Power-Efficient Routing Algorithm for Torus NoCs

D. Rahmati, A. E. Kiasari[1], H. Sarbazi-Azad[1] and S. Hessabi

Department of Computer Engineering, Sharif University
[1]IPM School of Computer Science, Tehran, Iran
E-mail : {d_rahmati,kiasari}@ce.sharif.edu, {azad,hessabi}@sharif.edu

ABSTRACT

Modern System-on-Chip (SoC) architectures use Network-on-Chip (NoC) for high-speed inter-node communication. NoC with torus interconnection topology is now popular due to its low dimension and simple structure. Torus NoC is very similar to the mesh NoC from a structural point of view, but has rather smaller diameter that makes it a suitable choice for NoCs. For a routing algorithm to be deadlock-free in a torus NoC at least two virtual channels should be used to avoid channel dependency, while mesh NoC can handle deadlock freedom using only one virtual channel. In this paper, we propose a novel approach on designing routing algorithms for mesh and torus NoCs. Also a deadlock free routing algorithm is proposed for Torus NoC that uses only one virtual channel per physical channel resulting in lower power consumption because of reduced hardware complexity and with no significant performance degradation. The algorithm works within a dimension and is applied to all dimensions individually for XY routing and various turn based deterministic routing algorithms like west first, north last and negative first. We have proved efficiency of the algorithm using simulation results obtained from synthesis of our implemented VHDL Register Transfer Level (RTL) model of NoC.

Keywords: *SoC, NoC, Torus, Mesh, Performance, Power Consumption, Routing, Virtual Channel, Deadlock, VHDL RTL model.*

1. INTRODUCTION

The simplest and hence widely used routing algorithm for the mesh NoCs is XY routing [1,2,3,8]. In this algorithm the packet is routed across the X axis and then across the Y axis until it reaches the destination node as shown in Fig.1. Since there are no wraparound links to connect the first and last nodes in each dimension, XY routing algorithm is deadlock free using only one virtual channel.

However, applying XY routing for the torus NoC may cause deadlock as a result of the channel dependency in each dimension between different messages [8]. By using more than one virtual channel there will be the flexibility of designing different deadlock free routing algorithms in the cost of hardware complexity, more area, and thus higher power consumption. Power consumption is the most important factor in the design and implementation of NoC architectures, while performance (network latency and throughput) is the key factor in multicomputer networks.In order to have a deadlock-free routing algorithm in the torus NoC, there should be at least two virtual channels to break the cyclic channel dependency, caused by wraparound links, into a spiral [4, 8, 10]. This is not the case when mesh NoCs are used without wrap-around links and thus requiring only one virtual channel. It is also shown that the number of virtual channels has a crucial effect on power consumed by the NoC [5, 6, 12].

In this paper, we first introduce IRN (Interconnection Routing Notation), a map-based

systematic approach on designing routing algorithms for mesh and torus NoCs. This notation is also extendable for other interconnection topologies. We then use IRN and propose a deadlock free routing algorithm called TRANC (Torus Routing Algorithm for NoC) for the torus NoC that uses only one virtual channel.

The proposed routing algorithm enjoys the low power consumption of a mesh NoC while possessing a good performance (near a torus NoC). It even exhibits better performance for light traffic because of a zero switching time between virtual channels compared to a torus NoC using two virtual channels to implement XY routing. There is a slight decrease in the performance of TRANC for heavy traffics and near the saturation point of the NoC when compared to XY routing in the trous NoC. However, as mentioned before, power consumption is a dominant factor when comparing routing algorithms in NoCs, since the network rarely works near its saturation point of operation.

2. ROUTING IN THE MESH AND TORUS NOCS

An $n \times n$ mesh or torus NoC consists of n^2 nodes arranged in a two-dimensional grid structure. Each node is addressed using an (x,y) tuple and has a neighboring node in the increasing and decreasing directions (positive and negative directions) in each dimension. The first node and the last node of each dimension are linked using a wraparound link in the torus NoC, while such a wraparound link does not exist in the mesh NoC. Fig. 1 shows a 4x4 mesh NoC and a 4x4 torus NoC. Each node in the network consists of two parts: IP and Router. Usually, the whole system (called SoC) except for the IPs is called the NoC.

2.1 Node structure in mesh and torus NoCs

A cycle accurate and synthesizable VHDL hardware model for NoCs has been implemented and several different topologies like mesh and torus have been tested based on it. The top most shared component in this hardware model is the NoC node in which *IP* and

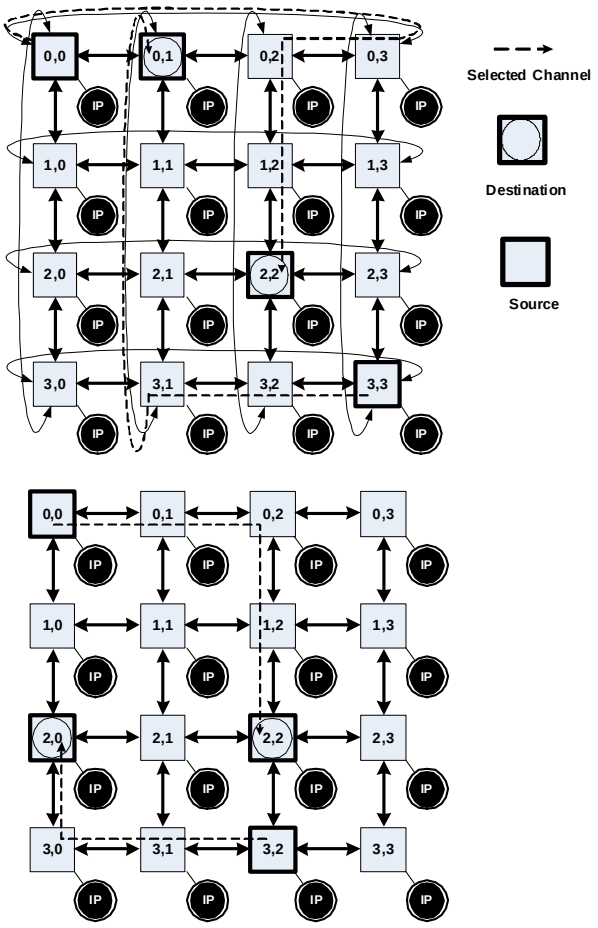

Fig. 1: A 4x4 torus NoC (Top) and a 4x4 mesh NoC (Bottom)

router are its main components. Fig. 2 shows the node structure in the implemented model.

The IP can be a processor with some local memory, or any other module that can send/receive packets over the network. In our implementation it generates packets based on a traffic model like uniform distribution for packet destinations. Also each IP generates packets on intervals based on a Poisson distribution.

The router has five input and five output channels. A node uses four inputs and four output channels to connect to its neighboring nodes; two per dimension, one in each direction. The remaining channels are used by the IP to inject/eject messages to/from the network,

respectively. Messages generated by the IP are injected into the network through the injection channel. Messages that arrive at the destination node are transferred to the IP through the ejection channel. The bandwidth of each channel is shared among a number, say *V*, of virtual channels. The hardware implementation of the router consists of several different units such as *Address Extractor* which determines and manipulates the packet headers and contains some buffer (of few flits) for each incoming virtual channel. It should be noted that the more the number of virtual channels is, the structure of the node is more complex. There are *Multiplexer* and *De-Multiplexer* units which handle the virtual channel operations, *Selector* unit which applies the virtual channel selection rule, *Crossbar switch* that can simultaneously connect multiple input channels to multiple output channels given that there is no contention over the output channels. *Reservator* unit which controls the crossbar switch and other related sub-modules. When a specific topology like mesh or torus is supposed to be modeled by such components, a top-level wrapper module is implemented that connects several nodes of this type to each other based on the structure of the specified topology. Based on this hardware model, different cases of mesh and torus topologies have been simulated and synthesized to extract accurate quantities, e.g. average message latency and power consumption values.

2.2 Routing in mesh and torus NoCs

Examples of XY-routing are also shown in Fig. 1 for torus and mesh networks (the route is indicated as dashed lines). The XY routing for mesh NoCs is straight forward: a message (or packet) first traverses its route towards its destination across X axis and then across Y axis. It is easy to see that such a routing algorithm prevent cyclic dependency in reserving and using network channels by the messages. The case is however different for the torus NoCs as where wraparound links can clearly make cyclic channel dependency resulting in deadlock situation. The straight forward deadlock free XY routing algorithm for this case needs at least two virtual channels per physical channel. In XY routing algorithm in the torus, the packet traverses first X dimension and then Y dimension (as in mesh NoCs); it uses the first virtual channel before reaching a wraparound link, thereafter it uses the second virtual channel until it reaches the destination node. Thus, XY routing in the mesh NoCs requires one virtual channel per physical channel while it requires 2 virtual channels per physical channel in torus NoCs.

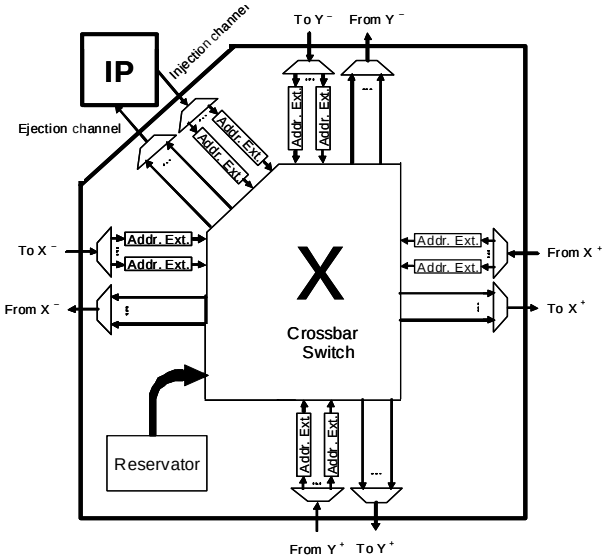

Fig. 2: Node structure in NoC

2.3 Performance and power consumption results

Fig. 3 shows the performance and power consumption of XY routing for a 4x4 mesh NoC with one and two virtual channels and a 4x4 torus NoC with two virtual channels. In the figure, horizontal axis shows the message generation rate at each node and the vertical axis shows either the average message latency (an important measure of NoC performance) or the energy consumed. The simulated topologies are of 4x4 nodes, message length of 32 flits (4 flits for the header and 28 data flits), and buffer size of 4 flits for each virtual channel. The figure shows that the torus exhibits a better performance compared to the mesh topology with one and 2 virtual channels per physical channel. This is because of the lower diameter and average inter-node distance in the torus NoC compared to its equivalent mesh network. The figure also shows that the number of virtual channels mainly determines the

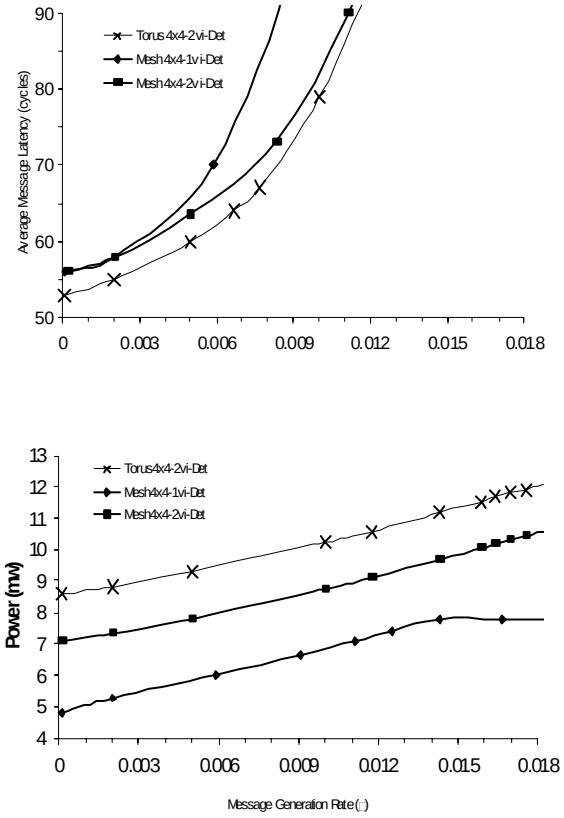

Fig. 3: Performance and power consumption of XY routing in a 4x4 mesh with one and two virtual channels and 4x4 torus with two virtual channels

power consumption of the network and the torus NoC with two virtual channels has a larger amount of dissipated power due to the complexity of the switch and higher buffering requirement and extra wraparound links.

3. INTERCONNECTION ROUTING NOTATION (IRN)

We propose a new notation to extract new routing algorithms for torus and mesh NoCs. This notation can also be extended to other interconnection topologies. By using this notation, it is possible to have better understanding and formulation of routing algorithms for interconnection networks. Consider the XY routing algorithm for the 4x4 mesh network with one virtual channel per physical channel (as shown in Fig. 1). The *IRN Map* and *IRN Graph* for this algorithm are shown in Fig. 4. In XY routing algorithm, a packet first traverses the X axis and then continues its journey towards its destination along Y axis. The IRN notation only explores the rule of movement through one axis (current axis or dimension). First row of the IRN Graph shows that if the source and destination nodes for a packet are adjacent across a dimension, then the packet moves towards the destination node directly with one step. The other rows show the direction of movements for distances more than one hop, individually for all node locations at a dimension. Corresponding to the IRN Graph, the IRN Map in each row shows the direction that the packet should traverse when it is in a specified location to cross the network to get closer to the destination. As shown in the figure, all the movements over the diameter of the map (or matrix), which means the packet should go from a smaller index to a bigger index, have a '+' sign. This sign indicates movement in the positive direction of that dimension; similarly the '-' sign is used in the lower part of the map.

Fig. 5 shows the notation for the proposed routing algorithm in the torus network but with only one virtual channel. At the first row of the IRN Graph the wraparound link is shown. Because of using wrap around links, a packet may reach its destination using positive or negative directions; but for a routing algorithm to be deadlock free only one of the directions should be selected. The plus and minus symbols with the circles refer to movements on the first row of IRN map in which it is not reasonable to select the opposite direction to reach the destination. Therefore, we suppose these moves are always

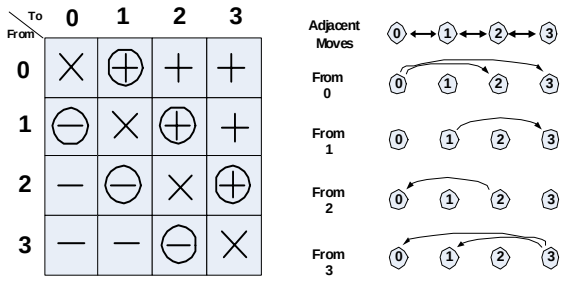

Fig. 4: The IRN Map and Graph for a 4 × 4 mesh using XY Routing

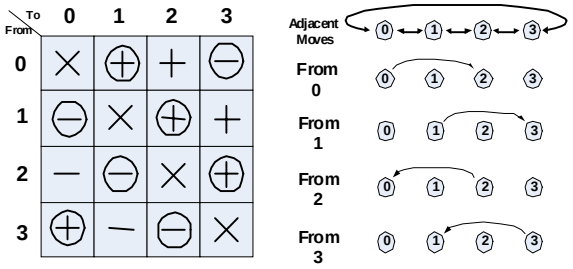

Fig. 5: The IRN Map and Graph for a sample routing in a 4 × 4 torus

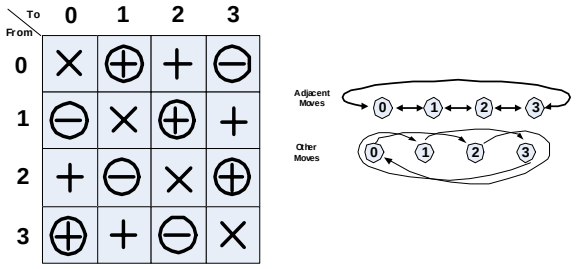

Fig. 6: A routing case in a 4 × 4 torus causing deadlock

unchangeable, and only the signs without the circle are selectable. It should be noted that there are 4 selectable moves that can make up 16 different routing algorithms some of which are deadlock free and some others are not. The goal is to find the best selection (being deadlock-free and as minimal and optimal as possible, which will be explored using minimality and optimality factors in next section). Here, the only change to the mesh XY routing is that the first and last nodes in a dimension can use the wraparound link for a one step movement. For example, in the case of 4x4 torus, nodes 0 and 3 can communicate with each other using the wraparound links.

3.1 The Rules

In order to complete the IRN Map, the following rules should be applied: In each column there should not be sign changing for more than once. This is because it may cause a livelock in the network. For the sake of minimality and optimality it is better to have equal number of '+' and '-' signs for the selectable area. The same case applies to the number of '+' and '-' in a row. Also there should not be more than one sign changes in a row. Fig. 6 shows a case for the 4x4 torus in which deadlock may occur. Deadlock is caused because of a loop between movements in positive or negative directions. There should be one row with all selectable '-' movements and also one row with all selectable '+' movements for the algorithm to be deadlock free. After filling these two rows, the other rows should be filled using the previous rules. Example for applying these rules is shown in Fig. 7 for the 6x6 torus NoC.

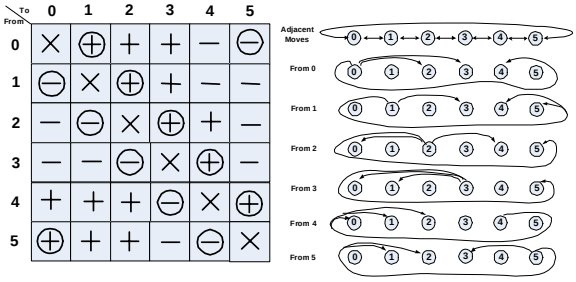

Fig. 7: IRN Map and IRN Graph for routing in a 6 × 6 torus

3.2 Optimality and Minimality

Minimality Factor (*M*): For a packet which traverses the network from the source to destination node, there is always a minimum number that determines the shortest path for the packet. Because of the limitations that the routing algorithm poses on the packet, the routing algorithm might not be always minimal.

As can be seen in Fig. 8, the selectable area of the IRN Map determines whether the proposed routing algorithm for a dimension is minimal or not. The fact is that for the dimensions with radix $n>4$ in the torus, there is not a minimal algorithm in which all the paths are the shortest possible ones. The shaded boxes in the figure show a subset of the selectable areas for odd and even values of n in which the minimality parameter is applicable.

A number is displayed in the top corner of the shaded boxes: it is *0* if the packet takes a direction with the shortest distance and other numbers show the extra steps that should be taken. When n is odd, all movements of the selectable area are shaded and when n is even this parameter is not applicable to the

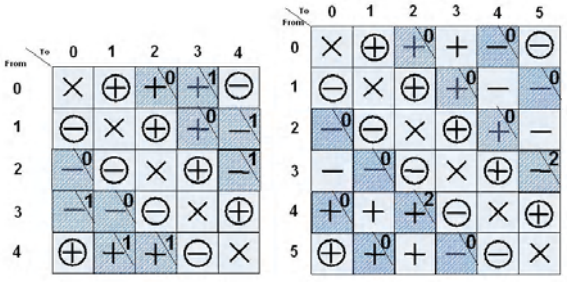

Fig. 8: Minimality factor for some routing methods in a 5 × 5 torus (left) and a 6 × 6 torus (right)

movements with distance *n*/2 from the source nodes, since both directions result in equal distance. At last for a specific dimension if we calculate the sum of all minimal path lengths when comparing different routing algorithms, the algorithm with the minimum total sum will be the best one (or here called the minimal one).

Optimality Factor (*Opt*): Although in some references in the area of interconnection networks, an optimal algorithm is known to be a *balanced* algorithm, here we propose a quantitative approach as a parameter based on IRN notation to measure the quality of traffic balance or *optimality factor*. This parameter describes how good an algorithm can balance the network traffic. In fact for a routing algorithm we need a measure that indicates if all the links are utilized properly with respect to the packet destination address distribution. For the case of uniform traffic, the links should be utilized equally. With uniform traffic pattern, it is supposed that all nodes send a packet to any other network nodes with equal chance, and therefore all the links should be utilized evenly. As shown in Fig. 9, some numbers have been presented on adjacent movements with the shaded boxes. Note that these movements are the representatives of their corresponding links and if we consider the sum of '+' ('-') signs for positive (negative) movements in their corresponding rows and columns (considering wraparounds), the result shows the number of times this link has been utilized. The numbers have been shown in lower corner of the shaded boxes and as discussed they should have the same value in order to have optimal routing; therefore the variance (*Opt*) of all the numbers is a good

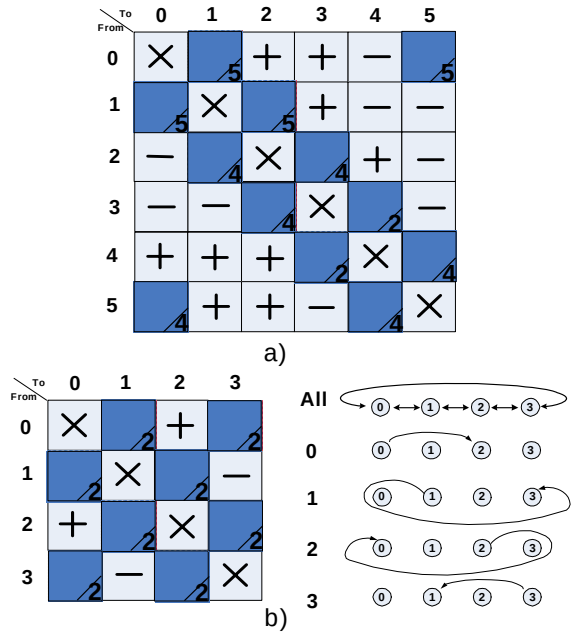

Fig. 9: IRN with optimality factors for different routing methods; (a) Non-optimal in 6 × 6 torus, *Opt* = 12 and (*b*) Optimal in a 4 × 4 torus, *Opt* = 0

candidate to represent the optimality of routing algorithms: the smaller the *Opt* is, the more optimal the algorithm is. As shown in the figure, two different algorithms have been proposed for the 4x4 torus in which one of the algorithms is optimal since all the optimality numbers are equal to 2, therefore *Opt* is *0*. The other algorithm is not optimal, although both algorithms are minimal.

4. THE TRANC ROUTING ALGORITHM

In this section, we introduce a new routing algorithm that is deadlock free and requires only one virtual channel per physical channel (i.e. no extra virtual channel to the existing physical channel is required). The algorithm uses an incremental approach based on the IRN notation in an *n*×*n* torus NoC.

4.1 Proposed Algorithm for any radix n

As shown in Fig. 10, the IRN Map for radix *n* is generated by adding a row and a column to the IRN Map of radix *n*-1, starting from *n*=4. The algorithm is straight forward for the 4x4 torus; for higher radices

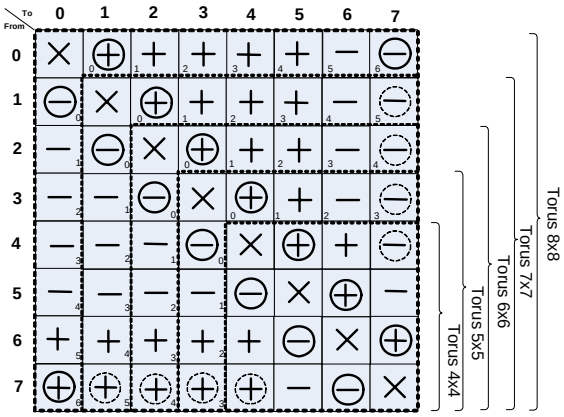

Fig. 10: The proposed IRN Map representing the TRANC routing algorithm in an $n \times n$ torus.

it is enough to add a row and a column as shown in Fig. 10. In order to complete the new row it is enough to fill the right most two boxes with '-' signs and others with '+' signs. Again for completing the new column it is enough to fill the two lower boxes with '+' signs and others with '-' signs. When two or more of the moves described in IRN Graph happen simultaneously, they may form a situation where some of the packets are waiting for other ones to free the path. In this situation, there is a packet contention. When the contention starts from a packet and lasts with the same packet, such that no activity is possible for the packets, it is said that a deadlock situation has occurred. A routing algorithm that never causes a deadlock situation is called a deadlock free routing algorithm.

The TRANC algorithm is a deadlock free routing algorithm. The intuitive justification that it is deadlock free is extracted from the IRN Map and Graph in Fig. 7. As discussed before, there is not a positive movement of more than one step from node 3, and therefore the positive loop is broken in the network. The same is correct for negative movements, as there are not any negative movements of more than one step from node 4 to other nodes and therefore the negative loops also are broken. There is mathematical justification that TRANC is deadlock free, that we do not present it here. Furthermore the justification is also supported by the extensive simulation experiments we have realized for different scenarios. As discussed before for the dimensions with radix $n>4$, TRANC is not fully optimal and fully minimal but with good optimality and minimality factors for different radices. The reason is that each packet traverses the shortest possible path to reach the destination which ignores deadlock, not the shortest physical path which potentially causes a deadlock. Also the usage of wraparound links is not balanced compared to other links because of the rules that have been applied to the algorithm. It is possible to use a different approach for different radices based on IRN that may result in better optimality and minimality factors but justification of deadlock freedom for each radix should be done separately. We ignore this approach for the sake of present discussion.

In Fig. 12, a pseudo code for TRANC algorithm is given. As can be seen in the code, the complexity of the hardware that utilizes this algorithm compared to the classic XY routing algorithm includes the extra comparisons that should be done with n-1, n-2 and n-

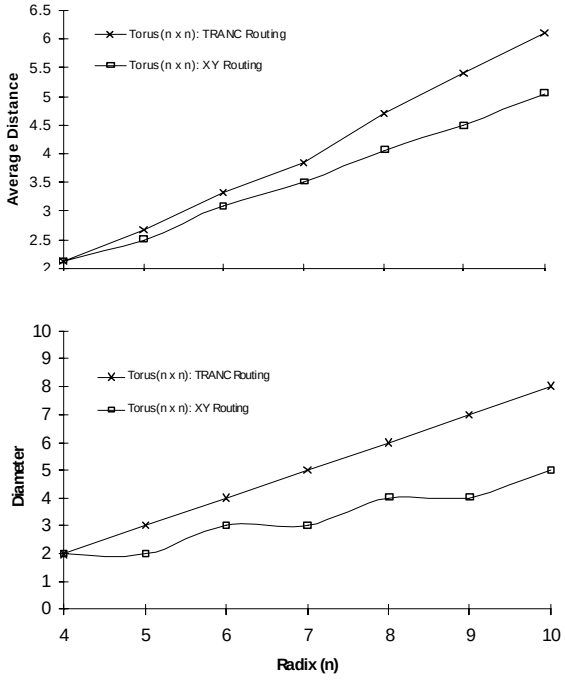

Fig. 11: The average inter-node distance and diameter using TRANC and XY routing algorithms

```
Algorithm TRANC for 2-Dimensional Torus NoCs.
    Inputs: Coordinates of current node (Xcurrent, Ycurrent),
            destination node (Xdest, Ydest), and radix n;
    Output: Selected output Channel
Begin
Xoffset := Xdest - Xcurrent;   Yoffset :=Ydest - Ycurrent;
if (Xoffset=0) and (Yoffset !=0) then return Ejection Channel;
else
  { if (Xoffset =1) or (Xoffset = -n+1) or
       (((Xdest = n-2) and (Xcurrent=n-4)) or (Xcurrent=n-2)) or
       (Xdest -2>= Xcurrent)
    then return X+;
    if (Xoffset = -1) or (Xoffset = n-1) or
       (((Xdest = n-3) and (Xcurrent=n-1)) or (Xcurrent=n-3)) or
       ((Xcurrent-2>= Xdest) or (Xdest =n-2))
    then return X-;
    if (Yoffset =1) or (Yoffset = -n+1) or
       (((Ydest = n-2) and (Ycurrent=n-4)) or (Ycurrent=n-2)) or
       (Ydest -2>= Ycurrent)
    then return Y+;
    if (Yoffset = -1) or (Yoffset = n-1) or
       (((Ydest = n-3) and (Ycurrent=n-1)) or (Ycurrent=n-3)) or
       ((Ycurrent-2>= Ydest) or (Ydest =n-2))
    then return Y-;
  }
End.
```

Fig. 12: Pseudo code for TRANC routing algorithm

3 and since *n* is a constant number (when a fixed radix is implemented in hardware), only some comparison operations with some constant numbers are added to the code. Such simple comparisons do not require considerable power and do not impose noticeable delay in routing as will be shown later in the simulation results.

5. SIMULATION RESULTS

We have implemented a VHDL cycle accurate and synthesizable hardware model for mesh and torus NoCs using both XY routing and TRANC with the possibility of using different number of virtual channels. To evaluate the performance and power dissipation for the proposed routing algorithm in comparison to XY routing two different network sizes (4x4 and 6x6 nodes) are considered. The message size is considered to be fixed and equal to 32 flits (or phits) and the destination of the messages is chosen uniformly over the network nodes. Messages are generated and entered into the network following a Poisson distribution. The VHDL implementation has been used for both performance evaluation using simulation tools and also power estimation using *Power Compiler* CAD tool [7,12, 13].

A primarily evaluation of the TRANC using a simple C++ program shows that TRANC slightly increases the maximum and the average inter-node distance in the network (or message path length) compared to XY routing in the torus NoC. This is shown in Fig. 11 for different network radices. Note that for popular and current network sizes used in practice today (i.e. NoC with up to 6x6 nodes) the difference between the average and maximum inter-node distances for the two routing algorithms in torus NoCs is small. Therefore, the lower complexity of the router in TRANC (due to the fewer virtual channels used) can improve the performance and reduce the power dissipation in the network.

Fig. 13 shows the simulation results for XY routing in mesh and torus NoCs and for TRANC routing algorithm in torus NoCs (for 4x4 and 6x6 wormhole-switched networks). The horizontal axis shows the traffic generation rate at each node while the vertical axis shows the average message latency (or dissipated power) in the network. As can be seen in the figure, the performance of TRANC routing (with one virtual channel) is slightly better than XY routing (using 2 virtual channels) while the power consumption is near that of a mesh NoC and much less than that of XY-routed torus (using 2 virtual channels). To have a unique measure to assess the suitability of the proposed algorithm for torus NoCs we have also used the product of average message latency and power consumption. Simulation results show that the proposed routing algorithm for the torus NoC using one virtual channel is superior to the equivalent mesh using XY routing (using one virtual channel) and equivalent torus NoC using XY routing with 2 virtual channels.

6. CONCLUSIONS

Current SoC designs have popularly employed point-to-point NoCs for inter-IP communication. The most popular NoCs are the mesh and torus networks. The mesh topology enjoys its simple structure and possibility of using XY routing with only one virtual channel. However, when wraparound links are used, in the torus NoCs, Two virtual channels should be used to ensure deadlock freedom for XY routing. On

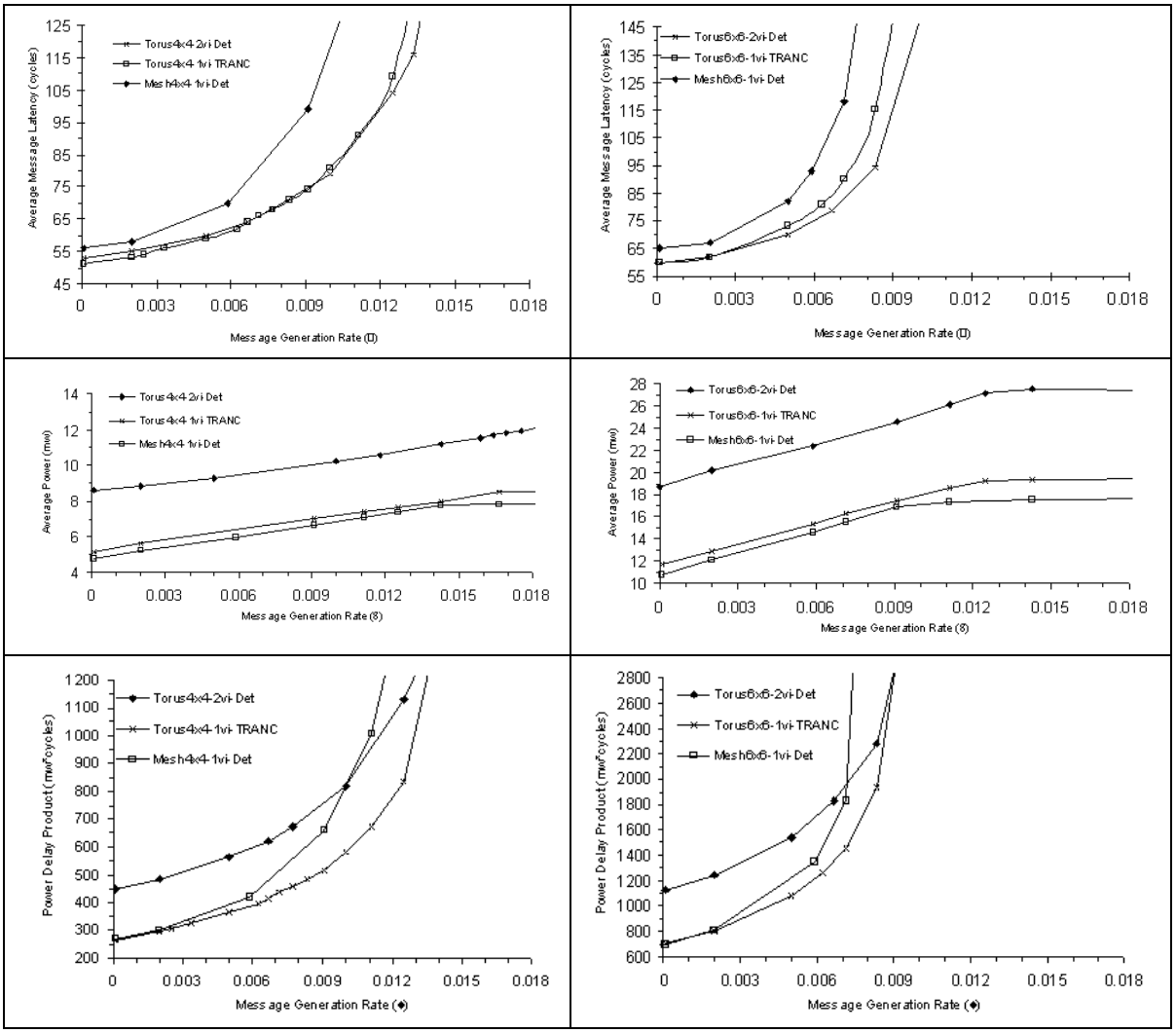

Fig. 13: Performance, power consumption, and power-delay product of XY routing in the mesh and torus NoCs and TRANC routing algorithm in the torus NoC with radices 4 and 6

the other hand, adding virtual channels increases power dissipation, although performance is increased (compared to the mesh NoC) as a result of lower inter-IP distance caused by wraparound links in the torus.In this paper, a new network routing notation, IRN, and based on it a new routing algorithm for the torus NoCs (called TRANC) were presented. The TRANC routing algorithm for the torus NoC uses only one virtual channel and thus consumes lower energy compared to XY routing that requires 2 virtual channels. Note that simplicity of the router (less buffering and switching hardware complexity) can well compensate for the slightly increased inter-IP distance (compared to the XY routed torus NoC with 2 virtual channels) and result in a slightly higher performance. Our next objective is to design partially and fully-adaptive routing algorithms for torus NoCs with a minimum virtual channel requirement.

REFERENCES

1. W. J. Dally, B. Towles, “Route packets, not wires: on-chip interconnection networks”, Proc. DAC, pp. 684–689, June 2001.

2. R. V. Boppana, S. Chalasani, “A framework for designing deadlock-free wormhole routing algorithms”, IEEE Transactions on Parallel and Distributed Systems (TPDS), 7(2): 169-183, 1996.
3. C. J. Glass, L. M. Ni, “The turn model for adaptive routing”, Proceedings of the International Symposium on Computer Architecture (ISCA), pp. 278-287, 1992.
4. A. Singh, W. J. Dally, A. K. Gupta, B. Towles, “GOAL: A Load-balanced Adaptive Routing Algorithm for Torus Networks”, in Proc. of the International Symp. on Comp. Arch., pp. 194-205, June, 2003.
5. Terry T. Ye, Luca Benini, Giovanni De Micheli, ”Analysis of Power Consumption on Switch Fabrics in Network Routers”, In Proceedings of DAC, 2002.
6. H-S Wang, L-S Peh, S. Malik, ”Orion: A Power-Performance Simulator for Interconnection Network”, In International Symposium on Microarchitecture, Istanbul, Turkey, November 2002.
7. D. L. Liu, C. Svensson, “Power consumption estimation in CMOS VLSI chips”, IEEE Journal of Solid-State Circuits, 29(6): 663-670, 1994.
8. W. J. Dally, H. Aoki, “Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels”, IEEE ransactions on Parallel and Distributed Systems, 4(4):466–475, April 1993.
9. Jae H. Kim, Ziqiang Liu, Andrew A. Chien, “Compressionless Routing: A Framework for Adaptive and Fault-tolerant Routing”, IEEE Transactions on Parallel and Distributed Systems 1996.
10. W.J. Dally, C.L. Seitz, “Deadlock-free message routing in multiprocessor interconnection networks”, IEEE Transactions on Computers, vol. C-36, no. 5, pp. 547-553, May 1987.
11. W. J. Dally, C. Seitz, “The torus routing chip”, In Distributed Computing, pages 187 196, 1986.
12. N. Banerjee, P. Vellanki, K. S. Chatha, “A Power and Performance Model for Network-on-Chip Architectures”, In Proceedings of DATE, Paris, France, February 2004.
13. K. Srinivasan, K. S. Chatha, “ISIS : A Genetic Algorithm based Technique for Custom On-Chip Interconnection Network Synthesis”, In Proceedings of the 18th International Conference on VLSI Design (VLSID’05)