An Efficient History-Based Routing Algorithm for Interconnection Networks

Sanaz Rahimi Moosavi[1], Chia-Yuan Chang[1], Amir-Mohammad Rahmani[1,2], Juha Plosila[1], Ka Lok Man[3,4,5], Taikyeong T. Jeong[4], Eng Gee Lim[3]

[1]Department of Information Technology, University of Turku, Finland
[2]Turku Centre for Computer Science (TUCS), Finland [3]Xi'an Jiaotong-Liverpool University, China
[4]Myongji University, South Korea [5]Baltic Institute of Advanced Technology, Lithuania

Email: {saramo, chyuch, amirah, juplos}@utu.fi, {ka.man, enggee}@xjtlu.edu.cn, ttjeong@mju.ac.kr

Abstract— **Network-on-chip (NoC) approach has been proposed as a solution to the complex on-chip communication problems by scaling down the concepts of macro- and tele-networks, and applying them to the system-on-chip domain. In this paper, an efficient routing algorithm for two-dimensional mesh network-on-chips is presented. The algorithm, which is based on Odd-Even turn model, is called History-Based Odd-Even (HB-OE). It is more fair and efficient in load balancing compared to the typical Odd-Even turn model algorithm. In this routing, based on the location of the current node, the network is divided into four sub-networks and the history of each sub-network regarding the direction of the last forwarded packet is saved using a flag register. We further enhance this routing by using a technique named Free-Channel to check the availability of the output ports as well as their history. To assess the latency of the proposed algorithm, transpose traffic profile for packet injection is used. The simulation results reveal that the HB-OE + Free-Channel routing policy can achieve lower latency compared to the conventional Odd-Even turn model with negligible area overhead.**

Keywords- Network-on-chip, Routing algorithm, Odd-Even turn model, Low latency routing

I. INTRODUCTION

Following the Moore's law, on-chip transistor densities have increased steadily enabling the integration of dozens of components on a single die. These components include regular arrays of processors and cache banks in tiled chip multiprocessors (CMPs) and heterogeneous resources in system-on-chip (SoC) designs [1][2]. One outcome of greater integration is that interconnection networks begin to replace shared buses and other forms of communication featuring long global wires. Networks-on-chip (NoCs) scale better than traditional forms of on-chip interconnect and have better performance and fault tolerance characteristics [3]. Many research groups have devoted their efforts to different aspects of NoC's, including topology, routing algorithms and architectures, power management, and core mapping (see, e.g., [2][4][5]).

In NoCs, routing algorithms are used to determine the path of a packet from the source to the destination. These algorithms are classified as deterministic and adaptive. In deterministic routing the same path is chosen every time a packet must be routed between a particular source and destination. It is suitable for on-chip networks where network traffic is predictable and relatively simple since the current state of the network in terms of traffic load, faulty link, temperature is not considered. Static routings provide in-order-delivery of packets and low-cost router implementation for NoC architectures. In contrast, adaptive routing, as the name suggests, uses information about the network state to adaptively discover routes in case of path changes as traffic conditions and requirements of the application change. The provided adaptivity results in more efficient traffic distribution in a network at the cost of additional run-time monitoring and management logic. This routing is desirable for networks in which traffic conditions show irregularity and unpredictability [6].

There are a number of routing algorithms which we briefly review those related to the algorithm proposed in this work. In [7], a static routing algorithm for two-dimensional meshes which is called XY is introduced. In this routing algorithm, each packet first travels along the X and then the Y direction to reach the destination. For this method, deadlock never occurs but no adaptivity exists in this algorithm. An adaptive routing algorithm named turn-model is introduced in [8] based on which another adaptive routing algorithm called Odd-Even turn model is proposed in [9]. To avoid deadlock, Odd–Even method restricts the position that turns are allowed in the mesh topology. Another algorithm called DyAD is introduced in [10]. This algorithm is a combination of a static routing algorithm called oe-fix, and an adaptive routing algorithm based on the Odd-Even turn algorithm. Depending on the congestion condition of the network, one of the routing algorithms is invoked.

In this paper, we propose an Odd-Even turn model based routing algorithm called History-Based Odd-Even (HB-OE) to minimize the network latency and enhance the load distribution across the network. In addition, we reinforce the proposed algorithm by a technique named Free-Channel to consider the status of the output ports while making the routing decision.

The remainder of the paper is organized as follows. Section II elaborates the demands for enhancement of the typical Odd-Even turn model routing and motivates this work. Section III presents the proposed History-Based Odd-Even turn model routing which supports channel-free technique. The simulation methodology and results are shown in Section IV and finally, Section V concludes this paper.

II. BACKGROUND AND MOTIVATION

As discussed in Section I, Odd-Even turn model is an algorithm used for our mesh structure of NoC. This model already limits few certain critical turns so no deadlock can ever happen as a message traverses. In order to avoid deadlock, virtual channels (VC) can be added into routers when network is active [6]. Adding VC is surely a solution to solve the deadlock issue of NoC but the cost of it has also increased and should be taken into account. Virtual channel needs to be implemented by increasing the size of existing buffer capability in each PE, and it requires significant amount of complex algorithm to realize it.

XY algorithm prevents the deadlock by strictly obeying the rules of X axis execution first then Y axis routing follows afterwards, but the cost of such a method is also considerable regarding to traffic efficiency. Picturing a 2D mesh system with 3×3 NoC when insertion packet rate is high, the central switch will be a communication hotspot with a great number of packets waiting to be transmitted; and in result it will stop other switches to be transferring packet flits. However, The way how odd-even model works is as packet data is propagation in a mesh system, when the current node is in odd column, then no North-West (NW) and South-West (SW) turns can be made. Likewise, when current node is in even column, East-North (EN) and East-South (ES) turns will not occur. Based on such rules, Odd-Even turn model can successfully send data to its target by checking how current node address is in comparison with target node without having deadlock. Fig .1 shows examples of Odd-Even turn model routing for three source-destinations pairs. In this figure, S and D prefixes indicate source and destination, respectively.

Two strict criteria for odd-even turn model:

- At even column: No turns of East → North, East → South are allowed
- At odd column : No turns of North → West, South → West are allowed

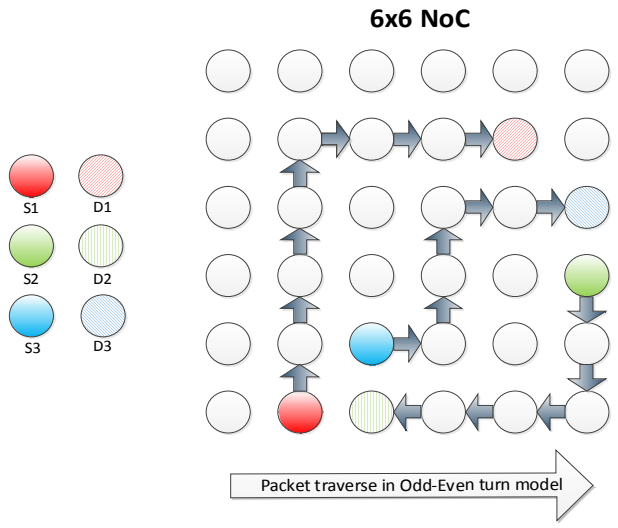

Fig. 1. Paths of Odd-Even model

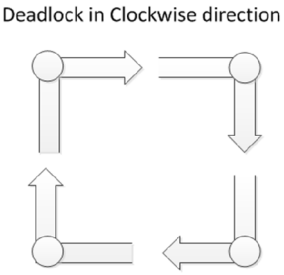

Fig. 2. Scenario of deadlock flow

Deadlock is caused by a cyclic loop involved with several routers in system, in which packets are all waiting for the next router to release the channel, but those participated routers will keep on waiting one after another and gradually the deadlock occurred as shown in Fig. 2.

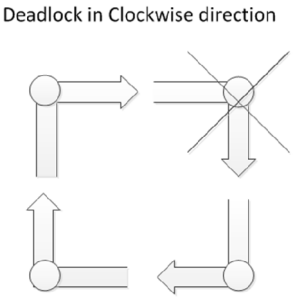

Fig. 3. Example of deadlock-free flow

In a deadlock-free routing algorithm, at least one of the turns in any possible cyclic-dependency loop must be avoided. Fig. 3 shows an example of a deadlock-free cyclic pattern.

III. HISTORY-BASED ODD-EVEN TURN MODEL

Stemmed from regular Odd-Even turn model, History-Based Odd-Even turn model features flag mechanism acting as a control to evenly spread incoming packets to the NoC. History-Based Odd-Even model is equipped with a flag control inside switch control; the flag control indicates the router to dispatch packets to different routes to the same domain over time.

A. The purpose of flag mechanism in History-Based Odd-Even turn model

The typical Odd-even turn model only sends the packets to certain directions on available columns which are not conflicting with restrictions; this limits the possibility to ease the traffic by sending the packets to alternative routes. History-Based Odd-Even turn model uses the flag to record which port the last packet was being transmitted, and if next packet will go to the same domain destination address where the previous packet went, this time the switch control will flag record and link the alternative port to send the message. The purpose of such a flag switching mechanism is to even dispatch the packets over the system, so in long term it is possible to balance the potential communication hotspots.

Even though routers in the system can manipulate the routes of where the packets should proceed, but they still have

to follow the rules of Odd-Even turn model. It means when current routers are on even columns, no EN and ES turns can be taken; as well as no NW and SW turns on odd columns.

B. History-Based Odd-Even turn model algorithm

The algorithm of History-Based Odd-Even turn model is almost identically as combination the OE model. The main difference between History-Based Odd-Even turn model and Odd-Even turn model is that the History-Based model uses flag register in switch control to decide which alternative pattern of port it should be assigned to packets. The detailed algorithm of the History-Based Odd-Even turn model is shown in Fig. 4. The list of abbreviations and their meanings are as follows:

- *lx* : current node in X-axis
- *tx*: destination node in X-axis
- *ly* : current node in Y-axis
- *ty* : destination node in Y-axis
- *domain_1_east* : store previous routes in domain 1 (with initial value of '1')
- *domain_2_east* : store previous routes in domain 2 (with initial value of '1')
- *domain_3_west* : store previous routes in domain 3 (with initial value of '1')
- *domain_4_west* : store previous routes in domain 4 (with initial value of '1')

Fig. 5 shows how the patterns of packets will be passing in east bond direction. Notice that at even columns of 0, 2, and 4, there is only one direction doable based on Odd-Even turn model. However, when on odd columns of 1, 3 and 5, there are always two options depending on previous and current flag contents.

C. History-Based Odd-Even turn model supporting channel-free technique

Based on flag switching idea, channel-free technique gives a router more flexibility to resolve the traffic conditions in the system. Previously it was introduced that History-Based model evenly sends packets to two different directions of North and East; but once East or North channel permission is not released yet, the router will keep waiting until it is free then send the packet. History-Based Odd-Even turn model supporting channel-free technique instead checks if the other path channel is free even though this path is not supposed to take based on current flag control. For example, last time a router has sent a packet to north, so this time the same router will send the packet to east. But if East path is not ready, the router will try to send the packet to North again even though the flag is set to East.

When a certain large amount of packets with high injection rate are injected into the system, a number of particular locations known as communication hot-spots (CHS) would appear and exacerbate the performance. This is due to the mechanism of flag switching once makes a requesting of taking over the free channel, a node have to wait for release of channel path to be confirmed with, and unless this path has been taken in the future otherwise the node will keep on waiting until it is set free.

```
If (lx=tx)   ;/*target destination is local*/
        send packet to local port;
elsif (lx=tx and ly/=ty)   ;/*target destination is in Y-axis*/
        send packet to Y-axis;
elsif (tx>lx and ly=ty)   ;/*target destination is in X-axis*/
        send packet to X-axis;
-----------East direction-----------
elsif (tx>lx) and (ly=ty) and (lx=odd columns or lx=source node)
        if (tx-lx=1) and (tx=even columns)   ;/*only choice for
going North to prevent EN deadlock*/
                send packet to North port;
-----------Start of domain 1-----------
        elsif (ty>ly) and domain_1_East='1')   ;/*North
neighboring buffer is available*/
                send packet to North port;
        elsif (ty>ly) and domain_1_East='0')   ;/*East neighboring
buffer is available*/
                send packet to East port;
-----------End of domain 1-----------
-----------Start of domain 2-----------
        elsif (ty<ly) and domain_2_East='1')   ;/*South
neighboring buffer is available*/
                send packet to South port;
        elsif (ty>ly) and domain_2_East='0')   ;/*East neighboring
buffer is available*/
                send packet to East port;
-----------End of domain 2-----------
-----------End of East direction-----------
-----------West direction-----------
-----------Start of domain 4-----------
elsif (tx<lx) and (ly/=ty) and (lx=even columns)
        if (ty>ly) and (domain_4_West='1')   ;/*West neighboring
buffer is available*/
                send packet to West port;
        elsif (ty>ly) and (domain_4_West='0')   ;/*North
neighboring buffer is available*/
                send packet to North port;
-----------End of domain 4-----------
-----------Start of domain 3-----------
        elsif (ty>ly) and (domain_3_West='1')   ;/*West
neighboring buffer is available*/
                send packet to West port;
        elsif (ty>ly) and (domain_3_West='0')   ;/*South
neighboring buffer is available*/
                send packet to South port;
-----------End of domain 3-----------
-----------End of West direction-----------
```

Fig. 4. History-Based Odd-Even turn model algorithm

The History-Based Odd-Even turn model can evenly dispense the data and pass onto the system in general, but as long as the requesting channel is busy, the system will be put on hold when certain channels of traffic are completed. Therefore, we introduced another modified model similar to it called History-Based Odd-Even turn model + channel-free technique. This model performs as even though a node is waiting for a free channel to use dependent on flag setting, but if the requesting channel is being occupied, and possibly another free channel to use based on the Odd-Even turn model regulation, the arbitrator of a node will still take the previously route link to dispatch the packet neglecting what flag indicates. In other words, the decision of where to send packet lies on

channel availability as first priority, then flag indication comes as secondary choice.

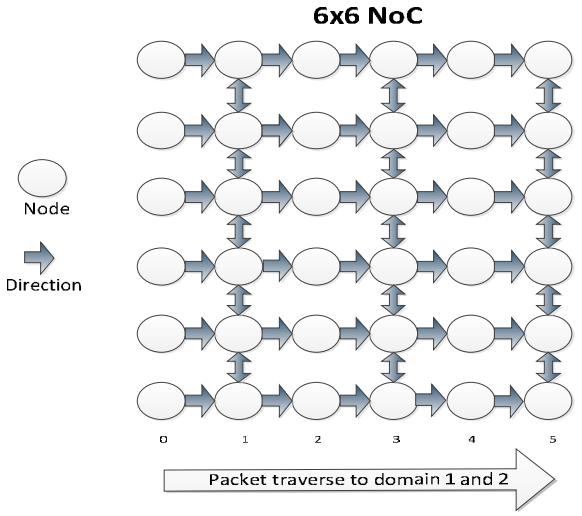

Fig. 5. Possible route links of History-Based Odd-Even turn model in a 6×6 NoC

IV. Experimental Results

To assess the efficiency of the proposed routing algorithm, simulations for synthetic traffics were conducted. We have simulated the Typical Odd-Even Turn Model and the proposed History-Based Odd-Even turn model supporting channel-free technique to characterize their latency. To demonstrate the better performance characteristics of the proposed routing algorithm, a cycle-accurate NoC simulation environment was implemented in VHDL. Wormhole packet switching is adopted. The implemented NoC consisted of typical state-of-the-art routers including input buffers, centralized control logic, with a round-robin arbiter and Odd-Even based routing, credit based flow control, internal crossbar to connect input to output ports and priority-based output scheduling. Owing to utilizing Odd-Even based algorithm, deadlock avoidance is guaranteed.

In synthetic traffic analysis, the NoC of our simulation environment consists of 6×6 nodes connected as a 2D mesh. The performance of the network was evaluated using latency curves as a function of the packet injection rate (i.e., the number of packets injected into the network per cycle). The packet latency was defined as the time duration from when the first flit is created at the source node to when the last flit is delivered to the destination node. For each simulation, the packet latencies were averaged over 3,000 packets. It was assumed that the buffer size was eight fits, and the data width was set to 32 bits. To perform the simulations, we used the proposed and typical Odd-Even routing algorithm under transpose traffic pattern [11]. In the transpose traffic pattern, each node (i , j) communicates only with node (j , i).

The average packet latency (APL) curves for transpose traffic patterns with varying packet injection rates are shown in Fig. 6. It can be observed that the network with proposed routing algorithm saturates at higher injection rates. The reason is that, by using the History-based mechanism and the free-channel technique, the proposed routing algorithm balances the load across the network.

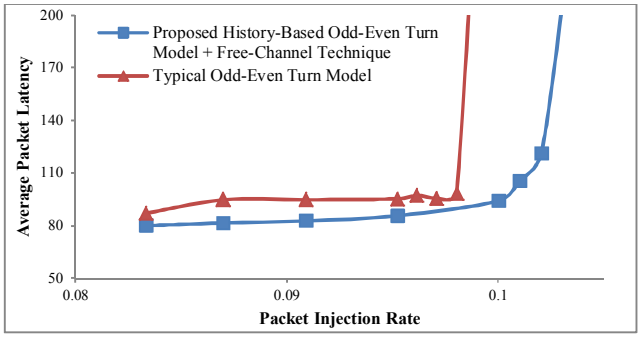

Fig. 6. Latency versus average packet injection rate on a 6×6 mesh for the proposed and typical routing algorithms under transpose traffic profile

V. Conclusions and Future Work

In this paper, a History-based Odd-Even turn model routing was presented to enhance latency and resource utilization of the networks-on-chip. The proposed routing algorithm utilizes stored information of the last forwarded packets to evenly distribute the network traffic. This routing was further enhanced by proposing a technique which monitors status of the output channels to improve the network throughput. The results showed lower latencies for the proposed routing algorithm for high packet injection rates at places where the congestion in the NoC occurs. In the future, our work will be extended by conducting a comprehensive simulation to estimate the system power and measure NoC performance under realistic traces as well as more synthetic profiles.

References

[1] L. Benini and J. De Micheli "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, 2002, pp. 70–78.

[2] A. Jantsch and H. Tenhunen (Eds.), Networks on Chip, Kluwer, 2003.

[3] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proc. of the International Conference on Design Automation*, 2001, pp. 684–689.

[4] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys* , vol. 38, no. 1, 2006, pp. 1-51.

[5] P. Lotfi-Kamran, A.-M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY - A low cost congestion-aware routing algorithm for network-on-chips," *Journal of System Architure*, vol. 56, no. 7, 2010, pp. 256-264.

[6] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, 1993, pp. 62–76.

[7] Intel Corporation, A touchstone delta system description, in *Intel Advanced Information*, 1991.

[8] C.J. Glass and L.M. Ni, "The turn model for adaptive routing," *Journal of the ACM*, vol. 41, 1994, pp. 874–902.

[9] G.M. Chiu, "The odd–even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 11, no. 7, 2000, pp. 729-738.

[10] J.C. Hu and R. Marculescu, "DyAD- smart routing for networks-on-chip," in *Proc. of the Design Automation Conference*, 2004, pp. 260–26.

[11] A.-M. Rahmani, A. Afzali-Kusha, and M. Pedram, "NED: A novel synthetic traffic pattern for power/performance analysis of network-on-chips using negative exponential distribution," *Journal of Low Power Electronics*, vol.5, no. 3, pp. 1-10, 2009.