2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools

An Efficient Method to Reliable Data Transmission in Network-on-Chips

Ahmad Patooghy, Hamed Tabkhi, Seyed Ghassem Miremadi
Dependable Systems Laboratory, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran.
{patooghy, tabkhi}@ce.sharif.edu, miremadi@sharif.edu

***Abstract*— Data transmission in Network-on-Chips (NoCs) is a serious problem due to crosstalk faults happening in adjacent communication links. This paper proposes an efficient flow-control method to enhance the reliability of packet transmission in Network-on-Chips. The method investigates the opposite direction transitions appearing between flits of a packet to reorder the flits in the packet. Flits are reordered in a fixed-size window to reduce: 1) the probability of crosstalk occurrence, and 2) the total power consumed for packet delivery. The proposed flow-control method is evaluated by a VHDL-based simulator under different window sizes and various channel widths. Simulation results enable NoC designers to make a trade-off between window size, reliability and power consumption of packet delivery. This method is also compared with other crosstalk tolerant methods in terms of reliability and power consumption. Comparison results confirm that the method is a cost efficient solution to overcome the crosstalk problem.**

Keywords-network-on-chip; crosstalk; reliability; power

I. INTRODUCTION

Network-on-Chips (NoCs) [1][2] are highly sensitive to transient faults due to the use of nano-scale VLSI technologies in their fabrication [7][8]. Crosstalks [10][11], particle strikes [9], electro-magnetic interferences [31], and power supply disturbances [31] are the most important transient faults which affect correct functionality of NoCs. Among the mentioned faults, crosstalks have the major contribution in causing errors in NoCs [14][22]. Crosstalks happen because of coupling capacitances formed between adjacent wires of communication channels in NoCs. The coupling capacitances may result in unwanted transition on a victim wire when transitions appear on the neighboring wires of the victim wire [9][15]. Such coupling capacitances have negative impacts on delay, power consumption, and signal integrity of data transmission in NoCs [16].

High sensitivity of NoCs to crosstalk faults makes the reliability as one of the main concerns in the design of these products. In this regard, several approaches have been proposed in the literature to mitigate the effects of crosstalk faults in NoCs. At the lowest level of design abstraction, i.e., layout of the chip, specialized routing strategies [16] try to route wires in a way that minimizes the coupling capacitances between the adjacent wires. Inserting shield wires between wires of communication channels has been proposed at the layout level as well. Insertion of the shield wires reduces the rate of adjacent transitions in opposite directions; subsequently, the probability of crosstalk occurrence is lessened [18][19]. Although, layout level methods try to prevent crosstalk occurrence, these methods require modifications in the fabrication process of the chip. This has negative impacts on the design time and production cost of chips. At a higher level of design abstraction, intentionally skewing signal transition timings on adjacent wires has been proposed to reduce the delay effects of crosstalks. This method is only applicable to repeater-enabled communication channels [17].

At the RTL level, data coding is widely used to reduce the probability of crosstalk occurrence in on-chip networks. In a coding technique, n bits of data are mapped to k bits of code such that the probability of crosstalk occurrence for the coded data is lower than the original data. Delay reduction codes [20][21] and crosstalk-avoidance codes [7][22][33] are examples of the coding techniques trying to mitigate the effects of crosstalk faults. Data coding has also been used in flow-control methods to detect and to correct crosstalk faults. Flow-control methods can be applied in either end-to-end or switch-to-switch methods [3][4]. In an end-to-end method, the source node adds error detection codes e.g., parity or cyclic redundancy check, to each packet and the destination node checks the integrity of the packets. If an error is detected, the source node is requested to resend the packet. In a switch-to-switch method, the data correctness checking is performed whenever a flit (a packet is divided into fixed size units called flits) reaches the next node. If an erroneous flit is detected, a NACK signal is sent to the sender node to indicate that the received flit is incorrect. In this situation, the sender node stops sending the next flit and resends the requested flit.

At a higher level of design abstraction, flood-based algorithms tolerate transient faults (including crosstalks) using packet redundancy [12][13]. These algorithms are similar to the spreading of a rumor throughout a large group of friends such that whenever a node receives a new packet, it chooses a subset of its adjacent nodes and sends the packet to them. At the next round, the selected nodes which have already received the packet, spread it in the same manner. Destination node then will be able to use multiple copies of the same packet to overcome probable data errors occurring in the packet during the transmission.

Although several methods have been proposed to tackle the problem of crosstalk fault in on-chip network, to the best of our knowledge, all of the methods have deteriorations in at least two out of three important NoC parameters namely performance, power consumption, and area occupation. This paper proposes an efficient flow-control method to enhance the reliability of packet transmission in Network-on-Chips. The main advantage of the proposed method is that it simultaneously provides reliability enhancement in packet transmission as well as power reduction for packet delivery. Moreover, performance and area overheads of this method

978-0-7695-4171-6/10 $26.00 © 2010 IEEE
DOI 10.1109/DSD.2010.23

are negligible in comparison with other methods (see Section V). Since opposite direction transitions are among the main causes of crosstalk faults [22][21][23], the method investigates the opposite direction transitions between flits of a packet to reorder the flits of the packet. To do this, flits of a packet are divided into windows with a fixed-size length. Flits of each window are examined to find a sequence of flits which has the lowest probability of crosstalk occurrence. Simulation results confirm that the method not only reduces the probability of crosstalk occurrence, but also decreases the total power consumed for the packet delivery. Based on the simulation results, size of the reordering window can be determined by the NoC designer to adjust the amount of reliability improvement and/or power saving.

The rest of the paper is organized as follows. Section II discusses flow-control methods in NoCs. An analytical discussion regarding probability of crosstalk occurrence is presented in Section III. The proposed method is introduced and evaluated in Sections IV and V respectively. Finally section VI concludes the paper.

II. Flow-Control Methods

Flow-control methods are widely used to improve the reliability of packet delivery against transient faults (including crosstalk faults) [6][26]. Flow-control methods add information redundancy to the packets traversing the network and use the redundancy to guarantee the integrity of packets. Based on how often the integrity checking is done, flow-control methods are classified into Switch-to-Switch and End-to-End categories [3][4][6][26].

In the switch-to-switch methods, the sender node adds information redundancy e.g., parity or CRC bits to each flit of the packet and the receiver node checks the integrity of each receiving flit separately. After sending each flit, the sender switch keeps a copy of the flit in a retransmission buffer and waits until the receiving switch activates an ACK (or NACK) signal. If the ACK signal is received, the sender switch sends the next flit of the packet. Otherwise, the sender switch calls the transmitted flit from the corresponding retransmission buffer and resends the flit. The receiver switch checks the correctness of each newly received flit and sends ACK/NACK signal to the sender switch based on the result of the check.

Due to the frequent flit correctness checking, switch-to-switch methods have rather low latency in detecting errors. In other words, these methods do not allow the erroneous flits to propagate through the network. This is achieved by adding codec modules to all channels of NoC switches. Figure 1 shows foundation of a typical switch-to-switch flow-control method.

End-to-end flow-control methods are the other way to enhance the reliability of NoCs. In these methods the sender adds error detection codes to the packet (e.g., the last flit of the packet contains error detection code) and data integrity checking is performed for the whole packet whenever the last flit of the packet reaches the destination node; instead of checking flits separately in each intermediate node. In the case of detecting an erroneous packet by the destination node, a control packet (containing sequence number of the erroneous packet) is sent to the source node requesting to resend the packet. Although the end-to-end flow-control methods need hardware support as well, their required hardware is much lower than what required in the switch-to-switch methods. The end-to-end methods require an encoder and a decoder module per each switch of NoC; instead of one per each channel in the switch-to-switch methods. In addition, retransmission buffers are not required in the end-to-end methods, because it is reasonable to assume that the packet can be regenerated by the corresponding application in the source node.

Figure 2 shows the hardware support which is needed in a typical end-to-end flow-control method. Lower hardware requirement allows the end-to-end flow-control methods to impose lower power overhead to the network in comparison with the power overhead of switch-to-switch methods [5].

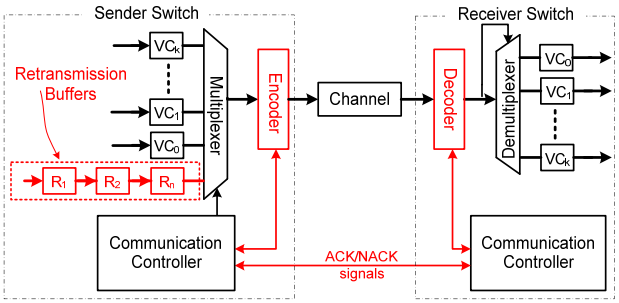

Figure 1. Foundation of a switch-to-switch flow-control method. VC_0 to VC_k and R_1 to R_n are flit buffers and retransmission buffers respectively.

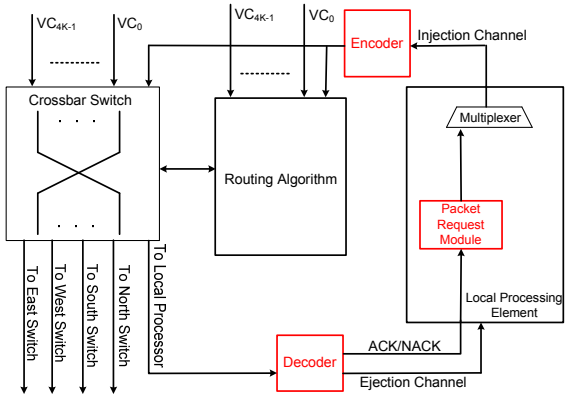

Figure 2. Foundation of a typical end-to-end flow-control method.

III. Crosstalk Analysis

This section provides the required background regarding crosstalk faults in on-chip communication channels. The analysis carried out in this section enable us to compare the proposed method with the previous crosstalk tolerant methods. For this purpose, the following notation is used to represent different types of transitions which may appear on a single bit communication channel. Symbols ↑ and ↓ are used to represent transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ respectively, and also symbols × and – refer to *don't care* and *no transitions*.

Investigations show that crosstalk fault occurs when the transitions appearing on wires of a communication channel inadvertently affect some other wires of the communication channel [10][11]. The affected wires which are called victim wires encounter either delay in their rising/falling edges or unwanted transitions [15][16]. To the best of our knowledge, the ↓↑ and ↑↓ transition patterns happening between the

adjacent wires are the most prevalent patterns which intensify the crosstalk fault [15][21][23]. However, the ↓↑↓, ↑↓↑, ↑−↑, and ↓−↓ patterns increase the probability of crosstalk occurrence, as well. In this way, researchers try to prevent or decrease the rate of such patterns to augment the immunity of communication channels toward crosstalk faults.

Crosstalk avoidance codes (CACs) [20][21][22] reduce the crosstalk probability by avoiding some specific patterns of transitions. Authors of [23] have shown that if a crosstalk tolerant method prevents all the patterns ↓↑×, −↑−, ↑−↑, and their complements, this method will minimize the delay effects of crosstalk faults. Duplicating wires of a communication channel as well as inserting shield wires between the duplicated wires is the simplest way to prevent the mentioned patterns of transitions. It has been shown that delay effects of crosstalk faults can be mitigated by preventing ↓↑ and ↑↓ transitions on two adjacent wires of a communication channel [15].

Opposite direction transitions, called OD transitions hereafter, can be eliminated by avoiding bit sequences '010' and '101' in all flits traversing the communication channel [20]. However, the complexity of codec modules of CACs rises dramatically when the width of communication channel grows [23]. Partial coding is proposed to tackle this problem; in this way, the communication channel is broken into several sub-channels with smaller widths. Each sub-channel is encoded separately and then the sub-channels are combined such that the probability of crosstalk occurrence at the boundaries of sub-channels is minimized [20][21].

Duplicate-add-parity [24][25] is proposed to reduce the probability crosstalk occurrence by duplicating each flit and adding a parity bit to the duplicated data. To do this, the communication channel should be expanded into double size of the flit width e.g., for flits with K bits width, the communication channel needs 2K+1 bits width. As shown in Figure 3, encoder of duplicate-add-parity makes a redundant copy for each flit and adds one-bit parity to the redundant flit. The original flit, redundant flit, and the parity bit are sent through the communication channel. In the decoder side, the parity bit is regenerated and compared with the received parity bit. The comparison determines which part of the received data should be stored and which part should be dropped. A close scrutiny in the duplicate-add-parity method reveals that although patterns ↑↓↑, ↑↓↑, −↑−, and ↑−↑ are prevented, the method is unable to eliminate or reduce ↑↓, ↓↑ patterns. That means the method is still vulnerable toward the crosstalk faults [15].

The boundary shift code scheme proposed in [15], attempts to reduce crosstalk-induced delay by avoiding a shared boundary between successive flits. This method is very similar to duplicate-add-parity method since it uses wire duplication and one parity bit to achieve crosstalk avoidance and single-error correction. However, the boundary shift code method places the parity bit on the opposite side of the double-width flit at each clock cycle. This is done to avoid dependent boundaries in subsequent flits.

After reviewing some previous crosstalk tolerant methods, let us to discuss the probability of crosstalk occurrence in an unprotected communication channel. We calculate the expected numbers of OD transitions i.e., ↓↑ and ↑↓, in a communication channel which has K bits width. It can simply be shown that the probabilities of other mentioned transition patterns, such as ↑↓↑ and ↑↓↑, have a direct relation to those of ↑↓ and ↓↑ patterns. Figure 4 shows all possible transitions appearing on a 2-bit communication channel when two consecutive flits f_0 and f_1 pass through the channel. Flits f_0 and f_1 have the width of 2 bits as well. As an example, in Figure 4.A flit f_0 is assumed to be '00' while flit f_1 can be any of its possible combinations. Transitions appearing on the channel in this situation are depicted in the right hand side of flit f_1 in Figure 4.A.

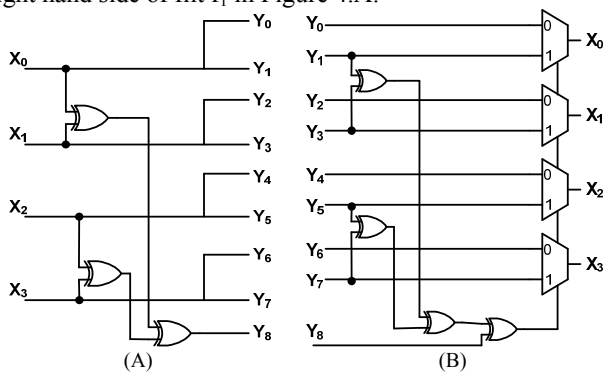

Figure 3. Encoder (A) and decoder (B) of the duplicate-add-parity method for an NoC with channel width of 9 bits.

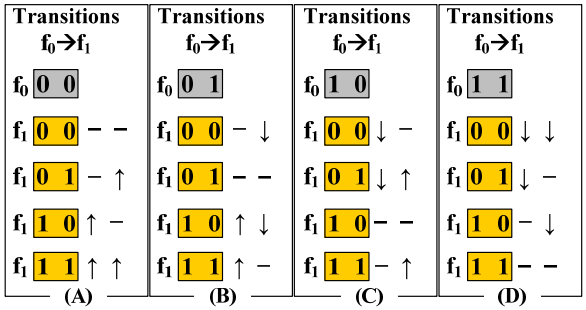

Figure 4. Transitions appearing on a 2-bit channel when flits f_0 and f_1 pass through the channel.

Considering the huge amount of data transferring in communication channels of an NoC, we can ignore the correlation between the flits f_0 and f_1 [15][32]. This means that the probability of a single bit in a flit being '0' is identical to the probability of being '1' i.e., $P_0 = P_1 = ½$. Table 1 shows how frequent the transition pairs of Figure 4 are. Some of the transition pairs shown in Table 1, e.g., I_5, I_7 may produce crosstalk faults since they lead to OD transitions on two adjacent wires of the communication channel.

Considering a communication channel with the width of K bits, there are still other possibilities which may cause crosstalk faults. For the sake of clarity, suppose transition pairs I_1 and I_8 occur in a 4-bit channel. In this situation the channel experiences the transition sequence '−↑↓↓' which has an OD transition. Generally, if one of the transition pairs $\{I_1, I_3, I_7\}$ appears at the left neighboring of one of the transition pairs $\{I_6, I_7, I_8\}$, the resulting transition sequences i.e., '×↑↓×' includes a pair of OD transitions. Similarly, the transition sequence '×↓↑×' can be seen if one of the transition pairs $\{I_4, I_5, I_8\}$ appears at the left neighboring of one of the transition pairs $\{I_2, I_3, I_5\}$. Transition sequences

'×↑↓×' and '×↓↑×' referred as S_1 and S_2 have the following probabilities of occurrence:

$$P_{S_1} = P(\{I_1, I_3, I_7\}) \times P(\{I_6, I_7, I_8\}) = \frac{4}{16} \times \frac{4}{16}$$

$$P_{S_2} = P(\{I_4, I_4, I_8\}) \times P(\{I_2, I_3, I_5\}) = \frac{4}{16} \times \frac{4}{16}$$

where $P(\{I_1, I_3, I_7\}) = P_{I_1} + P_{I_3} + P_{I_7}$ can be calculated according to Table 1.

TABLE 1. DIFFERENT TRANSITION PAIRS APPEARING ON A 2-BIT COMMUNICATION CHANNEL AND THEIR PROBABILITIS.

Symbol	Tran. pair	Freq.	Prob.	Symbol	Tran. pair	Freq	Prob.
I_0	− −	4	1/4	I_5	↑ ↓	1	1/16
I_1	− ↑	2	1/8	I_6	↓ −	2	1/8
I_2	↑ −	2	1/8	I_7	↓ ↑	1	1/16
I_3	↑ ↑	1	1/16	I_8	↓ ↓	1	1/16
I_4	− ↓	2	1/8	Total	---	16	1
Freq. = Frequency of occurrence				Prob. = Probability of occurrence			

As it is shown in Figure 5, there are $\frac{K}{2}$ transition pairs, and $\frac{K}{2} - 1$ boundary transition pairs for a communication channel with the width of K bits. In order to have no pair of OD transitions on the communication channel, the following conditions should be satisfied.

1) Transition pairs appearing on the communication channel are not allowed to be I_5 or I_7. Figure 5 shows the transition pairs appeared on the communication channel when K-bit flit f_1 follows K-bit flit f_0. According to Figure 5, transition pair TP_1 appears when the first two bits of f_0, X_1X_2 change to the first two bits of f_1, Y_1Y_2. Obviously, there are a total of $\frac{K}{2}$ transition pairs in a channel with the width of K bits.

2) Transition pairs appearing on the communication channel are not allowed to produce neither S_1 nor S_2 to prevent OD transitions in boundary transition pairs. In Figure 5, boundary transition pair b_i is made of right transition of the transition pair TP_i and the left transition of the transition pair TP_{i+1}.

Based on the abovementioned conditions, the probability of appearing no pair of OD transitions in a channel which has the width of K bits can be written as:

$$P^K(L=0) = \left[1 - \left(P_{I_5} + P_{I_7}\right)\right]^{K/2} \times \left(1 - P_{S_1}\right)^{K/2 - 1}.$$

The probability of appearing only one pair of OD transitions in the channel is:

$$P^K(L=1) = \binom{K/2}{1}\left(P_{I_5} + P_{I_7}\right) \times \left[1 - \left(P_{I_5} + P_{I_7}\right)\right]^{\frac{K}{2}-1} \times \left(1 - P_{S_1}\right)^{\frac{K}{2}-1} + \left[1 - \left(P_{I_5} + P_{I_7}\right)\right]^{\frac{K}{2}} \times \binom{\frac{K}{2}-1}{1} P_{S_1} \times \left(1 - P_{S_1}\right)^{\frac{K}{2}-2}$$

This equation considers that only one of transition pairs TP_1 to $TP_{K/2}$ is allowed to be I_5 or I_7, and none of the boundary transition pairs is allowed to be I_5 or I_7. Extending the above equation to the probability of appearing exactly i pairs of OD transitions in a channel which has the width of K bits, we have:

$$P^K(L=i) = \sum_{v=0}^{i} \left[\binom{K/2}{v}\left(P_{I_5} + P_{I_7}\right)^v \times \left[1 - \left(P_{I_5} + P_{I_7}\right)\right]^{\frac{K}{2}-v} \times \binom{\frac{K}{2}-1}{i-v} {P_{S_1}}^{i-v}\left(1 - P_{S_1}\right)^{\frac{K}{2}-i-1+v} \right]$$

where $i \leq \frac{K}{2}$.

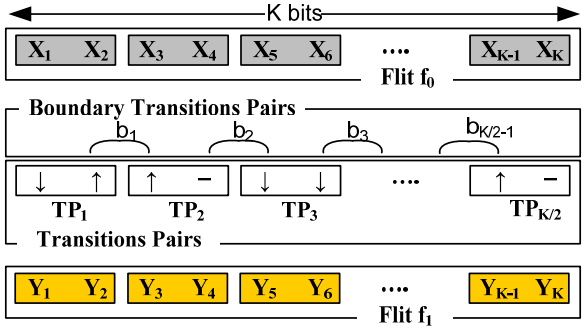

Figure 5. Transition pairs and boundary transitions appearing on a K-bit communication channel.

Finally, the expected number OD transitions in the channel is:

$$E^K(Crosstalk) = \sum_{i=0}^{K/2} i \times P^K(L=i)$$

$$= \sum_{i=0}^{K/2} i \sum_{v=0}^{i} \left[\binom{K/2}{v}\left(P_{I_5} + P_{I_7}\right)^v \times \left[1 - \left(P_{I_5} + P_{I_7}\right)\right]^{\frac{K}{2}-v} \times \binom{\frac{K}{2}-1}{i-v} {P_{S_1}}^{i-v}\left(1 - P_{S_1}\right)^{\frac{K}{2}-i-1+v} \right].$$

The above equation will help us to compare the expected number of OD transitions in an unprotected channel with a channel exploiting the proposed method.

IV. THE PROPOSED METHOD

This section presents the proposed flow-control method called Flit Reordering (FR) method.

As we discussed earlier, rate of crosstalk faults can be effectively decreased by preventing some specific transition patterns. The OD transitions, ↓↑ and ↑↓, are among these patterns [15][21][23]. The FR method reorders the flits of each newly generated packet to reduce the rate of opposite direction transitions appearing on communication channels of the NoC. Flit reordering is done at the time of injecting the packet into the network with at most a few cycles of delay.

In order to decrease the complexity of the FR method, the number of flits which can be reordered is limited to a fixed parameter i.e., window size. Encoder of the FR method divides flits of the packet into some non-overlapping windows. Each window is then separately examined to find a sequence of flits with minimum numbers of opposite direction transitions. Figure 6.A shows how flits of a packet are divided into *h* windows and Figure 6.B shows how the flits can be reordered by the FR encoder. For instance, in FR method the first flit of the first window (flit $f_{1,w1}$) is reordered as a last flit of the first window. Although the packet in Figure 6.A is divided into *h* windows of *n* flits, the FR

encoder supports the case that the last window contains less than *n* flits. The FR encoder adds tag bits into the reordered flits to enable the FR decoder to restore the original order of flits in each window. The reordered packet will be injected into the NoC through the injection channel of the sender node. From the NoC point of view, this is an ordinary packet which will be delivered to its corresponding destination. At the destination node, the decoder of FR method uses the tag bits and rearranges the flits to recover the original packet. Obviously the same window sizes are used in both the encoder and decoder sides.

In order to minimize the numbers of opposite direction transitions, the FR encoder considers the last flit of the window *i* when it is reordering flits of the window *i+1*. This is done by the means of an extra flit buffer namely Previously Sent Flit (PSF) buffer which is added into the architecture of the FR encoder. Figure 7 represents the block diagram of FR encoder. The FR encoder utilizes *n* flit buffers, *n* dirty bit flip-flops, *n* OD transitions detector modules, a PSF buffer, and a Min. distance detector module where *n* is the size of reordering window.

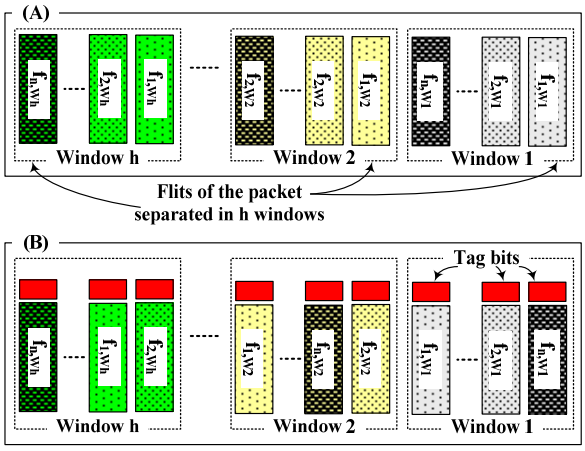

Figure 6. Flits of the packet are divided into h window (A), then flits each window are reordered separatly (B).

In each round of competition, flits with dirty bits of '0' are examined to find the flit which has the lowest number of OD transitions with respect to the previously sent flit. Such a flit so called winner flit, is chosen by the Min. distance detector module to inject into the network at the next cycle. After selecting the winner flit, following tasks should be done: 1) the winner flit is injected into the network, 2) a copy of the winner flit is sent to the PSF buffer for the next round of competition, and 3) dirty bit of the winner flit is set to '1' to stop the winner flit contributing in the next competition.

Flits of the packet are loaded into the buffers of FR encoder in a pipeline fashion to minimize the performance overhead of the FR encoder. In this way, lowest performance overhead which is *n* cycles delay for an encoder with the window size of *n* flits is achieved. In order to separate flits of consecutive windows, flits of the next window are loaded into the FR encoder with dirty bits of '1'. When all *n* flits of the next window are loaded into the FR buffers, Next Window Detector module, referred in Figure 7 as NW detector, resets all *n* dirty bits at the same time. The next window is begun to process by the FR encoder at this time.

In this way, the FR method minimizes the numbers of OD transitions which in turn reduces the probability of crosstalk occurrence. Since the flit reordering is done in the source and destination nodes i.e. the intermediate nodes do not contribute the reordering, the FR method is an end-to-end flow-control method. As we discussed earlier, in the end-to-end method one codes module is needed per each node of the NoC. This means that the hardware overhead of the FR method is very low (for more details please see Section V).

It should be noted that the FR method requires some additional wires in the communication channel to send the tag bits of flits. Numbers of these additional wires is equal to $\log_2(n)$ where *n* is the size of reordering window.

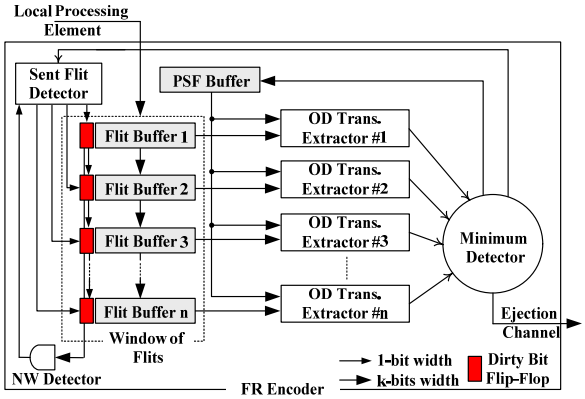

Figure 7. Block diagram of the encoder of FR method.

V. Evaluation of The Proposed Method

A. Analytical Evaluation

In this section we calculate the expected value of the number of OD transitions in an NoC exploiting the FR method. The expected value of the numbers of OD transitions of an unprotected NoC is compared with an NoC using FR method. The comparison reveals that the FR method reduces the rate of OD transitions. For the sake of fairness, this analysis uses the same assumptions made in Section III. As discussed in Section III flits are assumed to be uncorrelated which is a common assumption in the literature [15][32]. For the sake of simplicity, it has been assumed that the flits are still uncorrelated after adding tag bits. The analysis is carried out for a channel with the width of K bits and window size of *n* flits.

Let L be the random variable of OD transitions appearing between two consecutive flits. The probability of having L=*l* OD transitions in a channel with the width of K bits is calculated in Section III:

$$P^K(L=l)=\sum_{v=0}^{l}\left[\binom{K/2}{v}\left(P_{I_5}+P_{I_7}\right)^v\times\left[1-\left(P_{I_5}+P_{I_7}\right)\right]^{\frac{K}{2}-v}\times\binom{\frac{K}{2}-1}{l-v}P_{S_1}{}^{l-v}\left(1-P_{S_1}\right)^{\frac{K}{2}-l+v}\right].$$

According to the FR method, at the first round of competition the numbers of OD transitions between the previously sent flit and *n* flits loaded in the FR encoder are calculated. These *n* values follow random variables $L_1, L_2, \ldots, L_n$ which all have the same distribution i.e., $P^K(L=l)$. Since the flit with the lowest OD transitions is

selected as the winner flit, number of OD transitions appearing on the channel at the first round of competition follows a random variable L^1_{min} which is defined as:

$$L^1_{min} = Min(L_1, L_2, \dots, L_n).$$

At the second round of competition, the winner flit has passed the FR encoder, so *n-1* remaining flits of the window contribute in the competition. Using similar reasoning, number of OD transitions appearing on the channel at the second round of competition follows a random variable L^2_{min} which is defined as:

$$L^2_{min} = Min(L_1, L_2, \dots, L_{n-1})$$

and generally, number of OD transitions appearing on the channel at the i^{th} round of competition is a random variable L^i_{min}:

$$L^i_{min} = Min(L_1, L_2, \dots, L_{n-i+1})$$

where $1 \le i \le n$.

Probability of having less than *l* OD transitions at the i^{th} round of competition can be calculated as:

$$P^K_i(L^i_{min} < l) = P^K_i(Min(L_1, L_2, \dots, L_{n-i+1}) < l)$$
$$= \bar{P}^K_i(Min(L_1, L_2, \dots, L_{n-i+1}) > l)$$
$$= 1 - \big(P^K(L_1 > l) \times P^K(L_2 > l) \times \dots \times P^K(L_{n-i+1} > l)\big).$$

Since we assumed that there is no correlation between flits of a window, $P^K(L_q > l) = P^K(L > l)$ for every q, so $P^K_i(L^i_{min} < l)$ can be simplified to:

$$P^K_i(L^i_{min} < l) = 1 - \big(1 - P^K(L < l)\big)^{n-i+1}.$$

Using the above probability accumulative function, the probability of having exactly *l* OD transitions on the communication channel at the i^{th} round of competition can be calculate by:

$$P^K_i\big(L^i_{min} = l\big) = P^K_i\big(L^i_{min} \le l\big) - P^K_i(L^i_{min} < l).$$

The above equation can be used in extracting the expected number of OD transitions at the i^{th} round of competition as follows:

$$E^K_i(L^i_{min}) = \sum_{j=0}^{K} j \times P^K_i\big(L^i_{min} = j\big).$$

Finally, the average number of OD transitions appeared on the communication channel when all *n* flits have passed the FR encoder can be calculated as:

$$E^K_{FR}(H.D) = \frac{1}{n}\sum_{i=1}^{n} E^K_i(L^i_{min}).$$

Using the above discussion, an unprotected channel is compared with a channel equipped with the FR method in terms of average number of OD transitions, number of overhead wires and delay. Comparisons are done under two constraints 1) fixed flit size and 2) fixed channel size.

Table 2 compares the two mentioned communication channels under the constraint of fixed flit size. In this condition, both of the channels send flits with the same width. Based on Table 2, when the flit width is 8 bits, the FR method reduces the transitions by 9.75, 21.25, and 33.25 percents for the window sizes of 2, 4, and 8 flits respectively. But in this condition the FR method requires 1, 2, and 3 additional wires to send the tag bits for the window sizes of 2, 4, and 8 flits respectively. Generally, numbers of additional wires that the FR method requires to send the tag bits is $\log_2(n)$ where *n* is the size of the window. Since the reordering window is loaded by the flits in a pipeline fashion, the performance overhead of the FR method under the fixed flit size constraint is 2, 4, and 8 cycles respectively. Generally, the imposed performance overhead is equal to the window size under the fixed flit size constraint.

Table 3 shows the results of comparison done under the fixed channel size constraint where the size of data in each flit is reduced to provide some bits for the tag bits. For example, in the channel with width of 9 bits and window size of 2 flits, the size of data in each flit is equal to 8 bits. When the window size increases to 4 flits, the size of data is limited to just 7 bits. As shown in Table 3, the fixed size channel constraint imposes a performance overhead to the NoC for every flit sent through the channel. The performance overhead is equal to the number of wires dedicated for sending the tag bits. For instance, in a communication channel with width of 19 bits and the window size of 8 flits, the performance overhead is equal to 3 bits for each flit which is given by $(3/19)\,x$ in Table 3; where x is the number of flits in the packet. On the other hand, in both fixed flit size and fixed channel size constraints, an additional performance overhead is imposed to the system. This overhead is the result of the delay of sending the first flits of each packet and is equal to the size of window.

B. Experimental Evaluation

In order to experimentally evaluate the proposed flow-control method, a VHDL-based simulator is developed. The simulator composed of three modules including an FR encoder, an FR decoder, and a random flit generator. The random flit generator module generates flits according the simulation parameters e.g., flit size and amount of data. The FR encoder receives the flits from the random flit generator module and reorders the flits based on the size of reordering window. Finally, the FR decoder module reorders the flits and counts the number of opposite direction transitions as well as number of transitions. A synthesizable version of the simulator is used to investigate the power consumption, area, and performance overheads of the proposed method. To do this, Design Compiler tool is utilized to extract the overheads of the proposed method which is synthesized in 65nm technology size. Simulation experiments are done for different window sizes and different flit widths. In each simulation experiment, 5MB of data has been generated by the flit generator module and reordered by the FR encoder module. Results of the simulations confirm those obtained from the analytical modeling of the FR method.

Table 4 shows the number of transitions and opposite direction transitions in 16- and 32-bit communication channels. In Table 4, W=0 is used to refer the unprotected channel while W=2 and W=4 refer to the channels exploiting the FR method with the window sizes of 2 and 4 flits. Based on the experimental results, when the size of reordering window increases, more reduction in the number of transitions and number of opposite direction transitions are achieved in both 16- and 32-bit channels. This means that the FR method gives more reliability improvement in larger window sizes. The same behavior is seen when the channel width increases from 16 to 32 bits.

Since the communication channels consume 20%–36% of total power in most of NoCs [24], reducing transitions on the communication channels directly reduces the power dissipation of NoCs. In addition, more power saving is achieved by the FR method in larger window sizes and/or wider communication channels.

TABLE 2. IMPROVMENTS AND OVERHEADS OF T`HE FR METHOD UNDER FIXED FLIT SIZE CONSTRAINT.

Flit size	Unprotected channel	Channel protected by FR											
	Expected transitions	Expected transitions in window of W-flits			Percentage of transition reduction			Number of overhead wires			Performance overhead (cycles)		
		W=2	W=4	W=8	W=2	W=4	W=8	W=2	W=4	W=8	W=2	W=4	W=8
8	4	3.61	3.15	2.67	9.75	21.25	33.25	1 bit	2 bits	3 bits	2 cycles	4 cycles	8 cycles
16	8	7.03	5.86	4.94	12.1	26.7	38.2						
32	16	13.73	10.96	8.94	14.2	31.5	44.1						

TABLE 3. IMPROVMENTS AND OVERHEADS OF THE FR METHOD UNDER FIXED CHANNEL SIZE CONSTRAINT.

Channel size	Unprotected channel	Channel protected by FR								
	Expected transitions	Expected transitions in window of W-flits			Number of overhead wires			Performance overhead (cycles)		
		W=2	W=4	W=8	W=2	W=4	W=8	W=2	W=4	W=8
9	4.5	3.15			--	--	--	$2+\frac{1}{9}x$		
10	5		2.67						$4+\frac{2}{10}x$	
11	5.5			9.75						$8+\frac{3}{11}x$
17	8.5	5.86						$2+\frac{1}{17}x$		
18	9		4.94						$4+\frac{2}{18}x$	
19	9.5			12.1						$8+\frac{3}{19}x$
33	16.5	13.73						$2+\frac{1}{33}x$		
34	17		10.96						$4+\frac{2}{34}x$	
35	17.5			8.94						$8+\frac{3}{35}x$
								parameter x is the total number of flits per packet		

TABLE 4. IMPROVEMENTS OF THE FR METHOD IN 16- AND 32-BIT COMMUNICATION CHANNELS.

Flit size	Number of transitions			Transition reduction (%)			Power saving due to transition reduction (mW)			Opposite direction transitions reduction (%)		
	W=1	W=2	W=4	W=1	W=2	W=4	W=1	W=2	W=4	W=1	W=2	W=4
16	21113314	18261772	16855570	--	13.5	20.2	--	5.78	8.63	--	40.6	62.9
32	20588526	16750706	14848956	--	18.6	27.9	--	7.78	11.6	--	45.3	69.2

TABLE 5. POWER CONSUMPTION AND AREA OVERHEADS OF THE FR ENCODER.

Channel size	Power consumption (μW)			Area occupation (μm^2)		
	W=1	W=2	W=4	W=1	W=2	W=4
16	19.0	77.7	175.7	253.4	1171.4	3803.8
32	37.6	180.6	428.2	506.9	1524.6	8850.6

TABLE 6. THE FR METHOD IN COMPARISON WITH THE DUPLICATE-ADD-PARITY.

Flit size	Transition patterns ↓↑, ↑↓				Reduction of transition patterns ↓↑, ↑↓ (%)				Transition patterns ↑↓↑, ↑↓↑, −↑−, ↑−↑				Reduction of transition patterns ↑↓↑, ↑↓↑, −↑−, ↑−↑ (%)			
	W=1	W=2	W=4	DAP	W=1	W=2	W=4	DAP	W=1	W=2	W=4	DAP	W=1	W=2	W=4	DAP
16	8281237	4921862	3076158	8281237	0	40.6	62.9	0	6881512	3221862	2053161	0	0	53.2	70.2	100
32	9101424	4980364	2805099	9101424	0	45.3	69.2	0	8250457	3692553	1807615	0	0	55.2	78.1	100

Table 5 represents the power and area overhead of the FR encoder for various window sizes and channel widths. The overheads of the FR method are extracted by the use of a synthesizable version of the simulator which is synthesized in 65nm technology size. Table 5 reveals that the overheads of the FR method are negligible in comparison with its improvements.

In the next experiment the FR method is compared with the duplicate-add-parity method [24][25] which is designed to prevent transition patterns ↑↓↑, ↑↓↑, −↑−, and ↑−↑. To do this, the duplicate-add-parity method is also simulated by a VHDL model. Table 6 compares the reductions of the FR and duplicate-add-parity methods with respect to the both set of transition patterns {↓↑, ↑↓} and {↑↓↑, −↑−, ↑−↑}.

Since the duplicate-add-parity method does not consider the transition patterns ↓↑, ↑↓, it has no reduction in the number of such transition patterns. In contrast, the FR method has a noticeable reduction in the number of transition patterns ↑↓↑, −↑−, ↑−↑ while this method does not consider such patterns.

VI. CONCLUSIONS

This paper proposed an efficient flow-control method to enhance the reliability of packet transmission in Network-on-Chips. The method investigates the opposite direction transitions appearing between flits of a packet to reorder the flits in the packet. The main advantage of the proposed method is that it simultaneously provides reliability enhancement of packet transmission as well as power reduction for packet delivery. Simulations which were done for various working conditions confirmed that the proposed method significantly improves the crosstalk immunity and power consumption at the same time. The analytical discussion presented in the paper clears how different parameters such as window size and channel width impact the reliability improvement of the proposed method. Results obtained from the analytical modeling were validated by those obtained from the experimental simulations.

REFERENCES

[1] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, “A Network on Chip Architecture and Design Methodology”, Proceedings of ISVLSI, pp. 117-122, April 2002.

[2] L. Benini, and G. De Micheli, “Networks on Chips: A New SoC Paradigm”, IEEE Computers, Vol. 35(1), pp. 70-78, Jan. 2002.

[3] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, G. De Micheli, "Analysis of Error Recovery Schemes for Networks-on-Chips", IEEE Design and Test of Computers, Vol. 22(5), pp. 434-442, Sep-Oct 2005.

[4] D. Bertozzi, D. L. Benini, G. De Micheli, "Low Power Error-Resilient Encoding for On-Chip Data Buses", Proceedings of DATE, pp. 102-109, March 2002.

[5] A. Patooghy, M. Fazeli, S. G. Miremadi, "A Low-Power and SEU-Tolerant Switch Architecture for Network on Chips", Proceedings of the IEEE/IFIP Pacific Rim International Symposium on Dependable Computing (PRDC 2007), Melbourne, Victoria, Australia, Dec. 2007.

[6] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, C.R. Das, “Exploring Fault-Tolerant Network-on-Chip Architectures”, International Conference on Dependable Systems and Networks (DSN) 2006, pp. 93.

[7] R. Hegde and N.R. Shanbhag, “Towards Achieving Energy Efficiency in Presence of Deep Submicron Noise”, IEEE Transactions on VLSI Systems, vol. 8, no. 4, pp. 379-391, Aug. 2000.

[8] S. Murali, D. Atienza, L. Benini, and G. De Micheli, “A Multipath Routing Strategy with Guaranteed in-order Packet Delivery and Fault-Tolerance for Networks on Chip”, Proceedings of the 43rd ACM/IEEE Design Automation Conference (DAC ’06), pp. 845–848, San Francisco, Calif, USA, July 2006.

[9] A. P. Frantz, L. Carro, É. F. Cota, F. L. Kastensmidt, “Evaluating SEU and Crosstalk Effects in Network-on-Chip Routers”, IOLTS, pp. 191-192, 2006.

[10] M. H. Tehranipour, N. Ahmed, M. Nourani, “Testing SoC Interconnects for Signal Integrity Using Boundary Scan”, VTS, pp. 158-172, 2003.

[11] H. Zimmer, and A. Jantsch, "A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip", Proceedings of ISSS/CODES, pp. 188- 193, Sept 2003.

[12] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-Chip Fault-Tolerant Communication", Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 225-232, 2003.

[13] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, “Fault Tolerant Algorithms for Network-on-Chip Interconnect”, Proceedings of the ISVLSI, 2004.

[14] M. Kuhlmann, and S.S. Sapatnekar, “Exact and Efficient Crosstalk Estimation”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 20(7), pp 858-866, 2001.

[15] K. N. Patel, and I. L. Markov, “Error-Correction and Crosstalk Avoidance in DSM Busses”, IEEE Transactions on Very Large Scale Integration (VLSI), Vol. 12 , pp 1076-1080, 2004.

[16] T. Gao, and C. L. Liu, “Minimum Crosstalk Channel Routing”, Proceedings of International Conference on Computer-Aided Design (ICCAD), pp. 692–696, Nov. 1999.

[17] K. Hirose, and H. Yasuura, “A Bus Delay Reduction Technique Considering Crosstalk”, Proceedings of Design, Automation and Test Europe (DATE), pp. 441–445, 2000.

[18] H. Kaul, D. Sylvester, D. Blaauw, “Active shields: a new approach to shielding global wires”, Proceedings of Great Lakes Symposium on Very Large Scale Integration (GLS-VLSI), pp. 112–117, Apr. 2002.

[19] K. M. Lepak, I. Luwandi, L. He, “Simultaneous Shield Insertion and Net Ordering under Explicit RLC Noise Constraint”, Proceedings of Design Automation Conference (DAC), pp. 199–202, June 2001.

[20] C. Duan, A. Tirumala, S. P. Khatri, “Analysis and Avoidance of Cross-Talk in on-Chip Buses”, Hot Interconnects 9, pp. 133–138, Aug. 2001.

[21] B. Victor, and K. Keutzer, “Bus Encoding to Prevent Crosstalk Delay”, Proceedings of International Conference on Computer-Aided Design (ICCAD), pp. 57–69, Nov. 2001.

[22] D. Bertozzi, L. Benini, G. D. Micheli, “Low Power Error Resilient Encoding for on-Chip Data Buses”, Proceedings of DATE, pp. 102–109, 2002.

[23] S. R. Sridhara, and N. R. Shanbhag, “Coding for Reliable On-Chip Buses: A Class of Fundamental Bounds and Practical Codes”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 26, Issue 5, pp. 977 – 982, May 2007.

[24] S. R. Sridhara, and N. R. Shanbhag, “Coding for System-on-Chip Networks: A Unified Framework”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, pp. 655-667, June 2005.

[25] D. Rossi, A. K. Nieuwland, A. Katoch, C. Metra, "New ECC for Crosstalk Impact Minimization", IEEE Design and Test of Computers, vol. 22, no. 4, pp. 340-348, July/Aug. 2005.

[26] P. P. Pande, A. Ganguly, B. Feero, B. Belzer, C. Grecu, “Design of Low power & Reliable Networks on Chip Through Joint Crosstalk Avoidance and Forward Error Correction Coding”, IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'06), pp.466-476, 2006.

[27] F. J. MacWilliams, and N. J. A. Sloane, “The Theory of Error-Correcting Codes”, New York, North-Holland, 1977.

[28] M. Pedram, and J. Rabaey, “Power Aware Design ethodologies”, Norwell, MA: Klouwer, 2002.

[29] C. Svensson, “Optimum Voltage Swing on on-Chip and off-Chip Interconnect”, IEEE Journal of Solid-State Circuits, vol.36, no.7, pp.1108-1112, July 2001.

[30] A. Jantsch, R. Lauter, A. Vitkowski, “Power Analysis of Link Level and End-to-End Data Protection on Networks on Chip”, Proceedings of the IEEE International Symposium on Circuits and Systems, 2005.

[31] V. Raghunathan, M. B. Srivastava, R. K. Gupta, “Energy-Aware System Design: A Survey of Techniques for Energy Efficient On-Chip Communication”, Design Automation Conference (DAC), pp. 900–905, 2003.

[32] M. R. Stan and W. P. Burleson, “Bus-Invert coding for Low-Power I/O,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[33] A. Ganguly, P. P. Pande, B. Belzer, “Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, , Vol. 17, No. 11., March 2009, pp. 1626-1639.