.Average packet latency plotted against PIR in local traffic for JBR and source routing using Odd-Even routing algorithm in a 7X7 mesh NoC

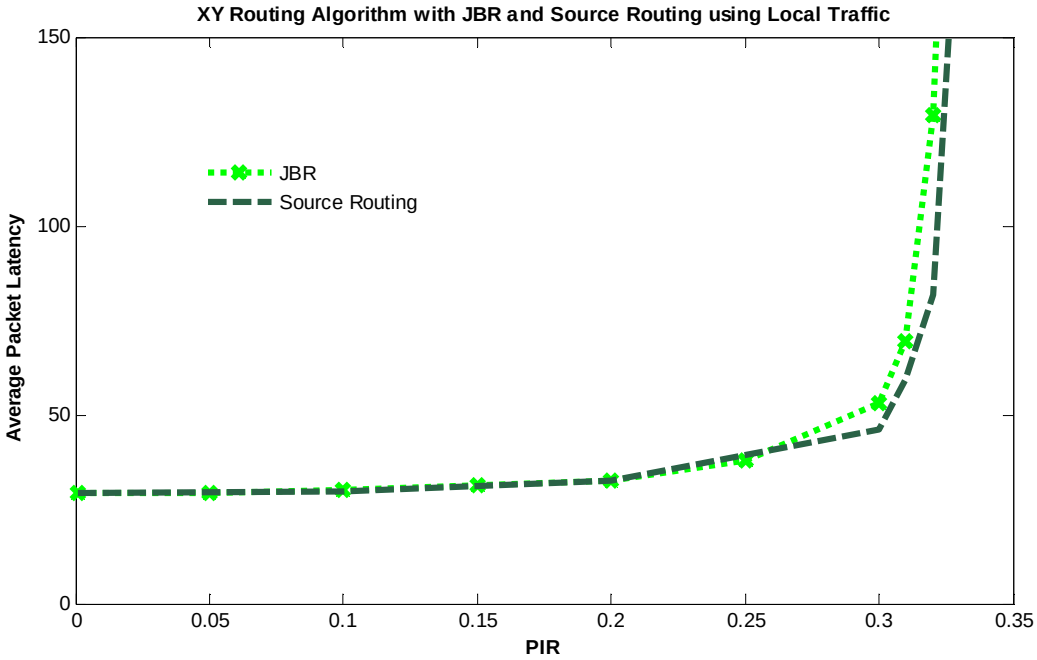


Figure 8-9.Average packet latency plotted against PIR in local traffic for JBR and source routing using XY routing algorithm in a 7x7 mesh NoC

8.3.2 Throughput Graphs

In this section, the graphs which we plotted for throughput against different values of packet injection rate in local traffic using various routing algorithms for JBR and source routing, are presented.

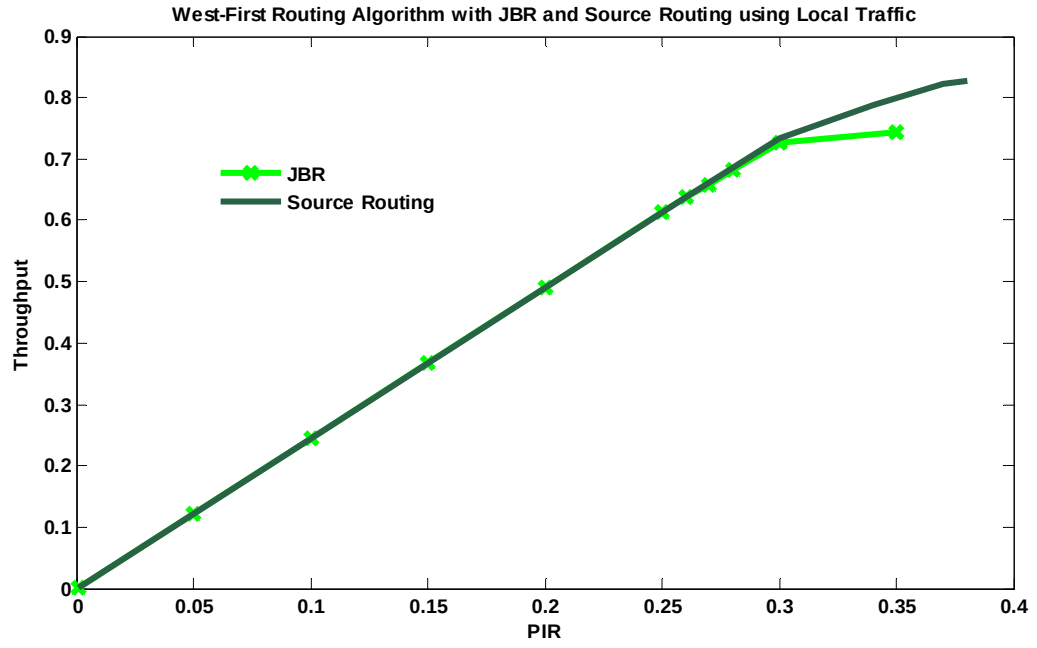


Figure 8-10.Throughput plotted against PIR in local traffic for JBR and source routing using West-First routing algorithm in a 7x7 mesh NoC

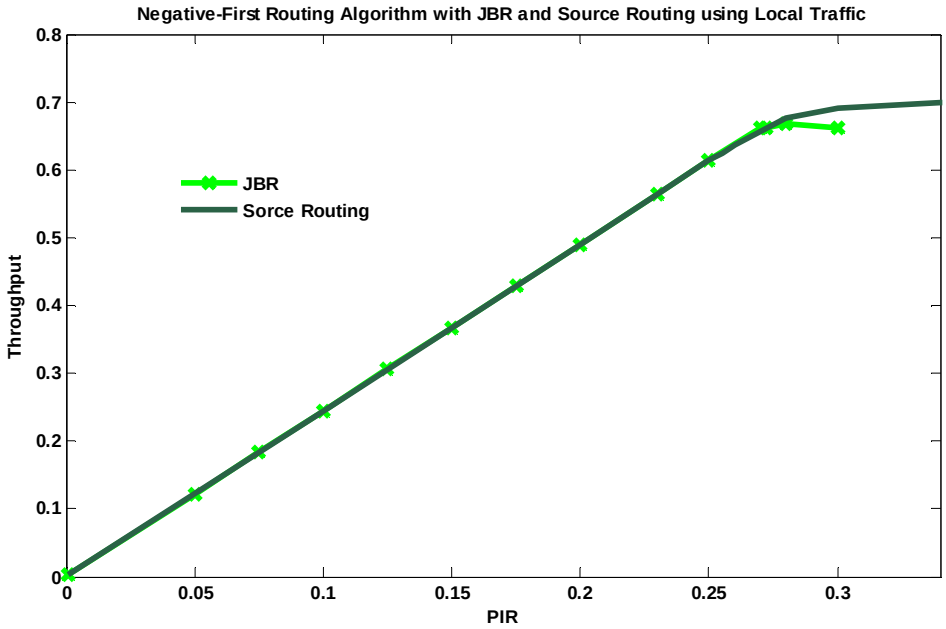


Figure 8-11.Throughput plotted against PIR in local traffic for JBR and source routing using Negative-First routing algorithm in a 7x7 mesh NoC

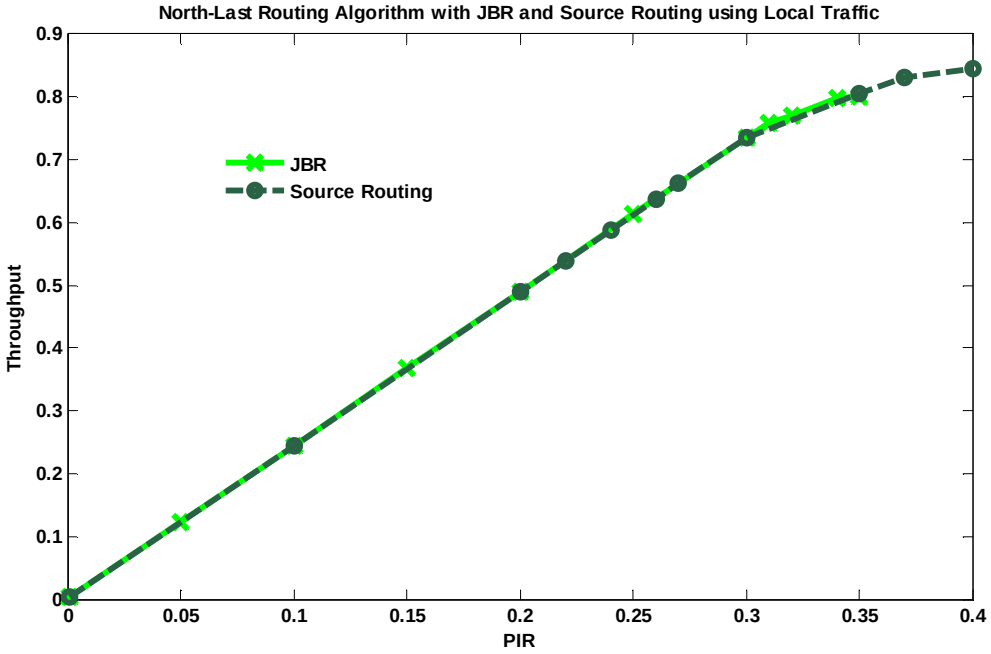


Figure 8-12.Throughput plotted against PIR in local traffic for JBR and source routing using North-Last routing algorithm in a 7x7 mesh NoC

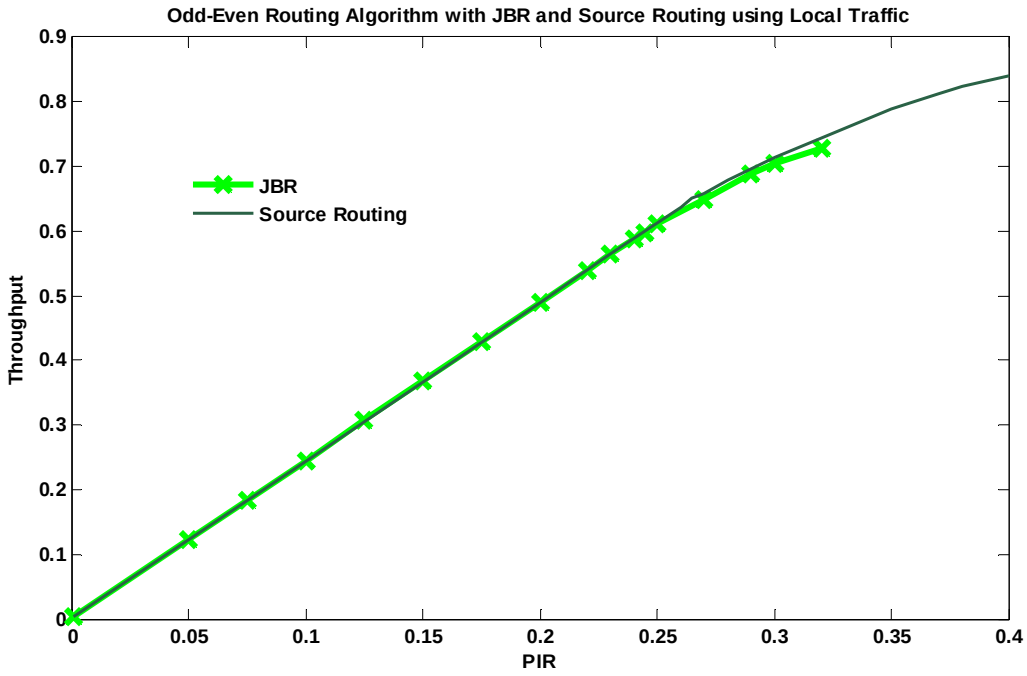


Figure 8-13.Throughput plotted against PIR in local traffic for JBR and source routing using Odd-Even routing algorithm in a 7x7 mesh NoC

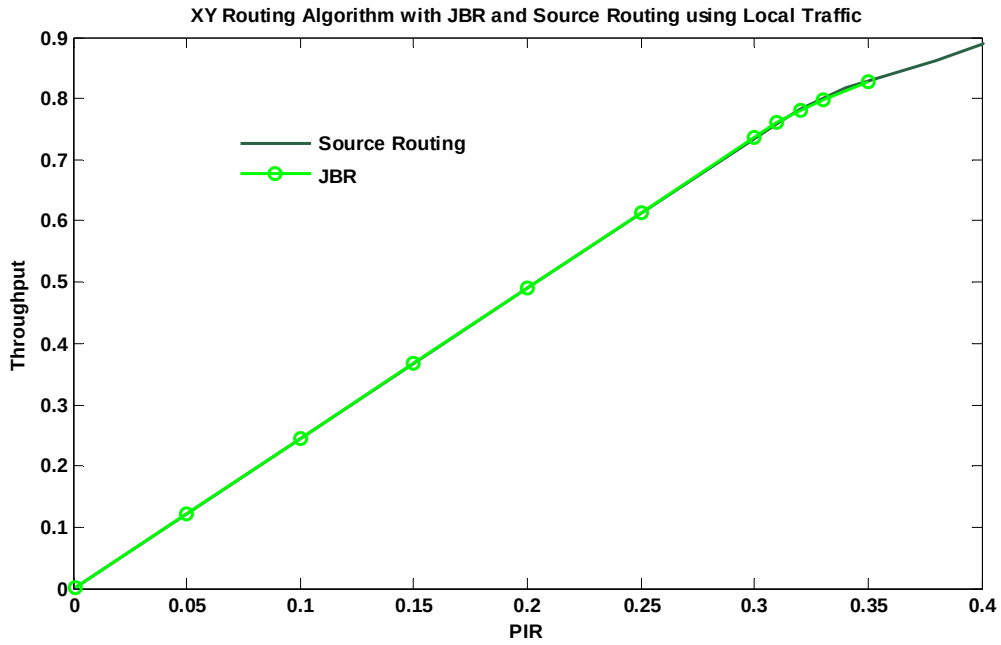


Figure 8-14.Throughput plotted against PIR in local traffic for JBR and source routing using XY routing algorithm in a 7X7 mesh NoC