A Novel Deadlock-Free Shortest-Path Dimension Order Routing Algorithm for Mesh-of-Tree Based Network-on-Chip Architecture*

Kanchan Manna[1], Santanu Chattopadhyay[2], and Indranil Sen Gupta[1]

[1] School of Information Technology
[2] Dept. of Electronics and Elec. Comm. Engg., Indian Institute of Technology Kharagpur, India - 721 302
{kanchanm@sit,santanu@ece,isg@cse}.iitkgp.ernet.in

Abstract. This paper presents a new dimension ordered routing algorithm for Mesh-of-Tree (MoT) based Network-on-Chip (NoC) designs. The algorithm has been theoretically proved to be deadlock, live-lock and starvation free. It also ensures shortest-path routing for the packets. The simplified algorithm, compared to the previously published works, provides same throughput and average latency measures, at a lesser hardware overhead (about 61% for routers, 46% for links, and 44% in total) due to possible reduction in the minimum flit-size. It allows us to vary router complexity more flexibly while planning the MoT based NoC for application specific System-on-Chip (SoC) synthesis.

Keywords: MoT, NoC, Interconnection Architecture, Routing.

1 Introduction

Network-on-Chip (NoC) has evolved as a viable alternative for on-chip communication. In this, a pre-designed network-fabric consisting of routers and links connecting them in a definite topology is utilized. The intellectual property (IP) cores are connected to the routers. The cores exchange messages (instead of wires) between themselves through this on-chip network fabric. Network interface (NI) module is put as interface between an IP core and the corresponding router.

Designing a routing algorithm is the most crucial part in NoC architecture design. A good routing algorithm should provide three features: low communication latency, high network throughput, and ease of implementation in hardware [5]. In this paper we have proposed a novel Dimension Order Routing (DOR) algorithm that enables us to reduce latency, power, and area requirements for the NoC [1]. DOR is based on ordering the dimensions in a network and routes the packets strictly in the same order [5]. Applied to two-dimensional Mesh-of-Tree (MoT) [3], it produces *xy routing algorithm*; route a packet first along the x dimension

* This work is partially supported by Department of Science and Technology, Govt. of India (SR/S3/EECE/0012/2009), Dt 20^{th} May, 2009.

N. Meghanathan et al. (Eds.): CCSIT 2011, Part I, CCIS 131, pp. 168–178, 2011.
© Springer-Verlag Berlin Heidelberg 2011

(row) and then along the y dimension (column). We also show that the algorithm is free from *Deadlock*, *Starvation*, and *Live-lock* [5]. The algorithm ensures simple addressing scheme, simpler routing algorithm, and reduced area overhead compared to [2, 3, 4]. The paper organized as follows. Section 2 describes earlier works and highlights our contribution. In Section 3, we have discussed about addressing scheme. The new algorithm is presented in Section 4. In Section 5, we have discussed about guaranteed shortest path. Section 6 proves that the algorithm is free from deadlock, starvation, and live-lock. In Section 7 we have discussed about metrics for performance evaluation. Simulation methodology has been discussed in Section 8. Simulation result is described in Section 9. Finally in Section 10, we conclude the work.

2 Related Works and Our Contributions

A good NoC architecture should have the following three features: large bisection width (for parallel communication), small diameter (for reduced communication delay), and lower node degree (lesser metal layers) [3]. There are several kinds of interconnection topologies reported in the literature. They include Cliche (Mesh), Torus, Folded Torus, Binary Tree, Octagon, Scalable Programmable Integrated Network (SPIN), Butterfly Fat Tree (BFT) and Mesh-of-Tree (MoT) [6] - each has its own advantages and disadvantages [2]. A hybrid of mesh and tree topologies is Mesh-of-Tree (MoT). It was first proposed by Balkan et al [6] for NoC architecture to communicate from processors (sources) to memory modules (destinations). The trees are categorized as row tree or column tree. They place the processors and memories at the root of row trees and column trees respectively. It thus requires a large number of routers.

Kundu and Chattopadhyay[2, 3, 4] proposed a modified MoT configuration (Fig. 1(b)). In their configuration MoT has three types of routers – root, stem, and leaf. IP cores are connected to leaf routers only. In this connection, a 4x4 configuration with 32 cores, need a total of 40 routers, out which 16 are leaf routers, 16 stem routers, and remaining 8 are root routers. Each leaf has 2 cores. For a 4x4 MoT, a total 13 bits have been utilized for identifying a source or destination core. This has forced a minimum flit length to be 32 bits - 13 bits for destination, 13 bits for source, 1 bit each for end of packet (EOP) and begin of packet (BOP), 2 bits for virtual channel (VC), 1 bit for message type (e.g. request-response type of message) and 1 bit for identifying quality-of-service priority. Naturally the buffers within routers are 32 bit wide. Our work presents an improved routing scheme that reduces the number of address bits needed in MoT significantly (from 13 to 5 for a 4x4 MoT). This plays a major role in reducing the network complexity. The minimum flit size can now to reduced to 16 bits which will definitely have positive impact over area and power profile of the NoC. Novel contributions of this paper are as follows. We present an improved addressing scheme for MoT routers. Next a novel MoT routing algorithm based on dimension order routing has been proposed. Routing algorithm has been shown to be deadlock, starvation, and live-lock free and to follow the shortest

path. The router has been synthesized in Synopsys Design-Vision to get area estimate. It shows significant area reduction, compared to [2, 3, 4]. Performance of a 4x4 network has been evaluated.

3 Addressing Scheme

An example 4x4 MoT is shown in Fig. 1. The leaf routers marked as 'L'. A leaf level router connects to a stem level row router (marked 'RS') and to a stem level column router (marked 'CS'). The stem level routers are further connected to root level router (marked 'R'). In our addressing scheme we identify only leaf routers and cores. We allocate an address to a leaf router based on its position in the two dimensional grid (matrix). We allocate a group address to a stem router. The group address is decided by the neighboring routers. For row stem (RS), group address will be decided by common bit (s) of column number of neighboring routers, located one level lower than RS. Similarly, column stem (CS) group addresses are decided. Since each leaf router can have two cores attached to it, complete address for a core would be [core_id,column_number,row_number].

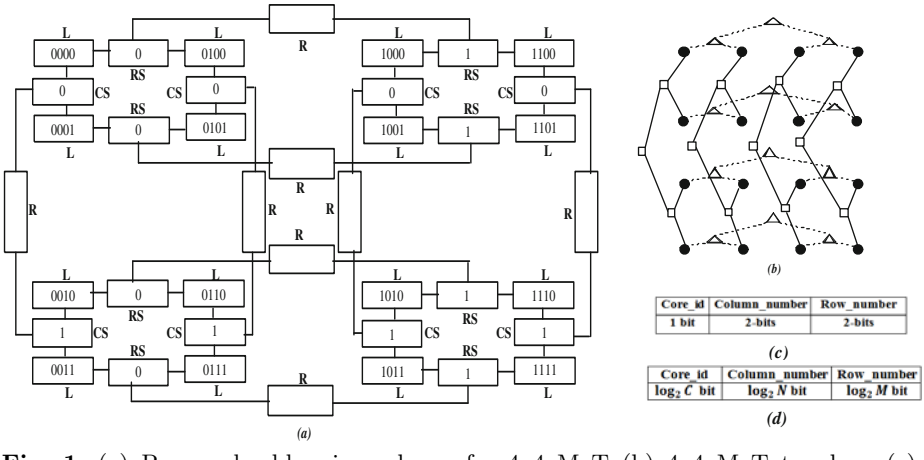


Fig. 1. (a) Proposed addressing scheme for 4x4 MoT (b) 4x4 MoT topology (c) Addressing format for 4x4 MoT (d) Addressing format for MxN MoT

4 Routing Algorithm

We first present the routing algorithm for a leaf level router. The leaf level router has its own address (CURRENT_ADDR in the algorithm). A leaf router is connected two cores (CORE_LINK) and two stem level routers (ROW_LINK and COL_LINK). A packet with destination address DESTINATION_ADDR is routed via the router as follows. The row part of DESTINATION_ADDR is matched with row part of CURRENT_ADDR. The variable ROW_CHECK is set accordingly. Similarly, the variable COL_CHECK is set by comparing the column parts of DESTINATION_ADDR and CURRENT_ADDR. If both ROW_CHECK

and COL_CHECK are *true*, the packet has reached the destination router, in which case it is forwarded to appropriate core based on the CORE_ID field of DESTINATION_ADDR. Otherwise, if ROW_CHECK is *true*, the packet is in the row to which it belongs. It is forwarded to the row stem(ROW_LINK). Otherwise, the packet is routed to the column stem (COL_LINK). Next we present the routing algorithm followed by a stem-level router at level l. We consider the leaf-level routers to form level-0 (l=0) of the tree. A stem-level router at level l connects to two $(l-1)$ -level routers - we call them LEFT_LINK and RIGHT_LINK respectively. It also connects to another router at level $(l+1)$, we call it UP_LINK. As noted earlier, a stem routers possesses a group-id.If a packet with destination address DESTINATION_ADDR reaches a stem-router of row tree at level l, the group-id of the packet is given by the most significant l bits of the column number of DESTINATION_ADDR. For a column-tree router the value is obtained from the row number part of DESTINATION_ADDR. This is accomplished by the function *router_group_addr()* in the algorithm. If the value does not match with the group-id of the router, the packet is forwarded to the UP_LINK. Else, the packet belongs to a core in the tree rooted at this stem router. If the $(l-1)^{th}$ bit (counted from MSB) is 0, the packet is routed to the LEFT_LINK, else, the packet is routed to the RIGHT_LINK.

The root routers do not need any routing algorithm. They just copy the packet arriving on one link to the other, acting like a buffer.

Algorithm 1. Proposed Algorithm for Leaf Router

Input: *DESTINATION_ADDR*
Output: *Generate request for CORE, or ROW_LINK, or COL_LINK*
if *row(CURRENT_ADDR) = row(DESTINATION_ADDR)* **then**
  $ROW\_CHECK \leftarrow true$
end
else
  $ROW\_CHECK \leftarrow false$
end
if *col(CURRENT_ADDR) = col(DESTINATION_ADDR)* **then**
  $COL\_CHECK \leftarrow true$
end
else
  $COL\_CHECK \leftarrow false$
end
if *ROW_CHECK = true and COL_CHECK = true* **then**
  Send packet to *CORE_LINK* base on *CORE_ID* of DESTINATION_ADDR.
end
else if *ROW_CHECK = true* **then**
  Route packet through *ROW_LINK*.
end
else
  Route packet through *COL_LINK*.
end

Algorithm 2. Proposed Algorithm for l^{th} level Stem Router

```
Input: DESTINATION_ADDR
Output: Generate request for UP_LINK, or LEFT_LINK, or RIGHT_LINK
if GROUP_ID = router_group_add(DESTINATION_ADDR) then
    UP_LINK ← false
end
else
    UP_LINK ← true
end
if UP_LINK = false then
    if extract_bit(DESTINATION_ADDR, l − 1) = 0 then
        (* l indicates at which level of tree the stem is present.*)
        Route packet through LEFT_LINK.
    end
    else
        Route packet through RIGHT_LINK.
    end
end
else
    Route packet through UP_LINK.
end
```

5 Guaranteed Shortest Path

All possible transactions in an MxN MoT can be enumerated as follows.

1. Source and destination cores are in same leaf router.
2. Source and destination cores are in same row but in different columns.
3. Source and destination cores are in same column but in different rows.
4. Source and destination cores are in different rows and columns.

Here, we will ensure that packet will always reach to the destination using shortest path. We are assuming that there is no error in the router and link. In our addressing scheme we have allocated address to a router based on its position (column number,row number). In a particular row or column, addresses are in strictly increasing order. Thus, for routers at row 'x' addressing order will be $(y_0, x) < (y_1, x) < (y_2, x) < (y_3, x) < ... < (y_N, x)$, y_i identifying the column number of the i^{th} router. For a particular column, say 'y', addressing order will be $(y, x_0) < (y, x_1) < (y, x_2) < (y, x_3) < ... < (y, x_M)$. In a dimension order routing, shortest path is ensured if the traversal does not violate the ordering. According to our algorithm, a MxN MoT path between any source and destination will be formed based on one of the comparison sequences noted next.

1. $[\{(Y_r = C_r) \wedge (Y_c = C_c)\} \rightarrow PC]$
2. $[\{(Y_r = C_r) \wedge (Y_c \neq C_c)\} \rightarrow RL] \vdash [\{Y_c(m : 1) > G(RS_1)\} \rightarrow UP_1^+] \vdash [\{Y_c(m : 2) > G(RS_2)\} \rightarrow UP_2^+] \vdash \ ... \ \vdash [\{Y_c(m : l) > G(RS_l)\} \rightarrow UP_l^+] \vdash [Y_c(l) \rightarrow (L_0|R_1)] \vdash [Y_c(l-1) \rightarrow (L_0|R_1)] \vdash \ ... \ \vdash [Y_c(0) \rightarrow (L_0|R_1)] \vdash$ [*Apply sequence* 1]

3. $[(Y_r \neq C_r) \rightarrow CL] \vdash [\{Y_r(m:1) > G(CS_1)\} \rightarrow UP_1^+] \vdash [\{Y_r(m:2) > G(CS_2)\} \rightarrow UP_2^+] \vdash \ ... \ \vdash [\{Y_r(m:l) > G(CS_l)\} \rightarrow UP_l^+] \vdash [Y_r(l) \rightarrow (L_0|R_1)] \vdash [Y_r(l-1) \rightarrow (L_0|R_1)] \vdash \ ... \ \vdash [Y_r(0) \rightarrow (L_0|R_1)] \vdash$ [*Apply sequence* 2]
4. $[\{(Y_r = C_r) \wedge (Y_c \neq C_c)\} \rightarrow RL] \vdash [\{Y_c(m:1) < G(RS_1)\} \rightarrow UP_1^-] \vdash [\{Y_c(m:2) < G(RS_2)\} \rightarrow UP_2^-] \vdash \ ... \ \vdash [\{Y_c(m:k) < G(RS_k)\} \rightarrow UP_k^-] \vdash [Y_c(k) \rightarrow (L_0|R_1)] \vdash [Y_c(k-1) \rightarrow (L_0|R_1)] \vdash \ ... \ \vdash [Y_c(0) \rightarrow (L_0|R_1)] \vdash$ [*Apply sequence* 1]
5. $[(Y_r \neq C_r) \rightarrow CL] \vdash [\{Y_r(m:1) < G(CS_1)\} \rightarrow UP_1^-] \vdash [\{Y_r(m:2) < G(CS_2)\} \rightarrow UP_2^-] \vdash \ ... \ \vdash [\{Y_r(m:l) < G(CS_l)\} \rightarrow UP_l^-] \vdash [Y_r(l) \rightarrow (L_0|R_1)] \vdash [Y_r(l-1) \rightarrow (L_0|R_1)] \vdash \ ... \ \vdash [Y_r(0) \rightarrow (L_0|R_1)] \vdash$ [*Apply sequence* 4]

where m : Most Significant Bit (MSB), $0 \leq l \leq log_2 N$, $0 \leq k \leq log_2 M$.

Table 1. Symbols used in the sequences

Symbol	Description
Y_r	Row bits of destination address
Y_c	Column bits of destination address
C_r	Row bits of current leaf router's address
C_c	Column bits of current leaf router's address
$G(S_l)$	Would return group address of a stem router (S) at level l
$Y_r(a:b)$	Bits from position a to b (including both) of Y_r
$Y_r(a)$	Bit at position a of Y_r
$L_0\|R_1$	Take left; if input is 0, Take right; if input is 1
RL	Row Link
CL	Column Link
PC	Packet Consumption
${UP_i}^+$	Move upwards at level i , when packet moving in increasing order
${UP_i}^-$	Move upwards at level i, when packet moving in decreasing order
$\vdash$	Next router
$(a\|b)$	Either a or b

Comparison sequence (1) signifies that the row number and column number of destination address are same as the address of the current leaf router. So packet is for consumption by a core attached to that router. Comparison sequence (2) corresponds to the case in which row number of destination address is same as that of the current leaf router, but column numbers do not match. It implies that the packet has to move towards row link (RL), means packet will be forwarded based on column number. Row stem at level-1 will receive the packet. It will compare group address with destination address. Destination's group address bits will consist of MSB to bit 1 of column bits. The group address of stem router will be given by $G(RS_1)$. If stem's group address is lesser than group address returned by $G(RS_1)$, the packet will be forwarded to up direction (${UP_i}^+$). Here '+' indicates that packet is moving from a lower column number to a higher

column number and i indicates the level to which the packet will move to. In this way packet will reach its corresponding level, where group address of the stem router will match with group address of the destination address. After that, the packet will come down level by level through left or right link up to level-0, as in all these cases group address will match. The packet will move based on the l^{th} bit of column number Y_c, l being the level of the router. At level 0, packet will come to leaf router and then based on condition (1) packet will reach the core. Similar explanation can be given for sequence 3, 4, and 5.

All these comparison sequences maintain a order that is either increasing or decreasing. If it encounters a path with comparison sequence not strictly increasing or decreasing, (i.e. a mix of these two), then that path could not be a shortest path. That comparison sequence implies there is a violation in dimension ordering. But we have already noted that dimension order is proper and assumed the router to be error free. Hence our algorithm always provide shortest path.

6 Freedom from Deadlock, Starvation, and Live-lock

A situation where a packet waits for an event that cannot happen is called deadlock. For example, a packet may wait for a network resource (bandwidth or FIFO) to be released by a packet that is, in turn, waiting for the first packet to release some resource. In worm-hole switching, such a circular wait condition causes deadlock. In dimension ordered routing algorithm, circular wait means violation in dimension ordering. As dimension is already ordered, routing follows shortest path technique. So our algorithm is deadlock free.

Starvation is similar to deadlock. This is a situation when a packet waits for an event that can happen but never does. For example, a packet may wait forever to get a network resource. Some other packets are also competing for the same network resource, but those packets always successfully get that resource. Starvation is possible only when the distribution of network resources is not uniform. Packets cannot be injected into the network or move inside network because of deadlock and starvation. We are using Round-Robin arbitration technique, which allocates the channel uniformly. So our routing technique is starvation free.

In contrast, live-lock does not stop the movement of a packet in network, but rather stops its progress toward the destination. Live-lock happens, when routing is adaptive and non-minimal [5]. Ordering the dimensions not only ensures that the xy routing is free from deadlock, but also its non-adaptiveness. So our algorithm is free from live-lock as well.

7 Performance Metrics

For evaluating a NoC architecture, we will have followings metrics [7]: Throughput, Latency, Area Overhead, and Energy Consumption.Throughput is a measure of how much data can be transmitted across a network. For a message passing system throughput T is given by, $T = (Total\ messages\ completed \times$

$Message\ length)/(Number\ of\ IP\ cores \times Time)$. Throughput is measured in terms of flits/cycle/IP. For our case 32 number of IP cores are used and message length is 64 flits.

Network bandwidth refers to the maximum number of bits that can be sent successfully to the destination through the network per unit time. It is represented as bits/sec (bps). It defines as follows.$BW = ((Throughput \times No.\ of\ IP\ cores \times flits\ size))/(Clock\ period)$. Latency can be defined as the time interval (in terms of clock cycle) between transfer of header flits by the source and the receipt of tailer flit by the destination. Let P be the number of packets received in a given time period and let L_i be the latency of the i^{th} packet. Average latency is defined as,$L_{avg} = (\sum_{i=1}^{P} L_i)/(P)$. In NoC design, area overhead comes due to routers, repeaters, network interface, and links. To estimate the silicon area occupied by each router, we have developed their Verilog models and synthesized using *Synopsys Design Vision in 90 nm* CMOS technology supporting *Faraday* library to generate gate-level net-list. The synthesis tool reports the silicon area occupied by the design in μm^2.

8 Simulation Methodology

For evaluating the performance of the interconnection network, we have designed a cycle-accurate System-C based simulator. Cores are modeled via traffic generators and receptors. Traffic generator uses self-similar model for generating the traffic. Real traffic follows bursty nature. Self-similarity is also bursty in nature. We categorized traffic as uniform and locality based. For example, if locality factor is 0, then traffic from a core will distributed uniformly to all other core. If locality factor is 0.3 then 30% traffic will go to nearer cores and remaining 70% will go to other cores. For 4x4 MoT, the possible distances (d) of the destinations from any source are d= 0, 2, 4, 6, and 8. There is only one destination core at the nearest cluster. We derived traffic for our simulation from the traffic generated by model noted in [4]. Each packet contains 64 flits.

9 Experimental Results and Analysis

In this section, we have applied the above mentioned evaluation methodology to find out the performance and area overhead of MoT based network consisting of 32 cores by varying offered loads (N) in self-similar traffic.

9.1 Accepted Traffic vs. Offered Load

The accepted traffic depends on the rate at which the IP blocks inject traffic into the network. Ideally, accepted traffic should increase linearly with increasing load at a slope of 45°. However, due to the limitation of routing and arbitration strategy and unavailability of enough buffer space within the wormhole router, the network suffers from contention. Therefore, the accepted traffic saturates after a certain value of the offered load. Fig. 4(a) depicts this scenario for uniformly

distributed traffic in MoT based network. The maximum accepted traffic where the network is saturated is termed as throughput as defined above and it relates to the maximum sustainable data rate by the network. In Fig. 4(a) and others, the graphs marked "16 bits flit" are our results, where as, the graphs marked "32 bits flit" are those corresponding to [2, 3, 4]. Science throughput is not dependent on flit size the value do not differ.

9.2 Throughput vs. Locality Factor

The effect of traffic spatial localization on network throughput is shown in Fig. 4(b) and 4(c). As the locality factor increases, more number of traffic are destined to their local clusters, thus traversing lesser number of hops which in turn increases the throughput.

9.3 Average Overall Latency vs. Locality Factor

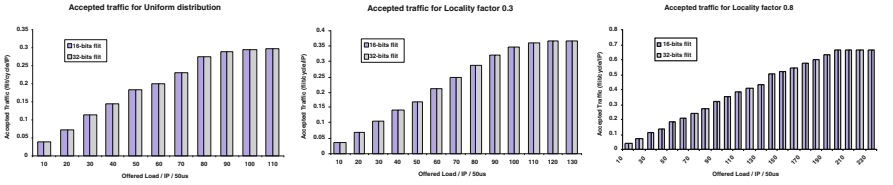
The average overall latency of any network depends on both the offered load and the locality factor. Fig. 5(a) shows the average overall latency profile for uniformly distributed offered load in MoT. It shows that at lower traffic, the latency variation is not significant. This is due to the fact that at lower traffic, the contention in the network is less, but it will increase as the offered load increases, which in turn increases the latency. The simulation result shows that as the offered load increases towards the network saturation point, the latency increases exponentially which signifies that the packets will take much longer time to reach their destinations. So, it is always desirable to operate the network below its saturation point. The effect on traffic spatial localization on average overall latency is also studied and is shown in Fig. 5(b) and 5(c). As the locality factor increases, more number of traffic will go to their local clusters, hence lesser contention in the network. Therefore, the network will be able to carry more traffic before going to saturation, which in turn enhances the operating point of the network.

9.4 Bandwidth Variation

Fig. 6 shows the bandwidth variation of our approach and compares with [2, 3, 4]. Since our flit size is 16-bit, compared to 32-bit for the previous work, bandwidth is also halved. However it should be noted that our router design does not put any restriction on flit size, which can be increased at will, and thus improving the bandwidth. The previous work [2, 3, 4] cannot have a flit size lesser than 32 bits due to address size restrictions.

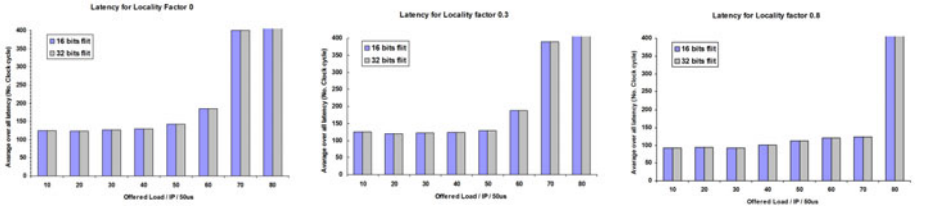
9.5 Area Overhead

With a restriction in flit size, the FIFO width of the routers also decreases in our work compared to [2, 3, 4]. The simplicity of routing also adds to this reduction. Fig. 7 shows the area reduction. We are getting 61% area reduction in router design and 46.38% area reduction in link area over [2, 3, 4]. The overall area improvement is about 45%.



(a) (b) (c)

Fig. 2. Throughput variation profile for different locality factor



(a) (b) (d)

Fig. 3. Latency variation profile for different locality factor

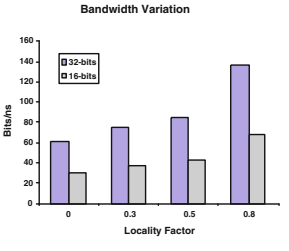


Fig. 4. Bandwidth variation

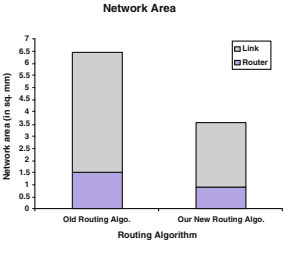


Fig. 5. Network area variation

10 Conclusion

In this paper we have proposed a simple and novel routing algorithm. For simplicity of routing algorithm we got reduction in area. This is expected to yield reduction in power. Due to the simplified addressing scheme, the minimum flit size could be reduced to 16-bits. We compared the performance of our work with the earlier work [2, 3, 4]. At minimum flit size, though we are losing bandwidth, it can be improved easily. Our design does not put any restriction. This flexibility will help us in SoC design, as we will be able to use reduced complexity routers with lower flit size for tasks demanding lesser bandwidth. For tasks demanding higher bandwidth, higher flit size can be utilized. The works reported in [2, 3, 4] lack this flexibility. We are currently working on obtaining the power profile for our NoC design.

References

[1] Flich, J., Rodrigo, S., Duato, J.: An Efficent Implementation of Distributed Routing Algorithm for NoCs. In: Procedings of Second ACM/IEEE International Symposium on Networks-on-Chip, pp. 87–96 (2008)

[2] Kundu, S., Chattopadhyay, S.: Network-on-chip architecture design based on Mesh-of-tree deterministic routing topology. Int'l Journal of High Performance System Architecture 1(3), 163–182 (2008)

[3] Kundu, S., Chattopadhyay, S.: Mesh-of-tree deterministic routing for network-on-chip architecture. In: In Procedings of ACM Great Lakes Symposium on VLSI 2008, pp. 343–346 (2008)

[4] Kundu, S., Manna, K., Gupta, S., Kumar, K., Parikh, R., Chattopadhyay, S.: A Comparative Performance Evaluation of Network-on-Chip Architectures under Self-Similar Traffic. In: Procedings of International conference on Advances in Recent Technologies in communication and Computing, pp. 414–418 (2009)

[5] Glass, C.J., Ni, L.M.: Turn Model for Adaptive Routing. In: Proceedings of the 19th annual international symposium on Computer architecture (ISCA 1992), pp. 278–287 (1992)

[6] Balkan, A.O., et al.: A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing. In: IEEE 17th International Conference on Application-specific Systems, Architectures and Processors, ASAP 2006 (2006)

[7] Pande, P.P., Grecu, C., Jones, M., Ivanov, A., Saleh, R.: Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures. In IEEE Transaction on computers 54(8), 1025–1040 (2005)