# Multi-path Routing for Mesh/Torus-Based NoCs

Yaoting Jiao[1], Yulu Yang[1], Ming He[1], Mei Yang[2], and Yingtao Jiang[2]

[1]*College of Information Technology and Science, Nankai University, China*
[2] *Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA*
*Emails: [1]yangyl@nankai.edu.cn, [2]{meiyang, yingtao}@egr.unlv.edu*

## Abstract

In networks-on-chip (NoC) designs, delay variations and crosstalk noise have become a serious issue with the continuously shrinking geometry of semiconductor devices and the increasing switching speed. The crosstalk between adjacent lines causes data dependent signal delay and noise, thus finally makes the communication channel unreliable. The crosstalk problem can be mitigated by wide spacing of serial lines, however, the wider spacing of serial lines will reduce the number of the lines, thus reduce the data throughput. In this paper, we propose a multi-path routing scheme to maximize the data throughput by utilizing multiple paths for concurrent data transmission. For the proposed multi-path routing scheme, we consider two transport models: the multi-path full bitbank transport model and the multi-path half bitbank transport model. Through theoretical analysis, we show that the proposed multi-path scheme achieves significant improvement in data throughput under both transport models.

## 1. Introduction

Due to the high degree of integration and limited chip geometry, future Network-on-Chips (NoCs) will become more sensitive and prone to delay variations, noise, transient faults, and other interferences [11]. One of the main noise sources is crosstalk, which becomes a serious issue with technology scaling and can cause errors across a range of adjacent bits [7] [11]. The crosstalk problem can be mitigated by wide spacing of adjacent wires [7]. However, for a fixed chip area, wider spacing of adjacent wires will reduce the number of wires between routers, thus reduce the data throughput.

With their simple structure, mesh/torus-type networks are widely used as on-chip interconnection networks [5][8]. On mesh/torus-type networks, there exist multiple paths between any pair of source and destination nodes, but the traditional routing schemes only choose one of them for data transmission. Based on this observation, we propose the multi-path routing scheme, which features in separating the data message to be sent into multiple data streams and sending them on different paths concurrently. Employing such a scheme, the data throughput can be retained while the crosstalk is reduced when wider spacing between adjacent wires is used. When the spacing between adjacent wires is unchanged, a higher data throughput can be achieved using this scheme.

In this paper, we present our study of an adaptive multi-path routing scheme on torus-type networks. We consider two transport models: the multi-path full bitbank transport model (FM) and the multi-path half bitbank transport model (HM). The proposed routing scheme is the same under both transport models. An important aspect of an adaptive routing scheme is deadlock avoidance. We justify that the multi-path routing is deadlock-free as it employs the same rule of using virtual channels as in the deadlock-free routing algorithm proposed by Dally and Seitz [4]. We further show that in the situation of single source, the bit streams transported on multiple shortest paths will not block each other. Using an analysis model, we show that significant improvement in data throughput is achieved using the proposed scheme under both the FM and HM models.

The rest of the paper is organized as follows. Section 2 describes the two transport models. Section 3 presents the multi-path routing scheme. Section 4 is focused on the proof of the blocking avoidance with single source node in the network. Section 5 gives the analysis of the data throughput achieved using the proposed routing scheme. Section 6 concludes the paper.

## 2. Preliminaries

### 2.1 Node and Channel Models

Each node in a torus-based NoC network is composed of a processor and a router which connects the processor node to the interconnection network. For simplicity, we represent a node as square in all figures. And we represent all nodes in a torus-based NoC as a $2N$x$2N$ matrix, where each node is indexed with a pair of coordinates ($x$, $y$), $0 \le x \le 2N-1$ and $0 \le y \le 2N-1$, on the $X$ and $Y$ dimensions, respectively.

Each node in the NoC has four physical channels, each connecting to a neighbor node. Fig. 1 shows the directions of the four channels.

We also assume that the physical channel on each direction of a node is split into two virtual channels, $v_0$ and $v_1$. The rule of using the virtual channels is the same as in the deadlock-free routing algorithm proposed by Dally and Seitz [5]. Hence we can justify that the multi-path routing scheme is also deadlock-free [5].

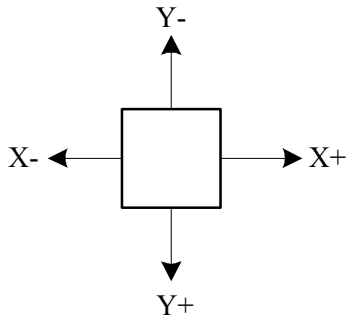


**Fig. 1 Directions of the four channels.**

## 2.2 Transport Models

We consider two transport models of the multi-path routing scheme: the full-wire-bank multi-path transport model, and the half-wire-bank multi-path transport model, which are same on the routing algorithm and transport control but different on their usage of the wire bank and the buffer size.

In the FM model, all wires on each communication link will be used for data transmission. When the source node needs to send data to a destination node, it will first compute the number of the shortest paths between the source and destination nodes, then partition the message into multiple data streams and send each on one of the shortest paths. Fig. 2 illustrates an example of the FM model on a 4x4 torus. In this example, node 01 is the source node and node 22 is the destination node. Three shortest paths (indicated by dashed lines with arrowhead in the figure) will be used for transmitting three data streams.

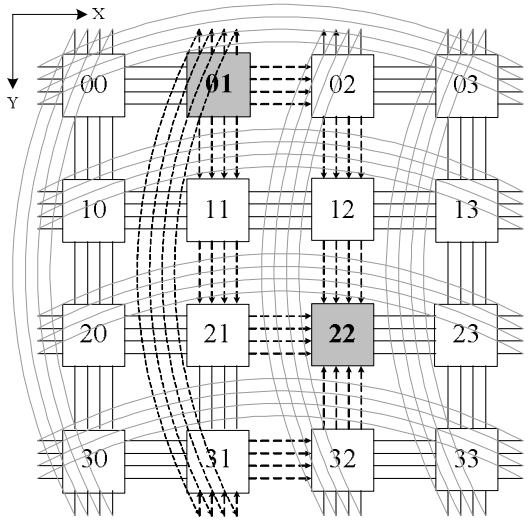


**Fig. 2 FM model on 4x4 torus.**

Similar to the FM model, in the HM model, the source node will compute all the shortest paths to the destination node and send data along all the paths. But different from the FM model, each data stream will be transmitted on half of the wires (either on odd numbered wires or even numbered wires) on each link to avoid crosstalk. Compared to the FM model, the crosstalk in the HM model is dramatically reduced according to the study in [2]. Fig. 3 illustrates the HM model on a 4x4 torus network with the same source and destination nodes as in Fig. 3. Three shortest paths (indicated by dashed lines with arrowhead in the figure) are also used in this model.

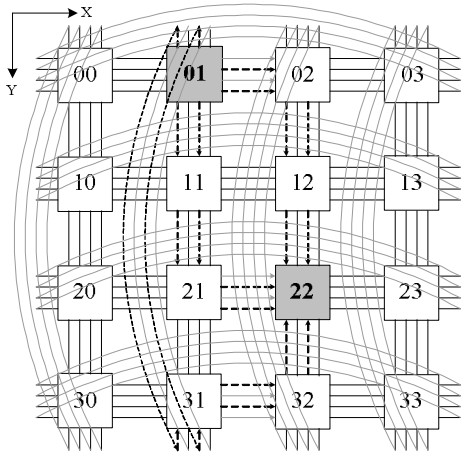


**Fig. 3 HM model on 4x4 torus.**

In the following, we will not differentiate the two models when we describe the routing scheme. We will analyze and compare their performance in Section 5.

# 3. Multi-path Routing Scheme

Before we describe the multi-path routing scheme, we first introduce the concept of priority dimension.

## 3.1 Priority Dimension

We have the following definitions.

**Definition 1 (Boundary nodes).** A node which has either 0 or $2N$-1 in one of its index number ($x$ or $y$) is called a *boundary node*. And the channel that connects two boundary nodes is called a *boundary channel*.

**Definition 2 (Slop over).** When a data stream is transmitted from one boundary node with its $x$ or $y$ is 0 (or $2N$-1) to another boundary node with its $x$ or $y$ is $2N$-1 (or 0), we say that the data stream *slops over*.

**Definition 3 (Reach boundary).** When a data stream is transmitted from a non-boundary node to a boundary node, we say the data stream *reaches boundary*.

**Definition 4 (Priority direction).** The *priority direction* of a data stream is the direction of the channel that connects the current node (the source node or an intermediate node) to the next node along the path. As shown in Fig. 1, there are four possible priority directions.

**Definition 5 (Priority dimension).** The *priority dimension* is the dimension that the priority direction belongs to. Note that for torus-type networks, each priority dimension can have maximally two priority directions.

The priority dimension and priority direction for a data stream at a particular node will be changed according to the following rules.

*Rule I*: When a data stream reaches a node, the node will find out the output directions of the shortest paths according to its index and the index of the destination node of the data stream. If there are several directions to choose, the direction on the priority dimension will be chosen. If two directions of

the priority dimension can be chosen, the non-slop over direction (i.e., the dimension that the direction belongs to won't slop over on the path from the current node to the destination node) will be chosen.

*Rule II*: When the bit stream reaches the boundary, the node will decide whether the priority dimension should be changed from dimension $X$ ($Y$) to dimension $Y$ ($X$) according to the values of the control bits (which will be discussed in Section 3.2).

*Rule III*: When there is blocking on the selected direction, the priority dimension will be decided such that blocking can be avoided.

The purpose of Rule II is to make the data streams from the same message will not block each other when there is only one source node in network at a time (as will be proved in Section 4). It is important to point out that the priority dimension can change only once for a data stream.

In the following, we will describe the multi-path routing scheme, which is composed of the operations at the source node and at intermediate nodes.

## 3.2 Operations at the Source Node

At the source node, the number of shortest paths (corresponding to the number of data streams that can be sent out) is determined based on the difference between the indexes of the source node and the destination node, which is explained as follows.

Let $(x_S, y_S)$ and $(x_D, y_D)$ denote the indexes of the source and destination nodes, respectively. Then we assign $x' = x_D - x_S$, $y' = y_D - y_S$, and name $x'$ as the low-order difference value, and $y'$ as the high-order difference value. The value of $x'$ or $y'$ falls in four different cases, each corresponding to a different operation, as shown in the following table

**Table 1 The different cases and corresponding operations of $x'$ or $y'$.**

| Cases | Value of $x'$ or $y'$ | Corresponding operation |
|---|---|---|
| 1 | [1, $N$-1] or [-$N$, -($N$+1)] | The data stream needs to travel along the positive direction of the $X$/$Y$ dimension |
| 2 | [-($N$-1), -1] or [$N$+1, $N$] | The data stream needs to travel along the negative direction of the $X$/$Y$ dimension |
| 3 | $\pm N$ | The data stream can travel along the positive direction or the negative direction of the $X$/$Y$ dimension. |
| 4 | 0 | The data stream doesn't need to travel on the $X$/$Y$ dimension |

The number of shortest paths is then decided by the combination of the cases of the values of $x'$ and $y'$, as shown in Table 2.

After determining the number of shortest paths, the source node will check how many output ports that are available and decide the actual number of data streams that can be sent out. Then the node partitions the message into the actual number of data streams and sends the data streams on the shortest paths through the corresponding output ports.

**Table 2 The number of shortest paths vs. the combination of different cases and corresponding operations of $x'$ or $y'$.**

| Combination | Case of $x'$ value | Case of $y'$ value | # of shortest paths | C's value |
|---|---|---|---|---|
| 1 | Case 1 or Case 2 | Case 4 | One | 1 |
| | Case 4 | Case 1 or Case 2 | | |
| 2 | Case 1 or Case 2 | Case 1 or Case 2 | Two | 1 |
| 3 | Case 1 or Case 2 | Case 3 | Three | 1 if (\|$x'$\|+\|$y'$\|) mod $N$ < $N$<br>0 otherwise |
| | Case 3 | Case 1 or Case 2 | | |
| 4 | Case 3 | Case 3 | Four | 0 |

Each data stream contains the source node index, destination node index, and two control bits ($D$ and $C$), which will be used for making the routing decision at the intermediate nodes on the path.

$D$ is used to record the priority dimension and $D$=0 or 1 represents the priority dimension is $X$ or $Y$, respectively.

$C$ is used to indicate if the priority dimension should be changed when the data stream reaches boundary and $C$=0 or 1 represents the priority dimension should be changed or should not be changed, respectively. The setting of $C$ is shown in Table 2, where (|$x'$|+|$y'$|) mod $N$ calculates the distance between the source and destination nodes.

## 3.3 Operations at Intermediate Nodes

Once receiving the data stream, each intermediate node will decide which output port it will forward the data stream, i.e., the corresponding priority direction to take. The decision is based on the following calculation

Each node will first calculate the difference between its index and the index of the destination node as $x'' = x_D - x_C$, $y'' = y_D - y_C$, where $(x_C, y_C)$ represents the index of the current intermediate node. And three sets of binary variables ($Ax$, $Ay$), ($Bx$, $By$), and ($Cx$, $Cy$) are derived, where

$Ax$ = 0 or 1 represents that $x''$ is positive or negative, respectively,

$Ay$ = 0 or 1 represents that $y''$ is positive or negative, respectively,

$Bx$ = 0 or 1 represents if $|x''| \neq N/2$ or not, respectively,

$By$ = 0 or 1 represents if $|y''| \neq N/2$ or not, respectively,

$Cx$ = 0 or 1 represents whether $x'' = 0$ or not, respectively,

$Cy$ = 0 or 1 represents whether $x'' = 0$ or not, respectively.

Then the priority direction on each dimension is determined according to the combination of these variables as shown in the following table.

The final priority direction of the incoming data stream can be determined based on Table 3 and the value of $D$. The directions on $X$ dimension is more preferred than those of $Y$ dimension for $D$=0, and vice versa for $D$=1.

**Table 3 The priority direction on $X$ and $Y$ dimension.**

| Ax Bx Cx | Priority direction | Other direction | *Ay By Cy* | Priority direction | Other direction |
|---|---|---|---|---|---|
| $X$ 0 0 | none | none | $X$ 0 0 | none | none |

| | | | | | |
|---|---|---|---|---|---|
| 0 0 1 | *X*+ | none | 0 0 1 | *Y*+ | none |
| 1 0 1 | *X*- | none | 1 0 1 | *Y*- | none |
| 0 1 1 | *X*+ | *X*- | 0 1 1 | *Y*+ | *Y*- |
| 1 1 1 | *X*- | *X*+ | 1 1 1 | *Y*- | *Y*+ |
| *X* 1 0 | impossible | | *X* 1 0 | impossible | |

Then the node will check if the availability of the output ports at the preferred directions following the order of the priority direction on the preferred dimension, other direction on the preferred dimension, the priority direction on non-preferred dimension, other direction on non-preferred dimension. The data stream is sent out from the first output port that is available. If there is no free output port, the data stream is blocked at the current node.

In the case that there is no direction to choose, the data stream reaches its destination node.

According to the above description, we can see that each data stream can choose its path based on the availability of the output ports of the current node, so the proposed routing scheme is an adaptive one.

The detailed steps carried at each intermediate node after receiving a data stream are listed below.

**Step 1.** Check that whether the bit stream reaches its destination. If it does, store the data stream to memory; otherwise, continue the following steps.

**Step 2.** When the data stream reaches boundary, if $C$=0, then flips the priority dimension (namely changes $D$'s value from 0 to 1 or 1 to 0) and sets $C$=1; otherwise, continue the following steps.

**Step 3.** Decide the output directions of the bit stream according to ($Ax$, $Ay$), ($Bx$, $By$), ($Cx$, $Cy$) and two control bits ($C$ and $D$) as described above. Then the data stream is sent out to one of the output ports or blocked at the current node.

## 4. No Blocking under Single Source

One of the advantages of the proposed multi-path routing scheme is that a message is partitioned into several data streams and transmitted on multiple paths, which helps reducing the transmission time. A question one may have is that if the data streams will conflict at some intermediate nodes. We have the following theorem.

**Theorem 1** There is no blocking when there is one pair of source and destinations nodes transmitting data.

**Proof.** Here we sum up all the possible pairs of communicating nodes into four cases. And the proof will be made for each case.

**Case 1.** One shortest path exists.

Since there is only one path, there is only one data stream on the network. Obviously, there is no blocking.

**Case 2.** Two shortest paths exist.

There are two subcases under this case.

1) Two output directions are on the same dimension. Since the distance only exists on one dimension between the source and destination nodes, the two data streams travel on the same loop line but on the opposite directions. Hence, they can only meet at the destination node. No blocking exists for this subcase.

2) Two output directions are on different dimensions. As shown in Table 2, $C$ is set to 1, i.e., they won't change the priority dimension. Hence, the two paths are set up based on $X$-$Y$ routing and $Y$-$X$ routing, respectively. Therefore, the two data streams can only meet at the destination node. No blocking is possible.

**Case 3.** Three shortest paths exist.

For this case, according to Tables 1 and 2, we can derive that the distance on one dimension must be $N$. Assume that on the $2N$*$2N$ torus network, the source node ($x$, $y$) is on the up-left direction of the destination node ($x$*, $y$*). And $x$*=$x$+$n$, $n\in[0, N)$. Then the distance on dimension $Y$ between source and destination must be $N$, namely ($x$*, $y$*) = ($x$+$n$, $y$+$N$), $x\in[0, N]$, $y\in[0, N\text{-}1]$.

According to the proposed routing scheme, the paths of three data streams from the source node to the destination node can be represented as follows:

Data stream A: ($x$, $y$) -> ($x$+$n$, $y$)
($x$+$n$, $y$) -> ($x$+$n$, $y$+$N$)

Data stream B: ($x$, $y$) -> ($x$, $y$+$N$)
($x$, $y$+$N$) -> ($x$+$n$, $y$+$N$)

Data stream C: ($x$, $y$) -> ($x$, 0) ($y\neq0$)
($x$, 0) -> ($x$+$n$, 0)
($x$+$n$, 0) -> ($x$+$n$, $y$+$N$)

When y=0, the route of data stream $C$ is:

Data stream $C$: ($x$, 0) -> ($x$, 2$N$-1)
($x$, 2$N$-1) -> ($x$+$n$, 2$N$-1)
($x$+$n$, 2$N$-1) -> ($x$+$n$, $y$+$N$)

Then we can sum up all the segments occupied by three paths into dimension X and dimension Y. So the following results can be obtained.

Dimension $X$: ($x$, $y$) -> ($x$+$n$, $y$)
($x$, $y$+$N$) -> ($x$+$n$, $y$+$N$)
($x$, 0) -> ($x$+$n$, 0) ($y\neq0$)
($x$, 2$N$-1) -> ($x$+$n$, 2$N$-1) ($y$=0)

Dimension $Y$: ($x$+$n$, $y$) -> ($x$+$n$, $y$+$N$)
($x$, $y$) -> ($x$, $y$+$N$)
($x$, $y$) -> ($x$, 0) ($y\neq0$)
($x$+$n$, 0) -> ($x$+$n$, $y$+$N$) ($y\neq0$)
($x$, 0) -> ($x$, 2$N$-1) ($y$=0)
($x$+$n$, 2$N$-1) -> ($x$+$n$, $y$+$N$) ($y$=0)

For the segments on dimension X:

When $y\neq0$, since $y\in(0, N\text{-}1]$, we have $y$+$N\in(N, 2N\text{-}1]$. The three values $y$, $y$+$N$, and 0 do not equal to each other. Hence, the three segments occupied on dimension $X$ will not overlap with each other.

When $y$=0, $y$+$N$=$N$, then three values $y$, $y$+$N$, and 2$N$-1 do not equal to each other. The 3 lines occupied on dimension $X$ will not overlap with each other.

Hence, there is no block on dimension $X$.

For the segments on dimension $Y$:

They can be separated into two groups according to their values of dimension $X$ which are Group $x$ and Group $x$+$n$.

For Group $x$:

($x$, $y$) -> ($x$, $y$+$N$)

$(x, y)$ -> $(x, 0)$ $(y \neq 0)$

$(x, 0)$ -> $(x, 2N\text{-}1)$ (y=0)

When $y \neq 0$, the first two segments have the same source but to the opposite directions on dimension $Y$. Since $y$=<$N$-1, it is not possible for these two segments to overlap with each other.

When $y$=0, the first and third segments cannot overlap with each other.

Similarly, we can show that the segments in Group $x$+$n$ are not possible to overlap with each other.

Hence, there is no blocking on dimension $Y$.

Therefore, no blocking is possible for the three data streams in case 3..

**Case 4.** Four shortest paths exist.

Similar to the proof of Case 3, we can show that there is no blocking among the four data streams in Case 4.

Summarizing the proof of the four cases, we conclude that there is no blocking when there is a single source node on the network. ■

## 5. Analysis of the Throughput Improvement

The proposed routing scheme works under both FM and HM models but may result in different performance. As described in Section 2, the FM retains the data throughput but suffers from crosstalk, while HM model reduces the crosstalk but also has less data throughput using half of the wires. To compare the performance of the two models, we use the traditional single-path full wirebank transport model (TM) as the baseline. In the TM model, the source node only chooses one path to transfer data and all wires are used on each channel. We compare the FM and HM models to the TM model separately to analyze how much performance improvement each model can achieve.

### 5.1 Analysis of FM and HM Models with Single Source

Here we define the *rate of the valid data* as the amount of valid (correct) data that is transferred between two nodes connected by a channel composed of a group of wires in a unit time.

The *average number of shortest paths* is defined as the average of the number of the shortest paths for all the possible cases.

We use the following notations in our analysis.

$P1$: the rate of the valid data when full wirebank is used.

$P2$: the rate of the valid data when half wirebank is used.

$V$: the data rate of the TM model.

$E$: the average distance of the network.

$H$: the average number of the shortest paths.

$Fb$: the length of a message.

$Fh$: the length of the head of the bit stream.

$T$: the average transfer time of a message in the TM model.

$T1$: the average transfer time of a message in the FM model.

$T2$: the average transfer time of a message in the HM model.

For the torus network, the number of shortest paths from any node $X$ to other nodes in the network can be derived as below,

$$H = 1*(n_1 / N) + 2* (n_2 / N) + 3* (n_3 / N) + 4*(n_4 / N), \quad (1)$$

where, $N$ represents the amount of the node that can communication with node $X$, namely all the nodes except node $X$ in torus. In the following, we use 4x4 torus as an example to illustrate the analysis. For 4x4 torus, $N$ = 15. Let $n_i$ ($i$ = 1, 2, 3, 4) represent the number of nodes that there are $i$ different shortest paths for node $X$. For example, we can find that $n_1$ = 4, $n_2$ = 6, $n_3$ = 4, $n_4$ = 1 for 4x4 torus. Then we can get $P$ = 32 / 15 for 4x4 torus.

For both FM and HM models, each message is divided into $H$ data streams on average. For the TM model, there is only one data stream. Assuming wormhole switching is used, the average transfer time of a message under the TM model can be derived as

$$T = ((Fh * E + Fb )) / ( P1 * V ). \quad (2)$$

In the FM model, the message is transferred by $H$ data streams at the same time with rate $P1$. The length of its body on each data stream is $Fb$ / $H$. Hence the average transfer time $T1$ could be derived as

$$T1 = ((Fh * E + Fb / H)) / (P1 * V) \quad (3)$$

In the HM model, the message is transferred by $H$ data streams at the same time with rate $P2$. The length of its body is also given by $Fb$ / $H$. Thus, the average transfer time $T2$ can be derived as below,

$$T2 = ((Fh * E + Fb / H)) / (P2 * V / 2). \quad (4)$$

The ratio of the average transfer time of the FM model to that of the TM model is given by

$$T1 / T = ((Fh * E + Fb / H )) / ((Fh * E + Fb)). \quad (5)$$

Since $Fh$ and $E$ are very small compared with $Fb$, we can derive Eqn. (5) as

$$T1 / T = 1 / H. \quad (6)$$

That is to say, the speedup of the FM model to the TM model is 32 /15.

The ratio of the average transfer time of the HM model to that of the TM model is given by

$$T2 / T = (((Fh * E + Fb / H ) ) / ((Fh * E + Fb ))) * (( P1 * V ) / ( P2 * V / 2 )). \quad (7)$$

The $Fh$ and $D$ are very small relative to $Fb$, so we can get

$$T2 / T = (2 / H) * (P1 / P2). \quad (8)$$

The valid date rate under full wirebank is low than that of half wirebank due to the severer crosstalk existing in the full

wirebank. Then we have $P2 > P1$, i.e., $P1 / P2 < 1$.

Thus we get the speedup of the HM model to the TM model as 16 /15.

This confirms that both the FM and HM models are superior to the traditional model.

From Eqn. (3) and Eqn. (4), we can get

$$T1 / T2 = ( 1 / 2 ) * ( P2 / P1) \quad (9)$$

From Eqn. (9), we have

1) when $2 * P1 - P2 < 0$, $T1 / T2 > 1$, namely HM is better than FM.

2) When $2 * P1 - P2 > 0$, $T1 / T2 < 1$, namely FM is better than HM.

Particularly, when $P1 > 0.5$, $2 * P1 > 1$. And $P2 <= 1$, thus it is true that $2 * P1 - P2 > 0$. Namely when the FM is used, if the rate of valid data is larger than 0.5, the FM model is better than the HM model.

### 5.2 Analysis with Multiple Source Nodes

When there are more than one pair of communicating nodes on the network, there may exist blockings. Correspondingly, the performance of proposed routing scheme will degrade. Since multiple data streams generated from each message are transferred in the network, the probability of blockings in our models is larger than that in the traditional model when there exist multiple source nodes. On the other hand, because the proposed routing scheme improves the transfer speed and shortens the average transfer time, it may reduce the probability of blockings. Due to the complexity of this scenario, we will not detail the analysis in this paper.

## 6. Conclusion

In this paper, we addressed the problem of reducing crosstalk and retaining high data throughput in NoCs. We introduced the FM and HM transport models and proposed a deadlock-free minimal adaptive multi-path routing scheme to work under both models for torus-based NoCs. The proposed routing scheme features in using multiple shortest paths to transfer message concurrently with the help of simple control schemes. Through analysis, we showed that the proposed routing scheme achieves better valid data throughput under both the FM and HM models compared with the traditional single path routing scheme when there is single source node. Future work includes the analysis of the throughput improvement when multiple source nodes exist in the network.

### References


[1] L. Benini and G. DeMicheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[2] Crosstalk calculation and analysis, available at: http://www.eetchina.com/ARTICLES/2004MAY/1/2004MAY10_BD_NTFORUM01.HTM.

[3] W.J. Dally, "A VLSI architecture for concurrent data structures," PH.D. Dissertation, Dep. Comput. Sci., California Instit. Technol., Tech. Rep. 5209:TR:86, 1986.

[4] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, pp. 547-553, May 1987.

[5] W.J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proc. Design Automation Conf (DAC)*, 2001, pp. 684-689.

[6] N. Eisley and L-S. Peh, "High-level power analysis for on-chip networks," *Proc. CASES*, 2004, pp. 22-25.

[7] A. Jantsch and H. Tenhunen, *Networks on Chip*, Kluwar Academic Publishers, 2003.

[8] N. Kavaldjiev and G. M. Smit, "An energy-efficient network-on-chip for a heterogeneous tiled reconfigurable systems-on-chip," *Proc. Euromicro Symp. Digital System Design (DSD)*, 2004, pp. 492-498.

[9] K. Lee, S-J. Lee and H-J. Yoo, "SILENT: serialized low energy transmission coding for on-chip interconnection networks," *IEEE Int'l Conf. Computer Aided Design (ICCAD)*, 2004, pp. 448-451.

[10] P. Magarshack and P. G. Paulin, "System-on-chip beyond the nanometer wall," *Proc. Design Automation Conf. (DAC)*, 2003, pp. 419-424.

[11] D. Rossi, C. Metra, A. K. Nieuwland, and A. Katoch, "Exploiting ECC redundancy to minimize crosstalk impact," *IEEE Design & Test of Computers*, vol. 22, no. 1, pp. 59-70 , Jan 2005.