How to Design Source Routing for Mesh Topology Network on Chip?

Saad Mubeen
Mälardalen University, Sweden

Shashi Kumar
Jönköping University, Sweden

1 Introduction

With the recent advancement in multi-core technology, more and more computing and memory cores are being fabricated on a single chip. The current trend of designing and using such a system on chip offers a large design space and many research challenges. One such challenge is the design of an efficient on-chip communication infrastructure for these Systems on Chip (SoCs). Network on Chip (NoC) paradigm has emerged as a competitive candidate for implementing communication in SoCs. Topology and routing algorithm are two important features which distinguish various NoC platforms. Communication performance of a NoC depends heavily on the routing used. Routing methods can be classified into two types, namely, source routing and distributed routing. In this chapter, we discuss an efficient methodology to design source routing for NoCs, especially for regular topology platforms like mesh. By following this development methodology, it can be shown that source routing has a potential of achieving higher throughput and latency performance than distributed routing.

1.1 System on Chip

A rapid progress in Very Large Scale Integration (VLSI) in the past recent years has resulted in the fabrication of millions of transistors on a single silicon chip. With the current CMOS technology it is possible to implement a design with approximately one billion transistors on a single chip. This advancement in the micro-electronics leads to the integration of various components of a computing system or any other electronic system on a single Integrated Circuit (IC) to implement a complete system on a chip. Thus a paradigm called System on Chip (SoC) came into existence that refers to the system made up of interconnected cores or Intellectual Property (IP) block on a single chip.

1.1.1 Chip Capacity

Development of a complete system on a single chip became possible due to advancement in chip capacity which is the number of transistors that can be fabricated on the chip. Increase in chip capacity has been following the trend given by Moore's law, i.e., the chip capacity doubles every 18 months approximately. This law has been holding for over 40 years.

1.1.2 Core Based Design

In order to shorten the time to market, time to test and exploit reuse, mostly SoCs are designed with cores or IPs (Gupta & Zorian, 1997). A core can be a general purpose processor, a DSP, a memory block, an application specific hardware component, an I/O controller, a Graphic Controller, a mixed signal module, a Radio Frequency (RF) unit, etc. A designer can build a SoC by developing own cores or reusing Commercial off-the shelf (COTS) cores, available from different IP vendors, and finally interconnect them on a single chip (Holsmark & Högberg, 2002). An example of core based system design is the 80-core processor built recently by Intel (Sriram Vangal, et al., 2008).

1.1.3 Limitation of Buses and Interconnections

Nowadays, direct interconnections and mostly shared busses are used for on-chip communication. The problem with direct interconnections is that they are not scalable and become inefficient with an increase in the number of cores. Shared buses do not scale beyond 8 to 10 cores. Contention for the bus and arbitration also slows down the data movement. They are only good for low communications. In order to build a system on chip with large number of communicating cores, a new design solution, other than direct interconnections and shared busses, providing communication among the cores is needed.

1.2 Network on Chip

Network on Chip (NoC) is being considered as the most suitable candidate for implementing interconnections in core based SoC design (Jantsch & Tenhunen, 2003). In NoC paradigm, cores are connected to each other through a network of routers and they communicate among themselves through packet-switched communication. The protocols used in NoC are generally simplified versions of general communication protocols used in data networks. This makes it possible to use accepted and mature concepts of communication networks such as routing algorithms, switching techniques, flow and congestion control in NoC. It allows significant reuse of resources and provides highly scalable and flexible communication infrastructure for SoC design (Kumar et al., 2002). The cores in a NoC operate in Globally Asynchronous, Locally Synchronous (GALS) mode. A SoC with a NoC communication infrastructure is shown in Figure 1.

1.3 NoC Design Issues

Performance of a NoC depends on many factors. Three main factors are discussed below.

1.3.1 Topology

Topology is a very important feature in the design of NoC because design of a router depends upon it. Different topologies are proposed in the literature for the design of NoC. Commonly used topologies are mesh, ring, torus, binary tree, bus and spidergon. Some researchers have also proposed topologies suitable for an application or an application area. We will focus on mesh topology in this chapter.

1.3.2 Core Selection

Core selection is an important step in the design of a NoC. Usually NoC is designed with a fixed size of tile slot. Therefore, a selected or designed core must fit in this slot. Some researchers have proposed the concept of “Region” to accommodate cores of different sizes (Holsmark & Kumar, 2005). Moreover, cores in NoC may be homogenous or heterogeneous. As compared to heterogeneous, homogeneous cores are all exactly the same having same size, instructions sets, equivalent clocks and perform same functions. Although NoC with

heterogeneous cores becomes complex, it may be relatively more efficient in terms of performance, power and thermal efficiency (Mubeen, 2009).

1.3.3 Routing Algorithms

Communication performance of a NoC depends heavily on the routing algorithm used. Routing methods have been classified in literature in several ways. One way to classify them is Source Routing (SR) and Distributed Routing (DR). In SR, a source core pre-computes the information about the whole path from the source to the destination; selects this information for a desired communication and provides it in the packet header. In DR, the header contains destination address only and the path is computed dynamically by participation of routers on the path (Dally & Towles, 2004a; Flich et al., 2000; Aydogan et al., 1996). Design of a router also depends upon the type of routing. For example, router design for SR will be simpler as compared to the router designed to handle DR algorithm.

A very large number of routing algorithms have been proposed in literature (Palesi et al., 2009; Chiu, 2000; Patooghy & Sarbazi-Azad, 2006). All the proposals fall under DR type. SR has not been considered so far for NoCs, due to its apparent large overhead to store path information in the packet header. Since the paths in SR are pre-computed offline, therefore it can provide no or limited path adaptivity in the case of faults and traffic congestion. In spite of these disadvantages, we feel, SR has many advantages over DR.

2 Source Routing : an Overview

In this section, we discuss SR in detail, illustrate its use in NoC with an example and present its merits and demerits. Moreover, we also investigate SR in NoC context and perform its analytical comparison with DR. Finally we present the necessary steps required to implement SR in NoCs.

2.1 Introduction to Source Routing

In SR, the information about the whole path from the source to the destination is pre-computed and provided in packet header as opposed to DR, where packet header contains destination address only and the path is computed dynamically by the participation of routers on the path (Dally & Towles, 2004a; Jantsch & Tenhunen, 2003; Flich et al., 2000). With SR, all routing decisions are made inside the source core before injecting any packet in the network. For this purpose, each source contains lists or tables that contain complete route information to reach all other resources in the network. Instead of storing tables in source, it is also possible to add extra logic or software in the source resources that implements any adaptive routing algorithm and dynamically computes paths for SR. In order to route a packet through the network using SR, a sender resource consults its routing table to get a complete path to the required destination. This path is then written in the dedicated field in the packet header. The packet is transferred to the network through network interface. The packet must follow the path while traversing through the network towards its destination. Each router that receives this packet reads the path field in the packet header and forwards it to the destined output port. Unlike a router used in DR, this router does not require any extra computation for making routing decisions because the packets already contain pre-computed decisions.

2.2 Illustrative Example of Source Routing for Mesh Topology NoC

In order to demonstrate working of SR, consider an example of a 4X4 mesh topology network on chip as shown in Figure 1. Assume that a DSP processor connected to router (1, 1) wants to send a packet to a memory resource

connected to the router (2, 3) as indicated by black arrow in the figure. Also consider that XY routing algorithm is used for this communication. Accordingly, the packet generated by DSP processor will traverse through routers (1, 1), (1, 2), (1, 3) and (2, 3) before reaching the destination memory resource. Thus the packet header will contain the address of all the routers traversed as shown in Figure 1. Similarly, the figure also depicts the packet format if DR was used. The packet header contains only destination address instead of complete path.

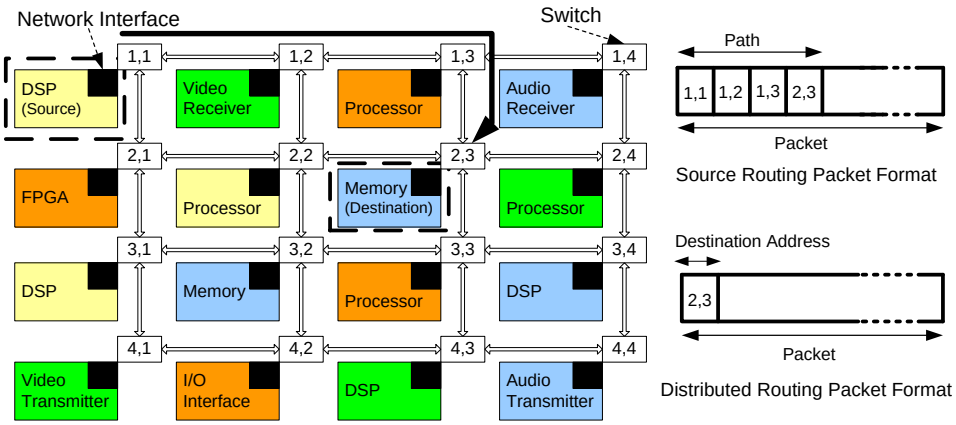


Figure 1: Illustrative example of source routing for a 4X4 mesh topology NoC

2.3 Advantages of Source Routing

Source routing is not perhaps suitable for dynamic networks where network size and topology are changing. But in a NoC with fixed size and regular topology like mesh, the path information can be efficiently encoded with small number of bits. It can be easily shown that two bits are sufficient to encode every hop in the path. We feel that the following advantages of SR (Dally & Towles, 2004a; Ni & McKinley, 1993; Aydogan et al., 1996) more than compensate its disadvantages.

2.3.1 Speed

The foremost advantage is speed. Once a path is selected from the table and included in the packet header inside source, no further time is spent in routing. As each packet arrives at a router, it can immediately select its output port from pre-computed path information in the packet header without any computation or memory reference. Thus, a router used for SR is faster than that used for DR.

2.3.2 Simpler and Smaller Router Design

Since the packet entering a router contains the pre-computed decision about the output port, there is no need for any routing logic or tables in the router and hence, the router design is significantly simplified and its implementation will also be less costly.

2.3.3 Topology Independence

Source routing is topology independent. It can route packets in any connected topology provided it does not change dynamically. It should be noted that this advantage of SR is limited by the number of router ports, the size of the source table and the maximum length of a route.

2.3.4 Mixed, Minimal and Non-minimal Routing

In case of SR, non minimal routing advantage can be applied. Non minimal routing offers a number of advantages such as link load distribution, congestion control and fault tolerance. It also provides possibility of mixing minimal and non-minimal paths.

2.3.5 Scalability

Since only a constant number of bits of the header are used in every router, its design is not only simple but also independent of the network size. Routers that use SR can be used in arbitrary-sized networks because all the limitations on network scalability including network size, source table size, and route length are determined by the source. We feel this to be a major advantage over DR where destination address field will depend on network size and topology.

2.3.6 Ease of Changing Paths

In SR, there is a possibility of changing paths very easily as path table is stored in memory of the source. The paths can even be changed dynamically.

2.3.7 Property of Load Distribution

Since NoCs used in embedded systems are expected to be application specific, we can have a good profile of the communication traffic in the network (Palesi et al., 2009). This allows us to analyze the traffic and compute offline, efficient paths giving the desired performance characteristics, like uniform link load distribution. On the other hand in DR, some extra logic is required in the routers to keep the status of the neighboring router ports for the purpose of load balancing and it is very hard to distribute link load uniformly when DR is used.

2.3.8 No Problem of Live Lock

Live lock is the situation in which a packet keeps traveling in the network and never reaches its destination. There is no problem of live lock in SR. A packet takes finite number of hops before reaching its destination because of the fixed length source route. Live lock may be a problem with some DR algorithms but it can be avoided with careful construction of routing tables.

2.3.9 Guaranteed Throughput

Source routing is better when guaranteed throughput is required especially in the case of real time traffic. This can be achieved by assigning “special paths” to such communication.

2.3.10 In-Order Delivery of Packets

The single path for each pair in the network avoids out of order packet delivery problem that is exhibited by adaptive routing algorithms.

2.3.11 Good for Troubleshooting a Network

Source Routing can also be used to troubleshoot a network. If any link in the network is broken or any router is down, then test packets are sent to all destinations using SR. Upon receiving test packets, destinations reply by sending response packets to the sender. Faulty link or router can be identified easily by performing the analysis of the response packets received.

2.3.12 Efficient Core Mapping Based on Source Routing

In (Tornero et al., 2010), a core mapping technique based on source routing is introduced which helps to achieve a mapping with a constraint over the path length. The authors demonstrate the feasibility of reducing the path length to just 50% of the diameter thus making core mapping based on source routing more efficient. This technique highly depends upon the computation of efficient paths for source routing.

2.4 Disadvantages of Source Routing

2.4.1 Routing Overhead

In SR, a packet must carry the routing information in the packet header thus increasing the size of packet. Packet header in SR is larger compared to that of DR.

2.4.2 Static and Non-Adaptive Nature of Source Routing

Source routing is static in nature. This means that the path cannot be changed after the packet has left the source unless some sophisticated techniques are used. It does not take into account the current traffic pattern in the network. Although some sort of adaptivity may be introduced by keeping more than one path to every destination in the source tables, still SR cannot achieve adaptivity. Moreover, SR is unable to work in the presence of faults in the network unless some fault tolerance is introduced.

2.4.3 Limitation of the Size of Source Table

In SR, there is a limitation of the size of source table. Storing large tables in sources may become a cost, size and performance overhead for resources, especially for resources which are not of processor type.

2.4.4 Scalability

In SR, there is a limitation of the maximum length of the route, i.e., the path may not fit in the flit unless some special technique is used. The technique should allow variable size path information. This will increase path overhead and complexity of decoding logic.

2.5 Analytical Analysis of Source and Distributed Routing for NoC

In this subsection, we perform analytical analysis of source and distributed routing. This analysis is focussed on the routing overhead, bandwidth utilization and router delay.

2.5.1 Overhead

As complete route information along with the payload should be stored in a packet in case of SR, large under-utilization is expected. But analytical comparison between SR and DR eliminates this fear of underutilization. The left part in Figure 2 shows a graph plotted for the number of bits required for routing against different size of square mesh NoC for both SR and DR. It is clear that with SR, the number of routing bits increases in proportion to the diameter of the network (i.e., square-root of network size). On the other hand, routing bits required for DR increase logarithmically with the network size. The gap between the two graphs keeps on increasing with an increase in the size of NoC. This apparently makes SR look unusable in practice. For both types of routing, the number of bits required to route data in a NXN mesh NoC is given by:

$$\text{Number of routing bits in DR} = 2\lceil \log_2(N) \rceil$$

$$\text{Number of routing bits in SR} = 2(2N-1)$$

But, if the overhead is measured in terms of extra flits to be communicated, the difference is very small or zero. We analyze this in the next subsection

2.5.2 Bandwidth Utilization

We get a different view of the comparison between SR and DR in NoC when bandwidth utilization is considered. Bandwidth utilization is defined as the ratio of the payload in bytes to be transmitted and the actual number of bytes to be sent carrying this payload as shown in the following equation.

$$\text{Bandwidth Utilization} = \frac{\text{Payload in Bytes to be transmitted}}{\text{Actual number of bytes to be sent carrying this payload}}$$

When wormhole switching technique (Dally & Towles, 2004b) is used in NoC, the data is transferred in the form of flow control digits called flits. We consider fixed size flit equal to four bytes. (Mubeen, 2009) can be referred for further details of a flit. In case of SR, a head flit carries the complete route information and we assume that it does not carry any payload. In case of DR we consider that a head flit can carry maximum of two bytes of payload. Let "P" be the payload in bytes to be transmitted. Based on these considerations, actual number of bytes to be sent carrying payload "P" in case of SR and DR is given by:

$$\text{Actual number of bytes required in SR} = 4 + 4\lceil (P/4) \rceil$$

$$\text{Actual number of bytes required in DR} = \begin{cases} 4, & \text{for P = 1, 2} \\ 4 + 4\lceil (P-2)/4 \rceil, & \text{for P} > 2 \end{cases}$$

In the right graph of Figure 2, utilization is plotted against the payload in bytes to be routed for both SR and DR for a 7x7 mesh topology NoC. It can be seen that with less number of bytes in payload, both SR and DR show very less utilization and relatively bigger gap between the two. By increasing the number of bytes in payload, utilization of both routings increases while the gap between them becomes smaller. There exists a correlated trend of increasing utilization with increasing payload in both types of routing. For larger payloads, utilization of both types of routing becomes very close to each other and near to 1, thus making SR as good as DR. This quality of SR was not evident from the analysis of left graph of Figure 2.

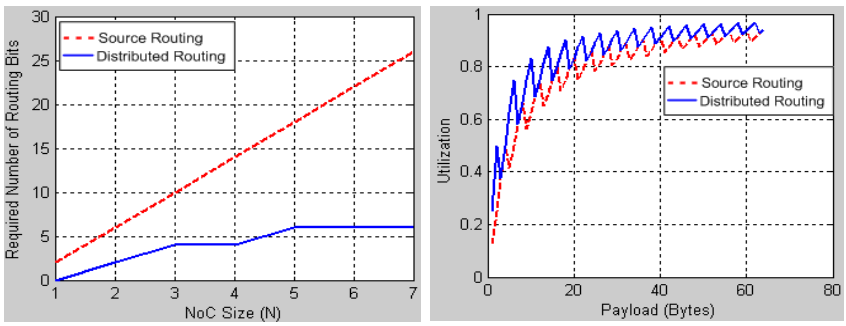


Figure 2: Routing bits requirement for NXN sized NoC for SR and DR; Bandwidth utilization for SR and DR in NoC

2.5.3 Router Delay

Router logic delay "T_{RL}" is a very important factor to be considered when source and distributed routings are compared analytically. T_{RL} is the time required to transfer a flit from the input buffer to the output buffer of a router. It depends upon the time required to decode flit type and routing information, selection of output port and

switching activity inside the router. As opposed to SR, routing tables are consulted in each router in DR, thus, creating an extra delay of at least one router clock. In case of SR, complete path information is already present in the header flit; therefore routing logic is simpler, faster and smaller compared to DR.

2.6 Steps for Designing NoC with Source Routing

Design of complete SR scheme involves a number of steps including algorithm selection for SR, computation of path for each communication, uniform distribution of link load, path improvement, encoding of the computed paths, simulation analysis of SR, and finally design of a router as shown in Figure 3.

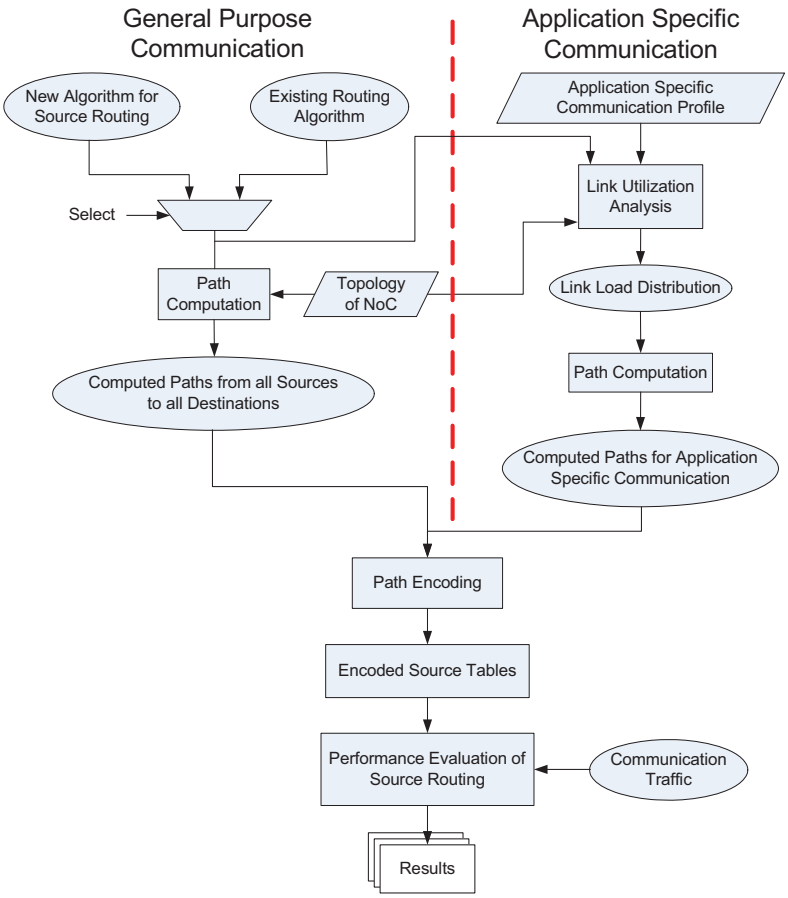


Figure 3: Design steps required for source based routing in NoC

2.6.1 Algorithm Selection and Path Computation for Source Routing

First step in designing SR is to make decision that whether an exiting routing algorithm will be used or a new algorithm will be developed for path computation. Once the decision is made, then paths should be computed for each source to every destination depending upon the selected communication, i.e., general purpose or application specific. In application specific case where a complete communication profile is available, analysis for link load distribution is performed before computing final paths by exploiting the adaptivity of the routing algorithm used. Path improvement algorithms can also be applied to get improved paths. Final computed paths are used in SR.

2.6.2 Path Encoding

Once path from source to destination is computed, it should be encoded in the packet header. Path encoding should be done in such a way that the overhead of routing information is minimized. Moreover, path encoding should be such that it is easy to decode in the routers on the way to the destination. In order to minimize the route information overhead in the head flit, we have used 2-bit clockwise router port address encoding scheme (Mubeen & Kumar, 2010).

2.6.3 Performance Evaluation of Source Routing

Before designing a router for SR, performance of SR should be analyzed for various routing algorithms using different traffic patterns. Therefore a simulator is required that is capable of simulating SR. We developed a simulator (Mubeen, 2009) for mesh topology NoC based on SR and performed a number of simulations to investigate SR using various traffic patterns.

2.6.4 Router Design

Router design for SR will be much simpler than the router design to handle a DR algorithm. It does not need to compute a routing function to select the output port for an incoming packet. The pre-decided information is available in the header flit. It decodes the 2-bit clockwise port address and forwards the head flit to the desired port. Other flits of the same packet follow the head flit if wormhole switching is used. But this router still needs to implement the other functions like packet buffering, credit-based flow control and arbitration to resolve port conflicts when two or more packets contend for the same output port. These terms are discussed in detail in (Mubeen, 2009). The simplicity of the source router makes it relatively smaller and faster. Several issues in the design of NoC router for SR are discussed in (Mubeen, 2009; Mubeen & Kumar, 2010).

3 Path Computation and Encoding for Mesh Topology Network on Chip

This section starts with the description of the requirements and methods to compute paths for SR. Then follows the description of MatPC Tool which is developed in Matlab to compute paths for SR and analyze link utilization by the computed paths. Moreover, different routing algorithms are evaluated on this tool for different types of communications and traffic patterns. Similarly, a method is described for selecting best routing algorithm for a specific application to compute paths for SR. Discussion continues with the path improvement algorithm that we implemented to obtain improved paths. Finally, we discuss the path encoding scheme used to encode paths.

3.1 What is Path Computation?

Path computation refers to finding a complete path or a route from a source resource to a destination resource. In general case, a path should be computed for each pair of resources in the network. For example in a 4x4 mesh topology NoC, paths should be computed from first resource to the rest of 15 resources as destinations; from second resource to the rest of 15 resources and so on until paths are computed for the last, i.e., 16th resource to the rest of resources as destinations. In an application-specific case, paths must be computed for only those pairs of resources which communicate. The computed paths must avoid any possibility of deadlock. Apart from that, computed paths should provide small delay and avoid link congestion. For good performance, paths should also be computed in such a way that the load is uniformly distributed in the network. The computed paths can be either stored in each respective resource in the form of lists or path is computed dynamically using an algorithm in each resource before sending a packet. We assume that paths are computed off-line and stored as tables in each

resource.

3.2 Options to Compute Path for Source Routing

In general, there are two options available for computing paths for SR. According to first option, paths can be computed by using an existing DR algorithm. The second option is to devise a new method and compute paths accordingly. The new method for path computation can also be developed by mixing existing DR algorithms through some clever means. We select the first option. Consequently, the paths for SR are computed by using variety of existing DR algorithms for mesh topology NoC for both general and application-specific communications. Later, we also propose methods to improve the computed paths to avoid congestion and provide better load distribution.

3.2.1 Path Computation using Existing Routing Algorithms

A large number of deadlock free routing algorithms exist for mesh topology NoC which can be used for path computation of SR. Depending upon the requirements, one can choose from deterministic, partially adaptive and adaptive available routing algorithms. We select only deterministic and partially adaptive existing routing algorithms for the computation of paths for SR. XY is a deterministic deadlock free routing algorithm for mesh topology. Similarly, turn model based partially adaptive wormhole routing algorithms can also be used for path computation in SR. Some of these algorithms, i.e., West-First, North-Last, and Negative-First are discussed in (Mubeen, 2009). These algorithms restrict at least half of the source-destination communications to one minimal path, while rest of the pairs can communicate with full adaptivity. Odd-Even routing algorithms is also a partially adaptive deadlock free routing algorithm based on restrictions of taking turns at some locations. It provides more even degree of adaptiveness comparatively (Chiu, 2000).

3.2.2 MatPC Tool: Matlab Based Path Computation and Link Utilization Analysis Tool for Source Routing

A tool is developed in Matlab which can compute paths for SR using existing DR algorithms (Mubeen, 2009). It is capable of computing paths for both types of communications, i.e., general case where all resources are communicating with all the other resources in the network and application-specific case where each resource communicates with few other resources in the network depending upon the application. Currently, this tool computes path for only minimal routing and does not allow 180 degree turns. It requires a lot of research and hard work for making this tool capable of computing paths for SR by also using non-minimal routing and thus, non-minimal routing is not included in the objectives of this work. The inputs, outputs and user interface of this tool are discussed in (Mubeen, 2009).

3.3 Selection of Routing Algorithms, Traffic Pattern and Performance Parameter for Evaluation

A large number of experiments were performed using different types of traffics with different routing algorithms to compute paths for SR for application-specific communication. The detailed results of these experiments are analyzed and interpreted in (Mubeen, 2009).

3.3.1 Types of Routing Algorithms used for Evaluation

One can use any existing DR algorithm for computing paths for SR. Both deterministic and partially adaptive routing algorithms have been evaluated for this purpose. The algorithms are selected on the basis of certain routing properties such as deadlock freedom and adaptivity. Adaptivity of a routing algorithm can be used for uniform distribution of the link load in the network. Accordingly, five routing algorithms namely XY, Odd-Even,

West First, Negative First and North Last are chosen for evaluation purpose.

Computed routing paths will be stored in a table in the core or in the Network Interface. Since, only one path will be stored for each communicating pair, we exploit adaptivity by using equal probability to the selection of available routes to the destination. We use only minimal routing, so adaptivity of routing algorithms can provide a maximum choice of two different paths in routers on the path. Thus we set 50% probability for the selection of each of the available output port of a router.

3.3.2 Types of Traffics used for Evaluation

After selecting five routing algorithms to compute paths for SR for the purpose of evaluation, a question arises that which routing algorithm shows best performance? The answer to the question comes from the literature and which tells us that different routing algorithms perform differently for different traffics (Chiu, 2000; Patooghy & Sarbazi-Azad, 2006). Therefore, we have selected five different types of traffic for the purpose of evaluation. These traffics include random, east dominated, south dominated, west dominated and hot spot.

Performance of all the selected routing algorithms will be analyzed using each of the above mentioned traffics. Local-biased traffic is used in the application-specific case when we consider east, south and west dominated traffics. This means that a typical resource has a higher chance of becoming a destination compared to other resources if it is closer to the source resource. All the traffics, which we discussed above, will be generated by the Matlab based path computation tool for SR.

3.3.3 Link Load: Performance Parameter used for Evaluation

Like any other network, the performance of a routing algorithm for a NoC can be evaluated by many performance parameters such as throughput, link load distribution, number of hops, latency, packet drop, fault tolerance, router area and many other parameters. In this section we want to analyze which routing algorithm distributes the link load more uniformly. So the interest lies in the link load as performance parameter and that is defined as the amount of data flowing on each link in each direction provided the links are considered bidirectional. Network delay is heavily dependent on the amount of traffic in the network. Network traffic is generally measured by number of packets injected by resources in the network. Average packet delay grows exponentially with the increase of network load after some point. The reason for increase in delay is that some links in the network experience congestion while other links may be unused. The delay is determined by the maximum load in any network link.

3.4 Algorithm Selection for Specific Application Traffic

A large number of experiments were performed using different size of mesh topology NoC with different types of traffic in order to demonstrate that the selected routing algorithms show different performance for different traffics (Mubeen, 2009). The results indicated that in hot spot traffic, Odd-Even routing algorithm shows best performance among the five routing algorithms considered, with least estimated maximum link bandwidth, mean and variance of the link load distribution. Odd Even routing algorithm gives 16.17%, 21.47%, 25.93%, and 15.34% better performance in terms of standard deviation compared to XY, West First, Negative First and North Last routing algorithms respectively. The performed experiments and their results lead to a solution proposal for the selection of routing algorithms for each type of traffic to compute paths for source routing. Figure 4 shows an algorithm for selecting a routing algorithm for path computation for the type of traffic in the case of application-specific communication.

A communication graph is obtained after identifying all the communications among resources for a specific application. From the communication graph of application, first the type of traffic should be analyzed.

Accordingly, West First, North Last, East first and Odd-Even algorithms should be selected for east, south, and west dominated and hot spot traffic respectively. Similarly, for any other traffic, including random traffic, XY routing algorithm should be chosen. Further, Odd-Even routing algorithm should be preferred if a traffic is hot spot as well as of any other type. After selection of routing algorithm, paths from source to destination for all communicating cores can be computed for source routing.

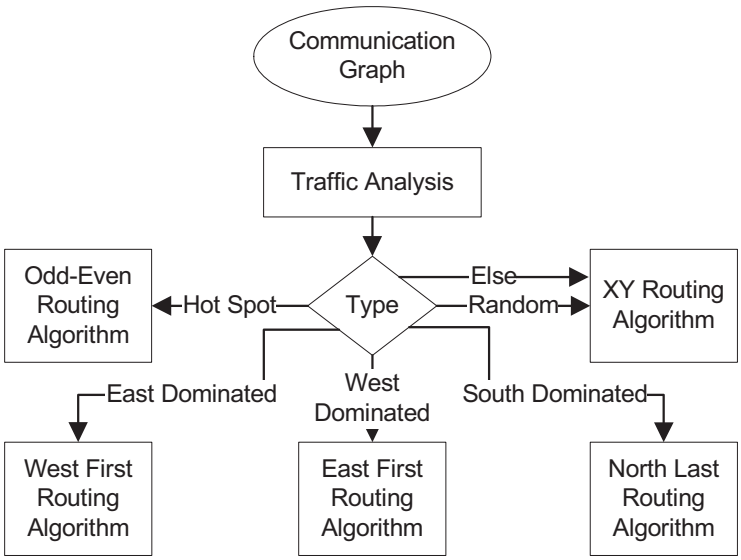


Figure 4: Algorithm for traffic analysis and selection of routing algorithm for source routing using application specific traffic

3.5 Path Improvement

Paths computed for source routing using algorithm shown in Figure 4 may not be the best in terms of link load distribution because real traffic is rarely pure hot spot or completely directed towards east. The idea in our second step is to use the adaptivity of the selected routing algorithm to distribute communications uniformly among paths. Two approaches, namely "constructive" and "iterative" can be used for this purpose.

3.5.1 Constructive Path Improvement Algorithm

Link load in the NoC can be balanced to some extent during the path computation process as shown in Figure 5 (a). Once all the cores which are communicating with each other for a specific application are known, they are ordered before starting the path computation process. Ordering depends upon the cost of communication which reflects the effect of communication volume between pairs and their relative distance in terms of hops.

Communication Cost = (Communication Bandwidth * Distance between source and destination)

Current load on the links in NoC is used for choosing a path for the next communication pair. If any link is found congested, it is avoided for any further communication if possible. This is achieved by the adaptivity of routing algorithm used and hence, alternative routes are used while computing paths for further communications. Although this simple heuristic algorithm may not lead to best link load balancing, we observed that it provides considerable improvement in link load distribution and communication performance.

3.5.2 Iterative Path Improvement Algorithm

After analyzing and selecting routing algorithm shown in Figure 5 (b), initial paths are computed for all communications. In the next step, link load variance is evaluated. If it is acceptably small then the paths which were computed in the first step are used for source routing. On the other hand, if link load variance is not acceptable then the most congested link is identified. One communication using this link is rerouted on an alternative path using adaptivity of the routing algorithm. This may or may not result in better link load distribution. The above process is iterated until link load distribution becomes acceptable or it does not show any further improvement. Hence, the final paths are used for source routing as shown in Figure 5 (b).

3.5.3 Improved Paths for Source Routing

After implementing path improvement algorithm, we performed a large number of experiments using four different traffic patterns, i.e., random, hot spot, east and south dominated. In each experiment, paths were computed using Odd Even, West First, North Last and Negative First routing algorithms with and without path improvement. Percentage path improvement in terms of standard deviation of link load distribution for each algorithm is averaged over 20 experiments and shown in Figure 5. Maximum percentage path improvement is also shown. In case of hot spot traffic, path improvement leads to 10.94% improvement in latency for Odd Even routing algorithm. In the best case, an improvement of up to 28.63% was observed. For hot spot traffic, best improvement was for Odd Even routing algorithm. When south dominated and east dominated traffics are used, North Last and West First routing algorithms give highest improved performance of 11.53% and 9% respectively.

Routing Algorithm	Average (%age) Improvement	Maximum (%age) Improvement
Hot Spot Traffic		
Odd Even	10.9485	28.6309
West First	6.8843	15.9115
Negative First	6.9189	16.638
North Last	6.0683	15.8145
South Dominated Traffic		
Odd Even	11.5331	21.4516
West First	8.7523	14.5442
Negative First	6.0588	12.7521
North Last	11.6872	16.2599
East Dominated Traffic		
Odd Even	7.9888	14.5121
West First	9.0078	12.328
Negative First	2.403	6.5631
North Last	4.6064	10.1174
Random Traffic		
Odd Even	9.7684	24.5034
West First	6.2278	13.3638
Negative First	6.2138	12.8795
North Last	5.4504	12.7377

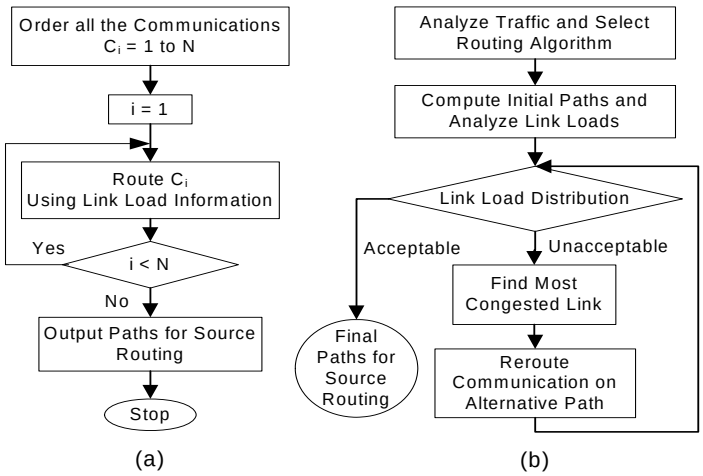


Figure 5: Path improvement results and algorithms: (a) Constructive (b) Iterative

4 Evaluation

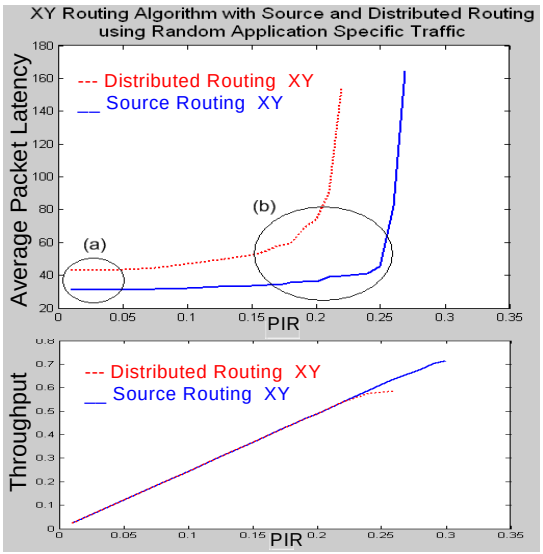
For evaluation of source routing, we enhanced an existing simulator that was earlier developed for distributed routing. It takes input from MatPC tool.

4.1 Comparison with Corresponding Distributed Routing

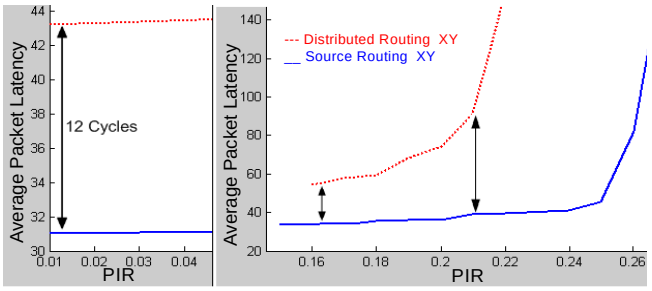
We compare the performance of source and distributed routing using XY and Odd Even routing algorithms. Average Packet Latency (APL) and throughput are considered as performance parameters. APL is plotted against different values of Packet Injection Rate (PIR) in random traffic using XY routing algorithm for source and distributed routing and it is shown by solid and dotted curves respectively in the top graph of Figure 6 (i). It is evident from the graph that latency in the case of source routing is much lower than that of distributed routing.

One remarkable advantage of source routing can be seen from the latency graph near saturation region shown in Figure 6 (ii). In case of distributed routing, when PIR value increases beyond 0.2, the latency increases abruptly and the network starts to saturate. When source routing is used, the latency remains low until PIR reaches 0.25 when the network starts to saturate and latency increases very quickly. The results show that the saturation load can be significantly higher while using source routing.

Throughput is plotted against PIR for both source and distributed routing and it is shown by solid and dotted curves respectively in the bottom graph of Figure 6 (i). At lower values of PIR, throughput increases linearly and is equal for both types of routing. When PIR is increased beyond 0.2, throughput starts to level off in case of distributed routing and the network gets saturated. In case of source routing, throughput keeps on increasing linearly and starts to level off only beyond PIR equal to 0.27. Thus at higher network load, SR provides comparatively higher throughput.



(i)



(ii)

Figure 6: APL vs. PIR and Throughput vs. PIR for a 7x7 NoC for source and distributed routing using XY routing algorithm

4.2 Effect of Path Improvement

In this section we demonstrate the simulation based performance of different partially adaptive routing algorithms used to compute paths for source routing before and after path improvement. Due to lack of space, only hot spot traffic is chosen. Detailed results can be found in (Mubeen, 2009). In Figure 7 (i)(a), (b), (c) and (d), graphs are plotted for APL against PIR in hot spot traffic using West First, Negative First, Odd Even and North Last routing algorithms for source routing respectively. Latency graphs before and after path improvement are shown

by dotted and solid curves respectively. There is no significant improvement in latency with path improvement for all routing algorithms at lower load. At higher values of PIR, path improvement results in lower latency for all routing algorithms except Negative First. This is because Negative First algorithm is not suitable for hot spot traffic and path improvement algorithm worsens its performance. Similarly, the saturation also starts at relatively higher value of PIR when improved paths are used. In case of hot spot traffic, Odd Even routing algorithm gives most improvement.

4.3 Paths for Source Routing Using Best Routing Algorithm

Latency and throughput graphs of the above mentioned routing algorithms in hot spot traffic are shown in Figure 7 (ii). At lower loads performance of these routing algorithms is almost same. At higher loads, Odd Even routing algorithm outperforms the rest by providing lower latency and higher throughput. These results support our proposal of routing algorithm selection for SR. Similarly a large number of simulations were performed for source routing and as expected XY, West First and North Last routing algorithms performed the best in random, east dominated and south dominated traffic respectively [8].

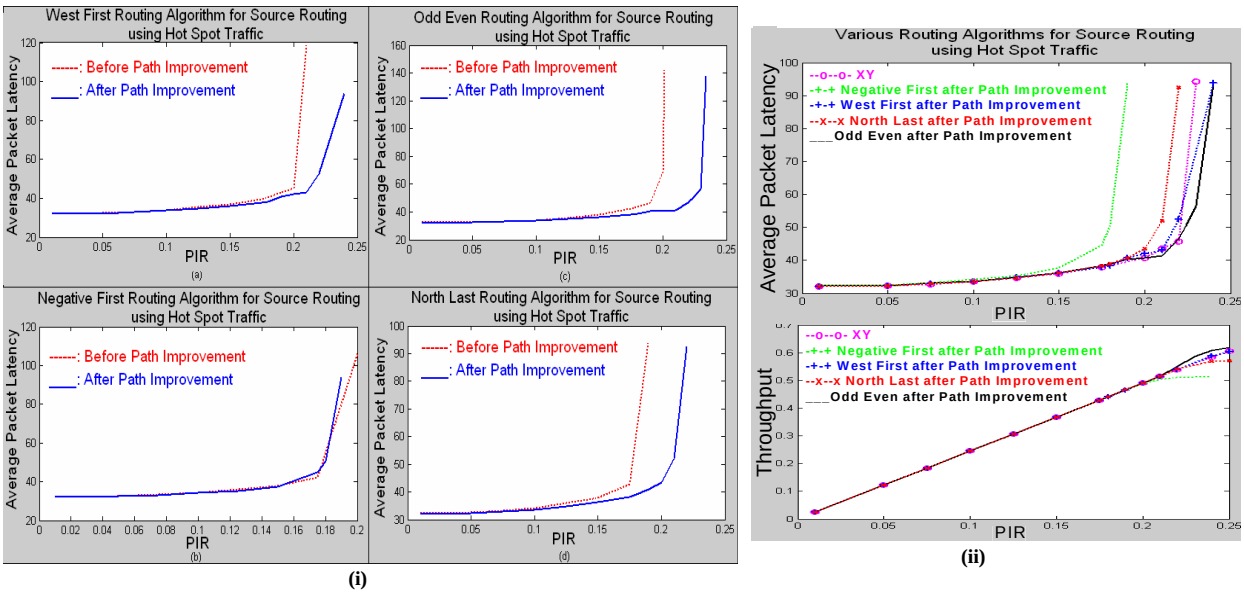


Figure 7: APL and throughput plotted against PIR for source routing using various routing algorithms before and after path improvement in a 7X7 mesh NoC

5 Conclusion

We made a case for the use of source routing in mesh topology NoC. Because of the small and fixed size of practical NoCs, the overhead of source routing is negligible and it is easily compensated by a large number of its advantages, including lower router cost and higher communication speed of the router. We proposed an efficient two step method to compute application-specific paths for source routing. A Matlab based tool called MatPC has been developed for this purpose. We demonstrated the efficacy of using two step approach of path computation. There is a lot of scope for using better heuristics for improving the second step. We developed a simulator for the

evaluation of our source routing approach. Evaluation results show that source routing gives higher latency and throughput performance as compared to corresponding distributed routing.

References

Aydogan, Y., Stunkel, C. B., Aykanat, C., & Abali, B. (1996). Adaptive source routing in multistage interconnection networks. In *Proceedings of the 10th International Parallel Processing Symposium*, IPPS '96 (pp. 258–267).: IEEE Computer Society.

Chiu, G.-M. (2000). The odd-even turn model for adaptive routing. *Parallel and Distributed Systems, IEEE Transactions on*, 11(7), 729 –738.

Dally, W. & Towles, B. (2004a). *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers an Imprint of Elsevier Inc.

Dally, W. & Towles, B. (2004b). *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers an Imprint of Elsevier Inc.

Flich, J., Lopez, P., Malumbres, M., & Duato, J. (2000). Improving the performance of regular networks with source routing. In *Parallel Processing, 2000. Proceedings. 2000 International Conference on* (pp. 353 –361).

Gupta, R. & Zorian, Y. (1997). Introducing core-based system design. *Design Test of Computers, IEEE*, 14(4), 15 –25.

Holsmark, R. & Högberg, M. (2002). Modeling and prototyping of network on chip. Master's thesis, Jönköping University, Sweden.

Holsmark, R. & Kumar, S. (2005). Design issues and performance evaluation of mesh noc with regions. In *NORCHIP Conference, 2005. 23rd* (pp. 40 – 43).

Jantsch, A. & Tenhunen, H. (2003). *Networks on Chip*. Kluwer Academic Publishers.

Kumar, S., Jantsch, A., Soininen, J.-P., Forsell, M., Millberg, M., Oberg, J., Tiensyrja, K., & Hemani, A. (2002). A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on* (pp. 105 –112).

Mubeen, S. (2009). Evaluation of source routing for mesh topology network on chip platforms. Master's thesis, Jönköping University, Sweden.

Mubeen, S. & Kumar, S. (2010). Designing efficient source routing for mesh topology network on chip platforms. In *Proceedings of the 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, DSD '10 (pp. 181–188). Washington, DC, USA: IEEE Computer Society.

Ni, L. & McKinley, P. (1993). A survey of wormhole routing techniques in direct networks. *Computer*, 26(2), 62 –76.

Palesi, M., Holsmark, R., Kumar, S., & Catania, V. (2009). Application specific routing algorithms for networks on chip. *Parallel and Distributed Systems, IEEE Transactions on*, 20(3), 316 –330.

Patooghy, A. & Sarbazi-Azad, H. (2006). Performance comparison of partially adaptive routing algorithms. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, volume 2.

Sriram Vangal, et al. (2008). An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1).

Tornero, R., Kumar, S., Mubeen, S., & Orduña, J. M. (2010). Distance constrained mapping to support noc platforms based on source routing. In *Proceedings of the 2009 international conference on Parallel processing*, Euro-Par'09 (pp. 16–25). Berlin, Heidelberg: Springer-Verlag.