2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools

Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms

Saad Mubeen[1,2] and Shashi Kumar[1]
[1]Department of Electronics and Computer Engineering, Jönköping University, Sweden
[2] Mälardalen Real-Time Research Centre (MRTC), Box 883, 721 23, Västerås, Sweden
saad.mubeen@mdh.se, shashi.kumar@jth.hj.se

***Abstract*—Efficient on-chip communication is very important for exploiting enormous computing power available on a multi-core chip. Network on Chip (NoC) has emerged as a competitive candidate for implementing on-chip communication. Routing algorithms significantly affect the performance of a NoC. Most of the existing NoC architectural proposals advocate distributed routing algorithms for building NoC platforms. Although source routing offers many advantages, researchers avoided it due to its apparent disadvantage of larger header size requirement that results in lower bandwidth utilization. In this paper we make a strong case for the use of source routing for NoCs, especially for platforms with small sizes and regular topologies. We present a methodology to compute application specific efficient paths for communication among cores with a high degree of load balancing. The methodology first selects the most appropriate deadlock free routing algorithm, from a set of routing algorithms, based on the application's traffic patterns. Then the selected (possibly adaptive) routing algorithm is used to compute efficient static paths with the goal of link load balancing. We demonstrate through simulation based evaluation that source routing has a potential of achieving higher performance, for example up to 28% lower latency even at medium load , as compared to distributed routing. A simple scheme is proposed for encoding of router ports to reduce the header overhead. A generic simulator was developed for evaluation and performance comparison between source routing and distributed routing. We also designed a router to support source routing for mesh topology NoC platforms.**

Keywords-Network on Chip (NoC); Distributed Routing; Source Routing; Routing Algorithms; Performance Analysis

I. INTRODUCTION

Majority of current embedded systems are using more than one processor core. This enables the designer to enhance functionality and performance of existing embedded systems. The driving force behind this trend is the capacity of integrated chips which is still growing exponentially according to Moore's law. With the current CMOS technology it is possible to integrate more than a billion transistors on the same chip. This capacity is enough to integrate hundreds of computing and memory cores on a single chip. Designing and using such a system with a large number of cores 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. In a NoC, cores are interconnected to each other through packet switched infrastructure consisting of a network of routers [1][2][3][4][14].

The computation power of a multi-core SoC will depend on the number and type of computing cores and size of on-chip memory. The computational capability of such systems will also be affected by the communication capability of the on-chip communication infrastructure (NoC). Topology and routing algorithm are two important features which distinguish various NoC platforms. Communication performance of a NoC depends heavily on the routing algorithm used. Routing methods can be classified into two types, namely, source routing and distributed routing. In source routing the information about the whole path from the source to the destination is pre-computed and provided in the packet header. In distributed routing, the header contains destination address only and the path is computed dynamically by participation of routers on the way to the destination [5][6][9][10].

Majority of routing algorithms proposed in the literature so far, fall under distributed routing type. Source routing has not been considered much for NoCs, due to its apparent large overhead to store path information in the header. Since the paths in source routing are pre-computed offline, therefore source routing can provide no or limited path adaptivity in the case of faults and traffic congestion. In spite of these disadvantages, source routing has many advantages over distributed 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. According to [8], it can be easily shown that two bits are sufficient to encode every hop in the path. Since the packet entering a router contains the pre-computed decision about the output port, the router design is significantly simplified. Since NoCs used in embedded systems are expected to be application specific, we can have a good profile of the communication traffic in the network [11]. This allows us to analyze the traffic and compute efficient paths giving the desired performance properties, like uniform link load distribution. Source routing also provides possibility of mixing minimal and non-minimal paths for this purpose. We will discuss the advantages of source routing in the NoC context in section II.

A. Related Work

A large number of deadlock free distributed routing algorithms for NoCs have been proposed in literature [12][13]. [11] proposes a methodology to compute deadlock

978-0-7695-4171-6/10 $26.00 © 2010 IEEE
DOI 10.1109/DSD.2010.57

free routing algorithms for application specific NoCs with the goal of maximizing the communication adaptivity. Deterministic source based routing has been considered efficient in scalable parallel systems and multiprocessors. It has been used in IBM SP1 multiprocessor [15]. Although source routing has been shown to be efficient for general networks [9], it has not been explored much for NoC architectures. Recently researchers started considering source routing as a routing candidate in NoCs.

Source routing based scheme has been proposed in [16] to provide guaranteed throughput in a streaming multiprocessor architecture. A class of source routing switches that can be used to efficiently form arbitrary network topologies has been presented in [19]. In [17], a fault tolerant source routing algorithm has been proposed that demonstrates 50% more fault tolerance as compared to the conventional algorithms. When source routing is used, size of routing tables in the network interface is also considered as an overhead. [18] presents an exploration and synthesis of low-overhead configurable source routing tables for network interfaces which results in up to 15 times reduction in the area cost of the NoC routing tables.

A framework to statically determine deadlock free routes using an application aware oblivious routing is proposed in [21]. To support this framework, the authors propose a router design with virtual channels. A number of benchmarks were used for the evaluation. A core mapping technique based on source routing which helps to achieve a mapping with a constraint over the path length is presented in [20]. It demonstrates 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.

In this paper we describe a method to design application specific source routing for mesh topology NoC platforms. Computation of efficient paths for source routing is one of our major contributions in this paper. We demonstrate that the proposed source routing methodology has a potential of achieving better latency performance as compared to the standard adaptive routing algorithms. As the currently available NoC simulators can only handle distributed routing algorithms, therefore, a NoC simulator for evaluating source routing was developed. Similarly, we have also developed a tool called MatPC that computes application specific paths for source routing for a mesh topology NoC resulting in a balanced link load in the network.

B. *Paper Layout*

In section II, we illustrate source routing with an example, present its advantages and disadvantages in NoC context, and formulate research problem and present solution methodology. In section III, we present an algorithm for the selection of a routing algorithm for path computation of source routing, and present and evaluate path improvement algorithms. Section IV depicts the performance evaluation of source and distributed routing and presents the simulation results. In section V, we describe the implications on router design and performance. Section VI concludes the paper.

II. SOURCE ROUTING IN THE NOC CONTEXT

When source routing is used in a NoC, each core (or its interface to the network) contains a table that includes complete route information to reach all the other cores in the network to which it needs to communicate. In order to route a packet through the network, a sender resource looks up the table and adds complete path from source to destination in the packet header. The packet is transferred from the source to the network through resource network interface (RNI). 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 distributed routing, this router does not require any extra computation for making routing decisions because the packets already contain pre-computed decisions.

A. *An Illustrative Example*

Consider an example of a 4X4 mesh topology NoC as shown in Fig. 1. Assume that a DSP is connected to the router (1, 1) has a packet to send to a memory connected to the router (2, 3) as indicated by an arrow in Fig. 1. Also consider that XY routing algorithm is used for this communication. A 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 on the left side of Fig. 2. Similarly, Fig. 2 also depicts the packet format containing destination address instead of complete path if distributed routing was used.

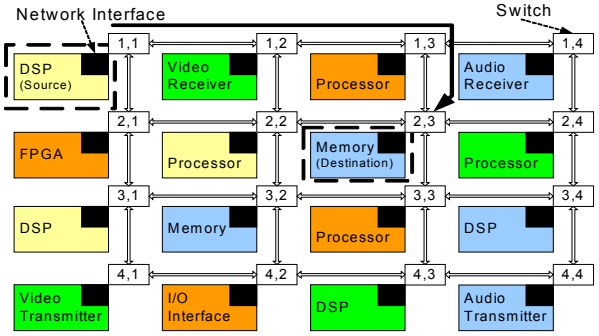

Figure 1. Illustrative example of source routing for a 4X4 mesh NoC.

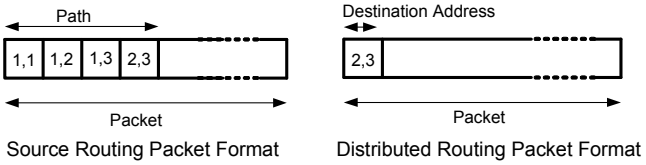

Figure 2. Packet formats for source and distributed routing.

B. *Advantages and Disadvantages Especially for NoCs*

1) Advantages of Source Routing: Source routing has the following major advantages over distributed routing.

- *Simpler, Smaller and Faster 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 and faster as compared to distributed routing.

- *Topology Independence:* Source routing is suitable for both regular as well as irregular topologies. This advantage of source routing is limited by the size of the source table and the maximum length of a route allowed.
- Source routing allows possibility of using minimal, non-minimal or mixed routing paths.
- *Network Size Independent Router:* Since only a constant number of bits of the header are used in every router, its design is independent of the network size. Routers that use source routing 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 distributed routing where destination address field will depend on network size and topology.
- *Balancing of Link Load for Application Specific NoC:* Since NoCs used in embedded systems are expected to be application specific, we can get a good profile of the communication traffic in the network [11]. This allows us to analyse the traffic and compute offline, efficient application specific paths giving the desired performance characteristics like uniform link load distribution.
- *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 and exclusive paths" in the network to such communications [16].
- *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) Disdvantages of Source Routing

- *Routing Overhead:* Packet header in source routing is larger compared to that of distributed routing. Similarly, there is a limitation of the maximum length of the route i.e. the path may not fit in one flit unless some special technique is used.
- *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. Source routing does not take into account the current traffic pattern in the network and it is unable to work in the presence of faults in the network.
- *Limitation of the Size of Source Table:* In source routing, storing large size path tables in sources may become a cost, size and performance overhead for resources, especially for resources which are not of processor type. Solutions have been proposed in literature to reduce this overhead [18].

C. Overhead of Source Routing

As complete route information should be stored along with the payload in case of source routing, large underutilization is expected. In the case of mesh topology NoC and using 2-bit encoding for the router output ports, number of bits required for encoding the routing path for source routing will be double than the network diameter. For a NXN mesh NoC it will be 2(2N-1). On the other hand, the number of bits to encode the address of destination is $2*(\log_2(N))$. As the size of NoC increases, number of routing bits in case of source routing increases at a much higher rate as compared to that of distributed routing. This comparison is graphically shown in [8]. This overhead apparently makes source routing look unusable in practice.

But if the overhead is measured in terms of extra flits or bytes to be communicated, the difference is rather small. Source and distributed routing in NoC are compared on the basis of bandwidth utilization in [8] and it is shown that for practical size NoCs, bandwidth utilization gap is negligibly small. 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.

D. Problem Formulation and Solution Methodology

Our goal is to find a methodology to offline compute application specific efficient paths for source routing in mesh topology NoC leading to a high degree of link load balancing in the network. The problem is solved using the following steps. A complete solution methodology is shown in Fig. 3.

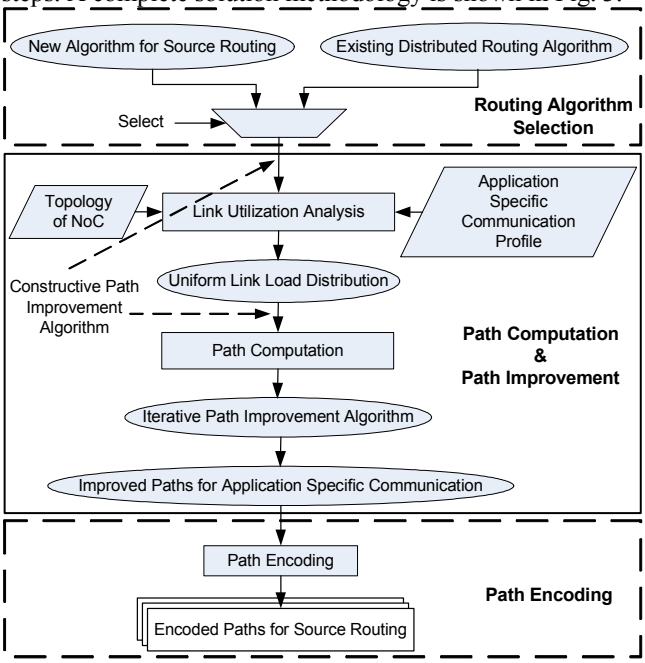

Figure 3. Complete solution methodology.

1) Selection of the most suitable routing algorithm from a set of available routing algorithm: Communication paths can be computed using an existing deadlock free algorithm or one can develop a new application specific routing algorithm. An issue regarding the first option is that a choice has to be made for selecting the most suitable routing algorithm for source routing from many existing deadlock free distributed routing algorithms. The selected (possibly

adaptive) routing algorithm should be able to compute efficient static paths with the goal of balancing link load in the network.

2) Path Computation: Size of the NoC, information about which pairs in the network communicate and communication volumes between pairs are inputs to the method. We assume that this information is available from offline profiling of the application. Based on selected routing algorithm and the inputs, paths for source routing are computed with a goal of uniform link load distribution.

3) Path Improvement: Once paths for source routing are computed based on most suitable routing algorithm, the next problem is the improvement of computed paths with a focus on more uniformly distributing traffic on links. In this paper, we propose two path improvement algorithms.

4) Path Encoding: Since routing information in packet header is an overhead in source routing, an efficient scheme is also required for encoding of router ports with minimum number of bits to reduce the header overhead. As a solution, we use a simple and efficient encoding scheme called "2-Bit Clockwise Router Port Address Encoding".

III. COMPUTATION OF EFFICIENT PATHS FOR SOURCE ROUTING

Path computation refers to finding a complete path or route from source to destination. In general case, a path should be computed for each pair of resources in the network. In an application specific context, the paths are computed only for those pairs of resources which communicate. The computed paths must avoid any possibility of deadlock. It should also try to provide small packet delay by avoiding link congestion and distributing traffic uniformly in the network.

A. Routing Algorithm Selection

One can easily think about two distinct options for computing paths for source routing. First is to compute paths by using an existing distributed routing algorithm and the second is to devise a new method or mix existing distributed routing algorithms. In this paper, we propose a two step approach. First, initial paths for source routing are computed by using the most appropriate existing deadlock free deterministic or partially adaptive distributed routing algorithms for mesh topology NoC. For the time being we consider five well known routing algorithms i.e. XY, West-First, North-Last, Negative-First and Odd-Even. Second, initial paths are modified to improve link load balancing which indirectly improves communication performance.

The basic motivation in the first step is that different routing algorithms perform best for a set of particular traffic patterns. Therefore we can select the most appropriate routing algorithm based on the analysis of the communication patterns in the application. By empirical analysis, we observed that each routing algorithm had different performance for different traffic pattern. The performance of a routing algorithm was evaluated on the basis of standard deviation of the link load distribution. Link load is defined as the amount of data flowing on each link in NoC. We performed a large number of experiments using a 7x7 mesh topology NoC. In these experiments, link bandwidth volume of each communication was uniform randomly selected from 1-10. Communication density was also uniform randomly selected in such a way that each core communicates with 2 to 5 other cores in NoC. Locality biased traffic is used in the application specific case when we consider east, south and west dominated traffic. Table I shows the performance comparison of Odd Even routing algorithm with other routing algorithms in hot spot traffic.

TABLE I. PERFORMANCE COMPARISON OF ODD EVEN ROUTING ALGORITHM WITH OTHER ROUTING ALGORITHMS IN HOT SPOT TRAFFIC

Routing Algorithm	Performance Comparison of Odd Even Routing Algorithm for Hot Spot Traffic
XY	16.17 % Better
West First	21.47 % Better
Negative First	25.93 % Better
North Last	15.34 % Better

Results of the experiments lead us to a solution proposal for the selection of the best routing algorithm for each type of traffic to compute paths for source routing and it is shown in Fig.4. A communication graph is obtained after identifying all the communications among resources for a specific application and then the traffic is analyzed. Accordingly, West First, North Last, East First and Odd-Even algorithms should be selected for east, south [13], and west dominated and hot spot traffic respectively [8]. Similarly, for any other traffic including random, XY routing algorithm should be selected.

After selection of routing algorithm, paths from source to destination for all communicating core pairs can be computed. Whenever at any intermediate node there is a choice of multiple output ports, a port is randomly selected. For this analysis and path computation, we have developed a Matlab based tool called MatPC [8].

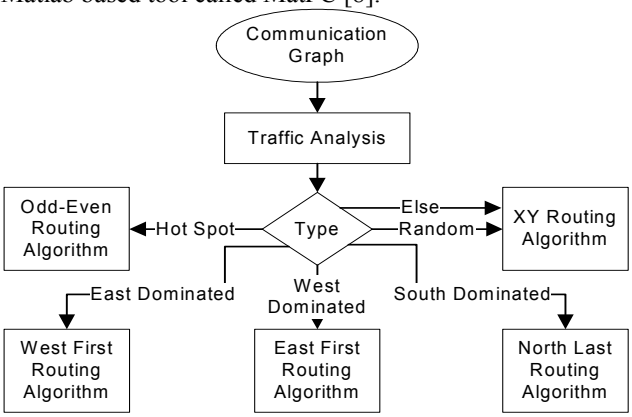

Figure 4. Algorithm for traffic analysis and selection of routing algorithm for source routing using application specific traffic.

B. Path Improvement

Paths computed for source routing using algorithm shown in Fig. 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.

1) Constructive Path Improvement Algorithm: Link load in the NoC can be balanced to some extent during the path computation process as shown in Fig.5a. 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.

2) Iterative Path Improvement Algorithm: After analyzing and selecting routing algorithm as shown in Fig. 4, 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 Fig. 5b.

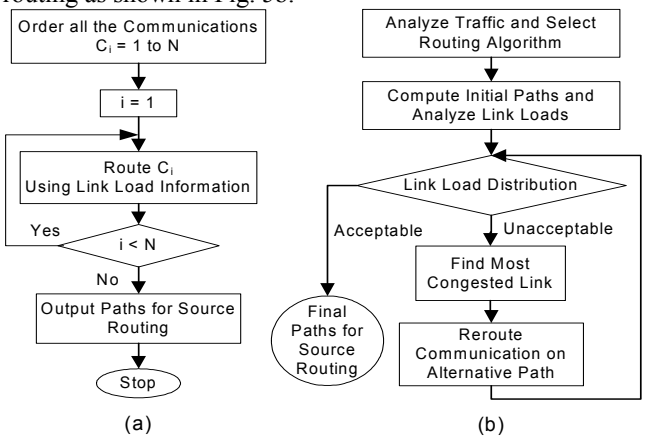

Figure 5. Path improvement algorithms: (a) Constructive (b) Iterative.

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 is tabulated in Table II. Maximum percentage path improvement is also shown.

TABLE II. RESULTS: PATH IMPROVEMENT IN EACH ROUTING ALGORITHM USING VARIOUS TRAFFICS

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

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.

IV. PERFORMANCE EVALUATION

For evaluation of source routing, we enhanced an existing simulator that was earlier developed by [7] for distributed routing. It takes input from MatPC tool.

A. Comparison with Corresponding Distributed Routing

In this section, 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 Fig. 6. It is evident from the graph that latency in the case of source routing is much lower than that of distributed routing.

Two regions in the latency graphs are encircled as (a) and (b). Region (a) shows the latency for low network load while

region (b) shows the latency for the load when the network starts to saturate. These regions are magnified and shown in Fig. 7. APL in source routing is about 12 cycles lower than that of distributed routing at lower network load as shown in Fig 7a. Lower latency in source routing is because of the faster router. In the simulator, router delay for source routing was set to be one clock less than the corresponding distributed router. The difference in latency keeps on increasing as the network load is increased until the network starts to saturate. Fig 7(a) also shows that source routing has a potential of achieving higher performance comparatively, for example up to 28% lower APL even at medium load.

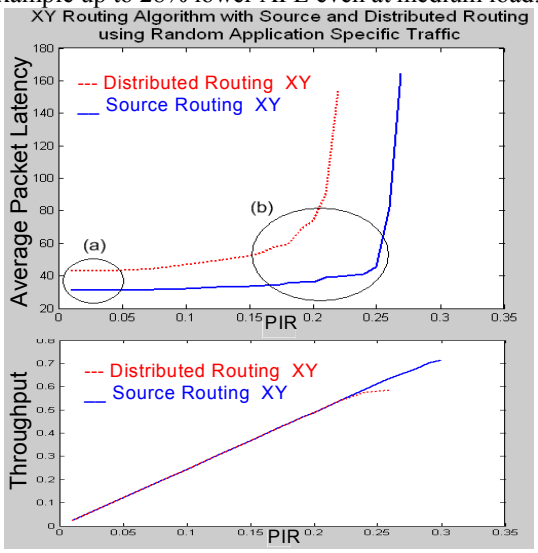

Figure 6. APL vs. PIR and Throughput vs. PIR for a 7x7 NoC for source & distributed routing using XY routing algorithm.

One remarkable advantage of source routing can be seen from the latency graph near saturation region shown in Fig. 6. 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.

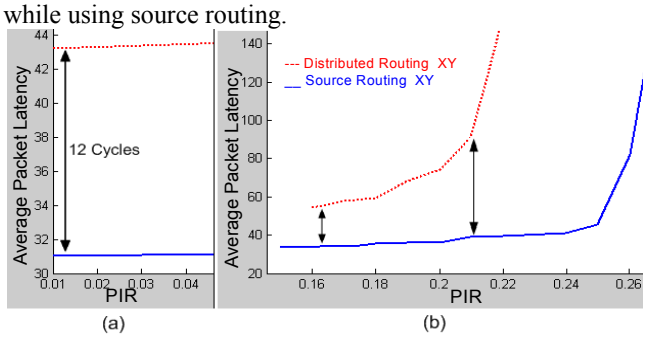

Figure 7. Magnified regions (a) and (b) from latency graphs in Figure 6.

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 Fig. 6. 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, source routing provides comparatively higher throughput.

Similarly, APL and throughput graphs for source and distributed routing using Odd Even routing algorithm in random traffic are shown in Fig. 8. APL and throughput of source routing, distributed routing before and after path improvement is shown by solid, dotted and "+" signs curves in Fig. 8. Two regions in the latency graphs are encircled as (a) and (b). Region (a) shows the latency for low network load while region (b) shows the latency for the load when the network starts to saturate. These regions are magnified in Fig.9. Advantage of path improvement in source routing is obvious at higher network load.

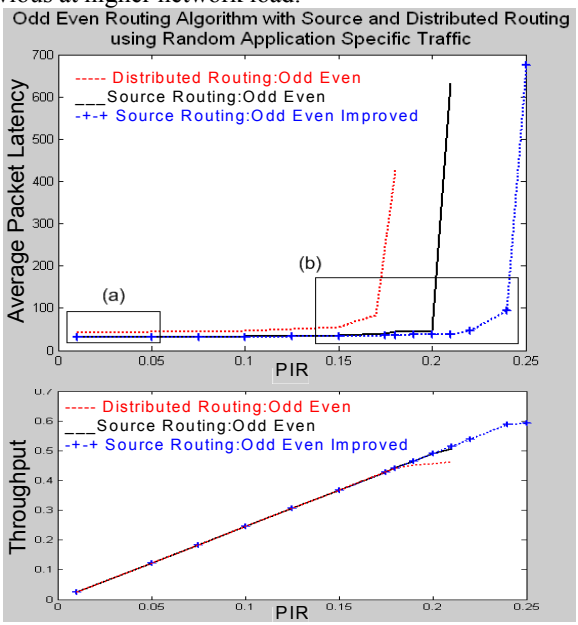

Figure 8. APL vs. PIR and Throughput vs. PIR for a 7x7 NoC for source and distributed routing using Odd Even routing algorithm.

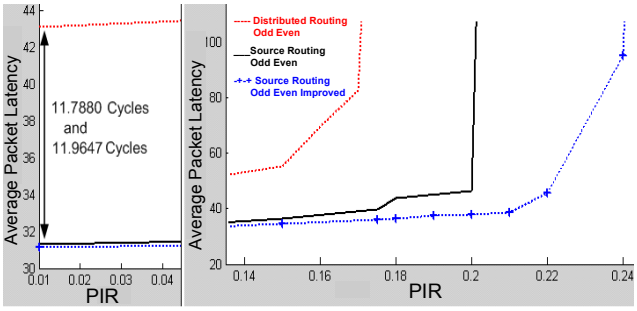

Figure 9. Magnified regions (a) and (b) from latency graphs in Figure 8.

B. *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 [8]. In

Fig. 10(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. These results also support the results presented in Table 2.

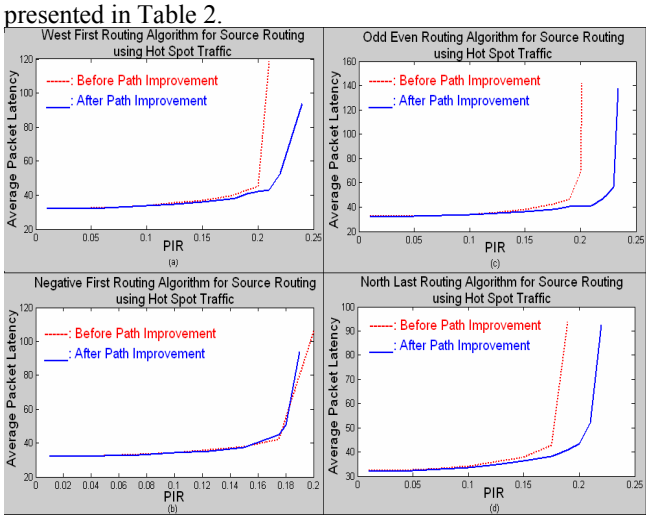

Figure 10. APL plotted against PIR for source routing using various routing algorithms before and after path improvement in a 7X7 mesh NoC.

C. *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 Fig. 11. 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 source routing as shown in Fig. 4. 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].

V. Implications on Router Design and Performance

A. *Path Encoding*

After path computation, paths should be encoded in the packet header or more precisely in the header flit in such a way that the routing overhead is minimized and the encoded information is easy to decode in the routers.

1) 2-Bit Clockwise Router Port Address Encoding Scheme: In order to minimize the route information overhead in the head flit, we implement a 2-bit clockwise router port address encoding scheme [8]. We assume that a flit cannot be forwarded to the same channel from where it came. Hence, a flit can only be forwarded to any of four out of five output ports in a router for a 2-D mesh NoC. Address of each output port of a router is encoded with two bits. Accordingly, "00", "01", "10" and "11" represent the addressees of output ports which are one, two, three and four steps away respectively in the clockwise direction with respect to the input port where flit was received.

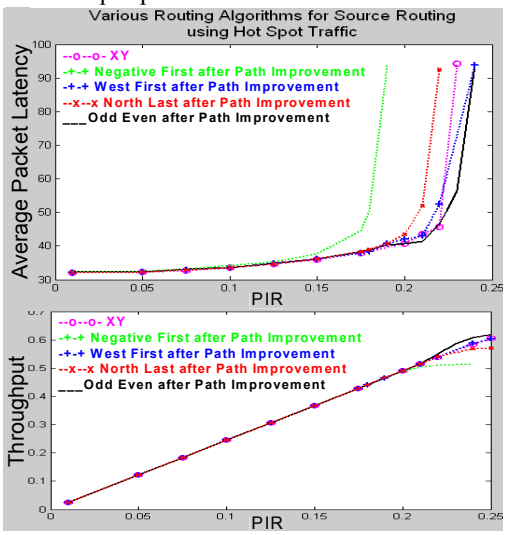

Figure 11. APL and Throughput vs. PIR in hot spot traffic using various routing algorithms after path improvement in a 7X7 mesh NoC.

Consider the example illustrated in *Section II-A*. The packet header will contain addresses of all the routers traversed as is shown in Fig. 12a. Corresponding encoded packet header with 2-bit clockwise router port address encoding scheme is depicted in Fig 12b. A packet coming from DSP resource to router (1, 1) is to be routed to router (1, 2) and which is connected to the east port of router (1, 1). East port of this router is three steps in the clockwise direction with respect to the incoming port as shown in Fig. 12c. Thus first position of packet header is encoded as binary "10"and shown in Fig. 12b. Similarly, at router (1, 1) the packet is to be routed again to east port where router (1, 3) is connected. Hence, east port is two steps in clockwise direction with respect to the packet incoming port as shown in Fig. 16d. Thus, second position of packet header is encoded as binary "01". Rest of the positions in the packet header is also encoded in the same fashion.

B. *Router Architecture*

Router design for source routing is much simpler than the router design to handle distributed routing. It does not need to compute a routing function to select the output port for an incoming packet. This pre-decided information is available in the header flit. The router 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. This router still needs to implement 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. Simplicity of the source router makes it relatively smaller and faster. We

designed and implemented a source router, with our own encoding scheme, whose detailed architecture and design decisions were presented in [8]. The schematic diagram of source router is depicted in Fig. 13.

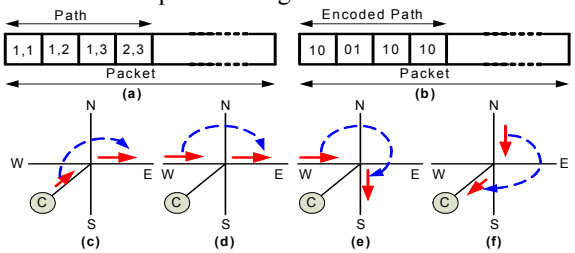

Figure 12. (a,b) 2-bit clockwise router ports address encoding in a packet header (c,d,e,f) Packet arrival input port and destination output port in each router for the communication shown in Fig. 1.

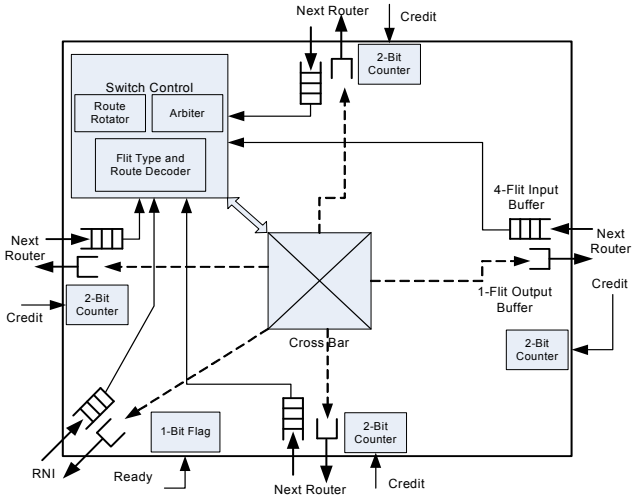

Figure 13. Schematic diagram of a NoC Router for Source routing.

VI. CONCLUSION AND FUTURE WORK

In this paper we have 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 have 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 have 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 have also proposed a very simple but efficient method for encoding ports of the router.

For evaluation of our source routing approach, we developed a simulator. Evaluation results show that source routing gives higher latency and throughput performance as compared to corresponding distributed routing. We also designed a router to support our source routing ideas.

Our current approach restricts the computed paths in both steps to be of minimum length. We plan to extend our approach for allowing non-minimal paths also in the second step. Development of a routing scheme which combines source and distributed routing will be another interesting direction for future research. We also plan to prototype a small NoC platform based on source routing to work out and compare router's hardware cost with the cost of the router in a corresponding platform using distributed routing.

REFERENCES

[1] Dally W.J. and Towles B. "Route packets, not wires: on-chip interconnection networks", in Design Automation Conference, Las Vegas, Nevada, USA, 2001, pp. 684-689.

[2] Kumar S. et. al., "A network on chip architecture and design methodology". in IEEE Computer Society Annual Symposium on VLSI, Pittsburgh, April 2002, pp. 117-124.

[3] Benini L., Micheli G.D., "Networks on chips: a new SoC paradigm"., IEEE Computer Society, 2002.

[4] J. Duato et al., "Interconnection network: an engineering approach", Elsevier Health Sciences, UK, (9781558608528), 2002.

[5] 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.

[6] Axel Jantsch and Hannu Tenhunen, "Networks on chip", Kluwer Academic Publishers, ISBN: 1-4020-7392-5, 2003.

[7] Holsmark R. and Högberg M. "Modeling and prototyping of network on chip". M.S. thesis, Jönköping University, Sweden, 2002.

[8] Mubeen S., "Evaluation of source routing for mesh topology network on chip platforms". M.S. thesis, Jönköping University, Sweden, 2009. Permanent link: http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-9591.

[9] J. Flich, P. lopez, M. P. Malumbers and J. Duato, "Improving the performance of regular networks with source routing". In the IEEE International Conference on Parallel Processing. 21-24 Aug, 2000.

[10] Y. Aydogan, C.B. Stunkel, C. Aykanat, B. Abali., "Adaptive source routing in multistage interconnection networks" in Proceedings of IPPS '96, the 10th International Parallel Processing Symposium., 15-19 April 1996, pp. 258 – 267.

[11] Palesi M., Holsmark R., Kumar S. and Catania V., "Application specific routing algorithms for networks on chip", IEEE Transactions on Parallel and Distributed Systems, vol.20, No.3, March, 09.

[12] Ge-Ming Chiu. "The odd-even turn model for adaptive routing". IEEE Transactions on Parallel and Distributed Systems, vol. 11, No. 7, July 2000.

[13] A. Patooghy, H. Sarbazi-Azad, "Performance comparison of partially adaptive routing algorithms". In 20th International Conference on Advanced Information Networking and Applications, April 2006.

[14] Evgeny Bolotin, Israel Cidon, Ran Ginosar, Avinoam Kolodny. "QNoC: QoS architecture and design process for network on chip". Journal of Systems Architecture 50 (2004), pp.105–128.

[15] Stunkel, C.B. Shea, D.G. Grice, D.G. Hochschild, P.H. Tsao, M. "The SP1 high-performance switch", in Proceedings of the Scalable High-Performance Computing Conference, 23-25 May, 1994.

[16] Kavaldjiev N. et. Al. "Routing of guaranteed throughput traffic in a network-on-chip". Available at: http://doc.utwente.nl/54538/.

[17] Young Bok Kim, Yong-Bin Kim, "Fault tolerant source routing for network-on-chip". In 22nd IEEE Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007), 2007. pp.12-20.

[18] Loi, I.; Angiolini, F.; Benini, L. "Synthesis of low-overhead configurable source routing tables for network interfaces". In Design, Automation & Test in Europe Conference, 2009. pp: 274-279.

[19] Ma Liwei and Sun Yihe. "On-chip network design automation with source routing switches". Tsinghua Science and Technology, volume 12, issue 1, February 2007. pp 77-85.

[20] Tornero R., Kumar S., Mubeen S. and Orduna J.M. "Distance constrained mapping to support NoC platforms based on source routing". In 3rd Workshop on Highly Parallel Processing on a Chip in conjunction with Euro-Par 2009. 25-28 August , 2009.

[21] Kinsy, M. A. et al. "Application-aware deadlock-free oblivious routing." In the 36th Annual international Symposium on Computer Architecture ISCA 2009. June 20 - 24, 2009.