2014 Fourth International Conference on Advances in Computing and Communications

A Novel Energy Efficient Source Routing for Mesh NoCs

Meril Rani John, Reenu James, John Jose, Elizabeth Isaac, Jobin K. Antony
Rajagiri School of Engineering and Technology, Kochi, Kerala, India
(merilrani, reenujms)@gmail.com, (johnj, elizabeth, jobinantony)@rajagiritech.ac.in

***Abstract*—Network on Chip (NoC) is the upcoming interconnection framework for multi-core processors. In chip-multicore processors, the performance of the underlying communication network is determined by the efficiency of the routing logic used in NoC routers. Distributed routing and source routing are the two main classification of routing techniques. For routers employing static routing policies, distributed routing leads to redundant computations in each router and increases complexity of routers. We analyse the existing state-of-art source routing technique and propose a novel alternative that makes routing process simpler and faster. Our proposed design reduces the number of bits needed to encode the route in the packet header, reduces the router pipeline latency and hence allows the network to be operated at a frequency approximately two times faster than the existing design.**

***Keywords*-route string; route pre-computation; route extraction; routing latency**

I. Introduction

As VLSI technology improves, the number of processing cores that can be integrated on a single chip has increased [1]. Generally, a shared bus system facilitates inter-core communication in a multi-core processor system. This bus-based interconnection cannot scale with the increase in the number of cores. Network on Chip (NoC) has been widely employed instead of traditional bus-based on-chip interconnects for such many core systems. In NoC, transmission of inter-core messages is in the form of packets. Packets are further subdivided into sequence of flow control units called flits. Flit is the basic unit of data transfer between a pair of routers [2].

NoC consists of an array of routers connected by well structured links as shown in Figure 1. The figure represents the interaction of a processing core with a router in a 2D-mesh topology. A Network Interface Controller (NIC) is associated with each core that connects it to the network. Each NoC router is connected to four of its neighbours along the North, East, South and West directions. These links carry data packets from one router to another [3], [4]. Typically each processing core consists of an out-of-order superscalar processor, a private L1 cache and a shared distributed L2 cache bank. NoC traffic is initiated during cache misses, coherence transactions and inter-core synchronization messages.

Packets are forwarded from source router to destination router based on the routing logic embedded in the NoC router. Based on when and where routing decisions are made, there are two types of routing; distributed routing and source routing [1]. In distributed routing, the destination address of the packet is embedded in the packet header. By using this, each intermediate router determines the direction of the next hop for the packet. Thus the path taken by a packet towards destination is computed collectively by a set of intermediate routers. To facilitate this, each router contains a computing logic that compares its address with the address of the destination of the incoming packet [5].

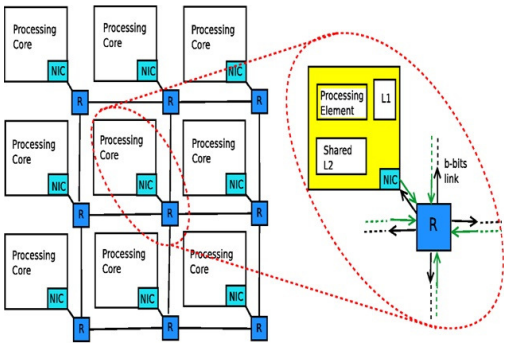

Figure 1. Processing core- router interaction in a 2D-mesh topology.

In source routing, the information about the whole path from the source to the destination is pre-computed and meaningfully encoded in the packet header. This encoded value is used in every intermediate router to compute the corresponding output directions [6].

Figure 2 shows a conventional NoC router with Virtual Channels (VCs) in the input ports. These VCs consist of buffers to accommodate incoming packets from four different directions as well as the local core. The routing unit determines the output port as well as the VC for the next downstream router for each packet. The VC allocator arbitrates amongst all packets requesting access to the same VCs. In switch allocator, arbitration amongst all packets requesting access to the same crossbar output is carried out. These winning packets traverse through the crossbar and reach their respective output ports. If there is more than one packet requesting for the same output port, only one packet makes the forward movement (based on switch allocation) while others remain in the VC buffers until they get a productive port in the subsequent clock cycles [5].

978-1-4799-4363-0/14 $31.00 © 2014 IEEE
DOI 10.1109/ICACC.2014.36

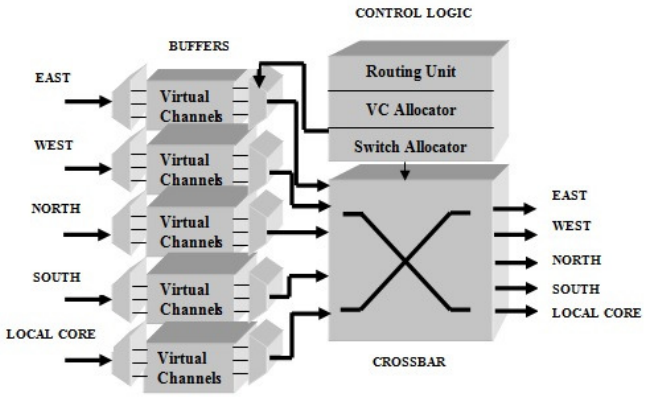

Figure 2. Conventional input buffered router.

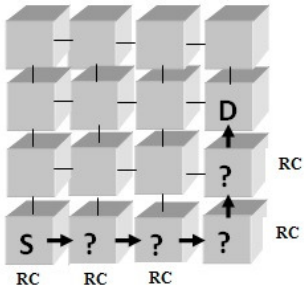

Figure 3. Distributed XY routing.

In this paper, we propose a novel source routing technique for 2D-mesh NoCs. The rest of the paper is organised as follows. A brief description of the distributed XY routing is covered in Section II. In Section III, we explain the motivation for the proposed work. An energy efficient source routing technique is proposed in Section IV. Implementation details and experimental results are given in Section V and we conclude our work in Section VI.

II. Distributed XY routing

XY routing is the most common static routing algorithm used in mesh NoCs due to its simplicity and deadlock free design [1]. It routes packets first in X-direction (or horizontal direction) until it reaches the same column as the destination and then in Y-direction (or vertical direction) until it reaches the destination. The packet will take atmost one 90°turn on its path to destination. The path consists of zero or more number of X-hops followed by zero or more number of Y-hops.

We have already mentioned that in distributed routing, *route computation* [RC] is done at each intermediate router by comparing the destination address of the packet with the address of the current router. At the time of packet generation, the packet is not aware of the path it must take to reach the destination. In each intermediate router, the next output port is computed by applying the XY routing method.

We will now illustrate distributed XY routing with the help of an example. Consider a 4x4 mesh network as shown in Figure 3. A packet is generated from router S and is destined to router D. Based on the destination address present in the packet header, the next downstream router is computed as per XY routing at router S. In this case, the East direction is selected as the output and the packet is forwarded to the East neighbour of router S. The *route computation* is repeated at every intermediate router until the packet reaches router D. In this example, the packet takes three hops towards East and then two hops towards North to reach its destination router. Then the packet is forwarded to the local core.

III. Motivation

For a static routing like XY routing, the presence of *route computation* logic at each router increases the router complexity. This increases the router pipeline latency and decreases the operating frequency of the router. In an NoC with mesh topology and static routing like XY routing, the routing decision (output port taken) is same whether it is computed at the source or at the intermediate routers. A computation intensive distributed XY routing can be replaced with a light logic source routing. Using source routing, the path information can be efficiently encoded in a packet header with a few number of bits. Since the packet entering a router contains the pre-computed decision about the output port, the router design is significantly simplified. Such an implementation will be simpler and faster as compared to distributed algorithms.

Considering the simplicity of routing logic, a source routed XY routing is implemented in [7]. They consider a 2 bit encoding for four different directions. In a 4x4 network, a packet can take a maximum of six hops to reach its destination. Hence a 12 bit pre-computed path is encoded in the packet header at the time of source routing. If we apply this policy in 6x6 and 8x8 networks, we need a maximum of 20 and 28 bits respectively to encode the path.

We carefully study the existing source routing design [7] and observe that we could encode the route with fewer number of bits in the packet header. Reducing the number of bits in packet header improves the energy efficiency of the system. Rather than encoding each hop by a 2 bit vector, we encode a minimal *route string*. The next section elaborates on our proposed source routing technique. We experimentally prove using Verilog synthesis that the proposed source routing technique has a potential of reducing the router pipeline latency as compared to the state-of-art designs.

IV. The Proposed Source Routing

In the proposed source routing, *route computation* is done only at the router which generates the packet. The entire path information needed for the packet to reach the destination is determined and encoded efficiently in the packet header as a *route string* at the time of packet generation. The downstream routers extract the required output port from the *route string* that is already computed at the source. Since computation of path is done only once, it reduces the latency of the *route extraction*[RE] logic.

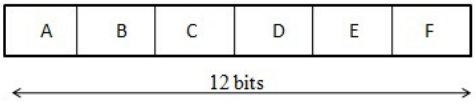

Figure 4. *Route string* format in the Existing Source Routing [7].

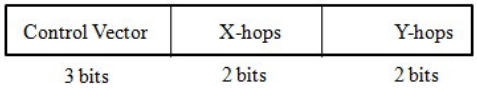

Figure 5. *Route string* format in the Proposed Source Routing.

Figure 4 shows the *route string* format in the existing source routing (ESR) [7]. Here each element (A/B/C/D/E/F) represents a pair of bits indicating the next output hop direction. The encoding in ESR is as follows: 00-North, 01-East, 10-South, 11-West. We propose a novel encoding for the *route string* which consists of three sections, namely Control Vector (3-bits), X-hops (2-bits) and Y-hops (2-bits) as shown in Figure 5.

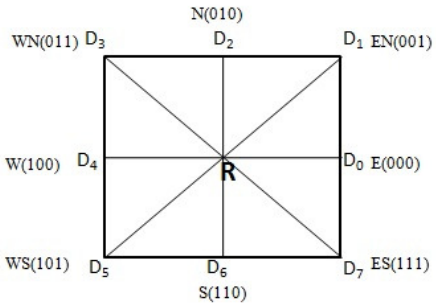

Figure 6. Quadrant routing and Control Vector logic.

We use quadrant routing logic [8] which uses a 3-bit Control Vector in encoding the relative direction of the movement of the packet with respect to the source router to reach its destination. The quadrant routing logic for generation of Control Vector is explained in Figure 6. Consider the location of the source router as R and the relative position of the destination router as any one of D_i. For a packet whose destination lies in the first quadrant D_1, we assign the Control Vector as 001. A packet whose destination is in the same column as the source router, say D_6 we assign the Control Vector as 110. Similarly the vectors associated with each possible direction is shown in Figure 6. The packets whose destination is in the same row or column as the source is given an even vector and others are assigned an odd vector. After the computation of the three bit Control Vector, we compute the number of X-hops and Y-hops needed for the packet to reach the destination by finding the difference in X and Y co-ordinates of source and destination routers. Since we use a 4x4 mesh, the X-hops and Y-hops can be maximum three each, thereby we can represent each one of them as two bits.

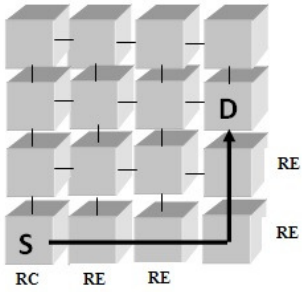

Figure 7. Proposed Source Routing.

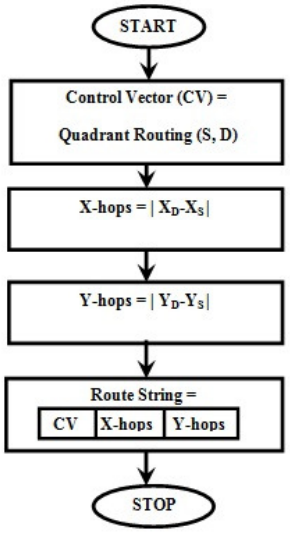

Figure 8. *Route Computation* (RC) Process (Input is Source Node- S(X_S, Y_S) and Destination Node- D(X_D, Y_D); Output is *Route String).*

Figure 7 shows the propagation of a packet in a 4x4 network using the proposed source routing. Router S generates the packet and router D is its destination. The

source router determines the Control Vector, X-hops and Y-hops and embed the *route string* in the packet header as mentioned above. Each intermediate router now performs *route extraction* and determines the next output port. Based on this path, the packet reads the first direction and moves towards East. After it reaches the new router, the next direction is read from the *route string*. This process continues till the packet reaches the destination. Here the packet takes three hops towards East and then two hops towards North. The packet gets forwarded to the local core once it reaches router D. Thus *route computation* is avoided in each router and only *route extraction* is needed.

The process flow for proposed source routing is shown in Figures 8 and 9. The *route computation* in the source router computes the *route string*. Each intermediate router performs the *route extraction* on the packet from the predetermined *route string*.

V. Implementation and Experimental Results

The routing logic discussed above are implemented in Verilog and synthesised in Xilinx ISE 12.2 [10]. The device chosen for synthesis is Spartan 6. We have modelled a 4x4 mesh NoC using the cycle accurate on-chip interconnection simulator, Booksim [9]. The router pipeline latency obtained from Xilinx is used in Booksim to get the average packet latency in nanoseconds.

Routing Technique	Delay in ns	
Distributed Routing	6.22	
Existing Source Routing	5.155(RC)	2.575(RE)
Proposed Source Routing	1.451(RC)	2.636(RE)

Table I
Latency of various routing algorithms using Verilog Synthesis.

Since distributed XY routing incorporates computation of the next output port in every hop, it takes high routing delay for existing source routing. The total delay involved in both the source routing techniques is less when compared to distributed XY routing. From the Verilog synthesis results (Table I), we can see that in existing source routing, the *route computation* logic dominates over *route extraction* logic and for the proposed logic, *route extraction* dominates over *route computation*. The router pipeline latency is 5.155 ns for existing sourc routing technique and 2.636 ns for the proposed technique, thereby acheiving a reduction in router pipeline latency of 48.86%. From the above observations, we find the maximum network frequency for distributed XY routing, existing source routing and proposed source routing as 161 MHz, 194 MHz and 380 MHz respectively. This is shown in Figure 10. Hence we conclude that by using our proposed source routing, we can operate the network at a frequency 1.96 times higher than the ESR design.

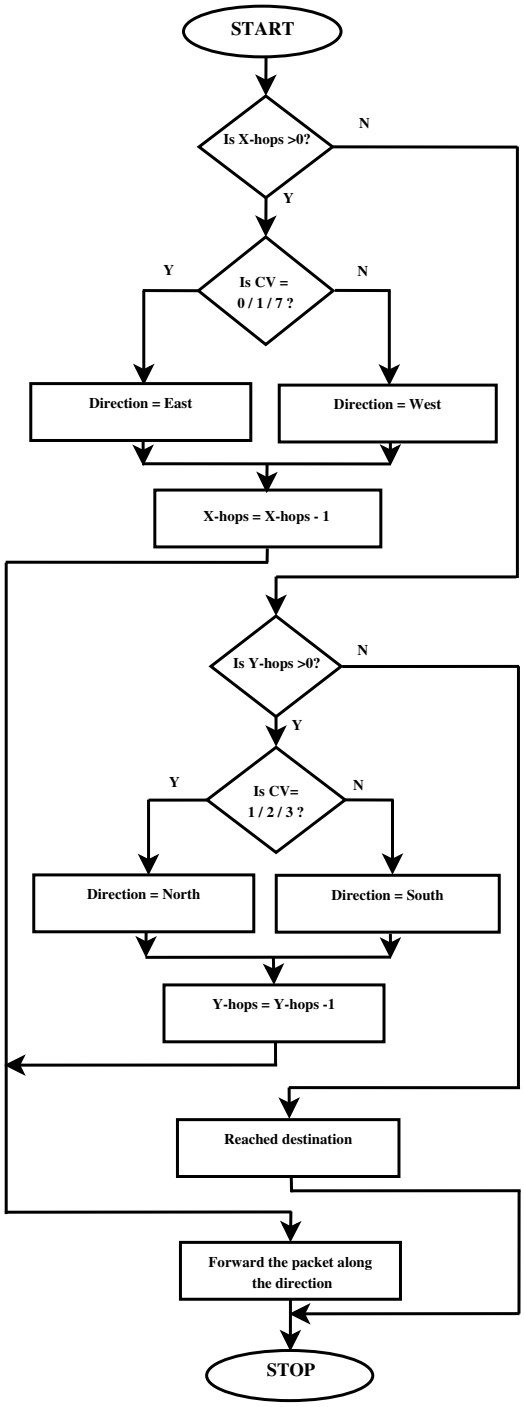

Figure 9. *Route Extraction* (RE) Process (Input is *Route String* and Output is next hop direction).

Another significant contribution of our PSR logic is the reduction of the packer header size. The existing source routing technique needs two bits to represent each hop taken by the packet. Thus the number of bits used to represent the path increases linearly with network size as shown in

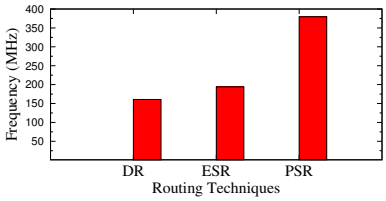

Figure 10. Comparative analysis of maximum network frequency using various routing techniques (DR: Distributed XY Routing, ESR: Existing Source Routing, PSR: Proposed Source Routing).

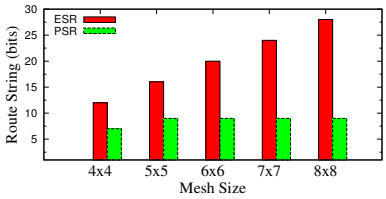

Figure 11. Comparative analysis of route string size needed for various source routing techniques (ESR: Existing Source Routing, PSR: Proposed Source Routing).

Figure 11. The proposed source routing technique is energy efficient as it uses lesser number of bits to represent the path. We can see that, the *route string* size is constant for our proposed technique upto 8x8 network. Hence the proposed technique becomes highly energy efficient as the network size increases.

Comparative analysis of average packet latency for various traffics such as uniform, transpose and bit-complement in 4x4 mesh NoC at pre-saturated load is shown in Figure 12. The proposed source routing technique reduces the latency by 51.51%, 50% and 47.83% in Uniform, Transpose and Bit-complement traffics respectively. We observe that across all traffic patterns, the proposed source routing technique gives the minimum average packet latency. Thus we prove that by using proposed source routing, we could design an NoC, where packets are routed faster to their respective destinations.

VI. Conclusion

Through this paper we emphasised the use of source routing in mesh NoCs with static routing policies. An energy efficient source routing algorithm is implemented and the results obtained are compared with the existing techniques. Results showed that router pipeline latency is less for the proposed design and hence the on-chip network which uses our routing logic can be operated at a higher frequency. We hope that our proposed design will be able to reduce the gap between processor core frequency and network frequency in future NoCs.

VII. Acknowledgement

The authors would like to thank the management of Rajagiri School of Engineering and Technology for providing the infrastructure facilities for carrying out the experiments.

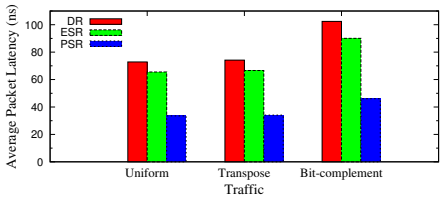

Figure 12. Comparative analysis of average packet latency for various traffics in 4x4 mesh NoC at pre-saturated load (DR:Distributed XY Routing, ESR: Existing Source Routing, PSR: Proposed Source Routing).

In addition to this, the authors also like to put a record of appreciation to Manu Varghese Mathew, P.B. Kevin, Rogen George and Shashank Gangadharan for the initial help given in setting up the experimental framework.

References

[1] W. Dally and B. Towles *Principles and Practices of Interconnection Networks* USA: Morgan Kaufmann Publishers Inc., 2003.

[2] W. Dally and B. Towles *Route packets, not wires: on-chip interconnection networks* in proceedings of the Design Automation Conference, 2001, pp. 684-689.

[3] J. Henkely et al., *On-chip networks: A scalable, communication-centric embedded system design paradigm* in proceedings of the International Conference on VLSI Design, 2004, pp. 845-851.

[4] T. Y. Terry et al., *Packetization and Routing Analysis of On-Chip Multiprocessor Networks*,Journal of Systems Architecture, Volume 50, Issue 2, 2004, pp. 81-104.

[5] L. Benini and D. Bertozzi *Network-on-chip architectures and design methods* in proceedings of the Computers and Digital Techniques, 2005, pp. 261-272.

[6] T. Bjerregaard et al., *A Survey of Research and Practices of Network-on-Chip* ACM Computing Surveys, Volume 38, Issue 1, 2006, pp. 1-51.

[7] S. Mubeen et al., *Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms* in proceedings of the Euromicro Conference on Digital System Design, 2010, pp. 181-188.

[8] J. Jose et al., *DeBAR: Deflection based adaptive router with minimal buffering* in proceedings of the Annual International Conference on Design, Automation and Test in Europe, 2013, pp. 1583-1588.

[9] N. Jiang et al., *A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator* in proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, 2013, pp. 86-96.

[10] www.xilinx.com.