A Reconfigurable Switch Architecture to Enhance Reliability of Network-on-Chips

Z. Shirmohammadi, M. Jalal, A. Patooghy, S. G. Miremadi
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
shirmohammadi@kish.sharif.edu, mjalal@ce.sharif.edu, patooghy@ce.sharif.edu, miremadi@sharif.edu

***Abstract*–Switches and communication links of Network on Chips (NoCs) are highly vulnerable to transient faults due to the use of nano-scale VLSI technologies in fabrication of NoCs. This paper proposes a reconfigurable switch architecture which is capable of operating in four configurations with different levels of reliability. This is done by the use of a local configuration controller logic which is fully protected against transient faults. When a controller detects a high error rate situation, it configures the switch to a high reliability mode and vice versa. Reconfiguration policy is designed in a way which minimizes the imposed performance and power overheads to the switch. Evaluations are done by a cycle accurate NoC simulator with Orion patch for power estimation. Simulation results show a noticeable reliability improvement with a negligible performance overhead. In addition, power saving of at least 20% is achieved by the proposed architecture.**

I. INTRODUCTION

Recent advances in VLSI technology enable chip designers to integrate a huge numbers of transistors on a single silicon die [1]. System-on-Chip (SoC) which is a product of the technology shrinkage provides many advantages such as reducing time to market, solving the challenges of telecommunication, multimedia and consumer electronic domains. However, SoCs suffer from performance and/or scalability limitations of the traditional communication architectures such as bus based architectures [2]. Network-on-Chip (NoC) has been emerged as a viable communication architecture for SoCs. In NoC paradigm, every on-chip modules employs a switching element and some communication channels to transmit packetized data to other on-chip modules. The switching element determines the way that packets visit intermediate nodes during their path toward destinations. NoCs are desired to provide as higher performance as possible to minimize the packet delivery time [3]. They are also required to consume as low energy as possible, because higher energy consumption reduces the mission duration (for battery-operated systems), increases temperature which may cause chip damage, and decreases reliability [4].

Use of nano-scale VLSI technologies in fabrication of NoCs places serious concerns in the correct functionality of these products [5]. NoCs are highly sensitive to transient faults, mostly including crosstalk and single/multiple event upsets [6]. Crosstalk faults happen between adjacent wires of an NoC communication channel because of the coupling capacitances formed between adjacent wires [7][8]. In crosstalk conditions unwanted transition, unwanted voltage glitch, or delayed edge appear on some wires so-called victim wires [9]. Each of these malfunctions may result in erroneous packet delivery. When a high energy particle strikes a sensitive region of a memory element (such as latch or flip-flop), a single or multiple event upset (SEU and MBU respectively) may occur [10]. Particle strikes produce unwanted voltage glitch in the internal circuit of the memory element, which in turn results in unwanted bit flips.

In order to tolerate transient faults occurring in NoCs, flow-control methods which are among the widely used ways can be utilized in the network [6][9][11]. They can be applied in either end-to-end or switch-to-switch manners. In end-to-end flow-control methods [6][11] the source node adds error detection codes to each outgoing packet. On the other side, the destination node checks the correctness of incoming packets. In switch-to-switch methods, data correctness checking is performed for every flit (each packet is divided into a sequence of fixed size units, called flit) by every receiving node, regardless of whether the receiving node is destination or not. The most serious drawback of the flow-control methods is that the performance and the power consumption of these methods vary as either error rate or traffic generation rate vary [12] This means that, overheads of flow-control methods is negligible in some working conditions, while their overheads are not acceptable in some other working conditions. As an example, frequent flit correctness checking done in switch-to-switch flow-control methods is not reasonable when the NoC is in a low error rate situation. The encoder/decoder modules dissipate a noticeable power while their reliability improvement is low.

Since most of the recent NoCs are intended to run wide range of applications rather than a single application [13][14] they should be able to adapt their functionality with the running application as well as the working condition [14]. In this regard, this paper proposes a reconfigurable switch architecture which is capable to provide different levels of reliability depending on the running application and working condition. The reconfiguration policy is designed in a way which minimizes the overall performance and power consumption overheads of the reliability improvement mechanisms. Evaluation experiments show a noticeable reliability improvement and power saving of at least 20% with a negligible performance overhead. Simulations experiments are performed using Xmulator and analyzing the results show us the enhancement of reliability, with minimizing the overheads of power and performance. Also an analytical Markov model is presented to estimate the overall reliability, performance and power of the proposed switch architecture. Results of the model show that the proposed switch is an efficient solution to improve the reliability of NoCs. In addition, the proposed model

provides a significant speed up in the evaluation phase of NoC design.

The rest of the paper is organized as follow. An overview on Network-on-Chips and their threats is presented in Section II. Section III discusses the related work. The proposed switch architecture is presented in Section IV and evaluated in Section V, finally conclusion remarks are given in Section VI.

II. BACKGROUND

In this section we describe the concept of routing algorithm, switching methodology and network topology. After that, the assumed switch architecture which is used as the foundation of the proposed switch architecture is described. At the rest of this section most important NoC faults e.g., crosstalk faults and SEU/MBU faults and their mitigation techniques are presented.

NoC Overview

Let us firstly clear important concepts of the NoC context. Topology defines how the nodes (on-chip modules) are interconnected in the chip. Mesh and hypercube are two most widely used topologies which are addressed by several researches and manufacturers due to their layout efficiency, good electrical properties and simplicity in addressing on-chip resources [15].

When a packet header reaches an intermediate node, the *switching method* determines how and when the switch is set; that is the way that the input channel is connected to the output channel. Wormhole is a widely used switching method [16], due to its low buffering requirements and more importantly, because it makes average packet delivery time almost independent of the distance between source and destination nodes. In wormhole switching, a packet is divided into a sequence of fixed-size units, called flits. The header flit (containing routing information) establishes a path through the network while the remaining data flits follow it in a pipelined fashion.

Figure 1 shows a NoC node architecture which can be utilized in mesh-based NoCs. The switching element of the node consists of 5 physical channels to communicate with its north, south, east and west neighbors, and the processing element.

In order to increase the performance of the network, each physical channel is timely multiplexed between some buffers, namely virtual channels [16]. A virtual channel associated to a physical channel shares the bandwidth of the physical channel with other virtual channels of that physical channel (in figure 1 each physical channel is shared between two virtual channels).

When a header flit arrives at an input virtual channel of a switch from either the previous node or the processing element connected to the same switch, the *router* decides the packet's destination direction (north, south, east, west or the processing element), and configures the crossbar switch by sending appropriate signals to the crossbar. Crossbar switch then connects the flit's incoming virtual channel to the selected outgoing one. In the case of existence of a free virtual channel in the selected outgoing physical channel, the header flit will be transferred to the next node and the other flits in the packet follow it in a pipelined fashion, otherwise the header flit has to wait until a virtual channel of the selected outgoing physical channel becomes free.

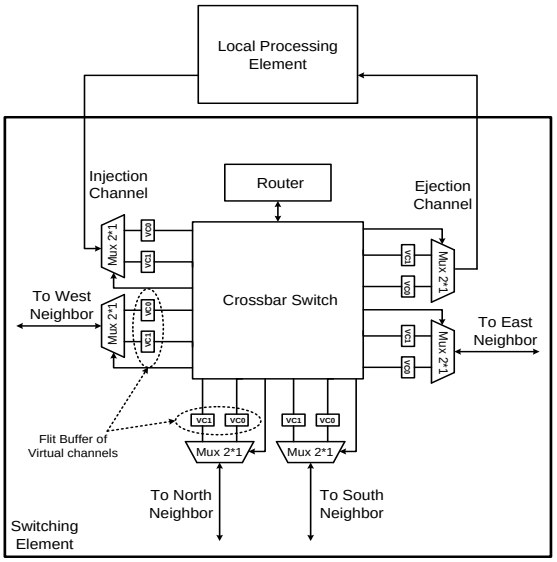

Figure 1. A mesh based NoC node architecture.

Crosstalk Faults in NoCs

Crosstalk fault is among the main reliability concerns in NoCs. The crosstalk happens between adjacent wires of a communication channel because of coupling capacitances formed between wires [7]. Nano-scale VLSI technologies used in the fabrication of NoCs produces closer wires which in turn increases the coupling capacitances of wires.

In crosstalk conditions, changing the logic of the adjacent links can affect each other. In these cases unwanted transitions in aggressive wires can force the victim wires to change the logic or even the rising transition on a wire can cause the neighborhood wires to delay or speed-up in rising or [9] falling. Several techniques have been proposed in the literature to prevent/mitigate the crosstalk faults. Inserting shield wires is one of the simplest ways to do this. In this technique, permanently ground wires are inserted between the main wires of the communication channel to reduce the coupling capacitances between the wires [17]. Crosstalk Avoidance Codes (CACs) are the other way, they try to prevent some problematic transition patterns on the communication channel such as ↓↑↓, ↑↓↑, ↑−↑, and ↓−↓ patterns (↓, ↑, and – respectively denote '1' to '0', '0' to '1' and no transitions on a wire). Duplicate-add-parity (DAP) is one of these codes, which reduces the crosstalk effect with duplicating each wire of the channel and adding one bit parity to the channel [18]. In this way the hamming distance between two consecutive flits increases to 3. The similar idea is used in the Modified Dual Rail (MDR)[17], and the Boundary Shift Code (BSC) techniques. In MDR technique, for a K bit channel K+1 redundant wires are utilized [17]. The Boundary shift code similar to the DAP, calculates one bit parity for the duplicated channel. But unlike the DAP, the parity bit is alternatively located in the left or right side of the channel in each cycle of clock signal.

Single/Multiple Event Upsets

As the technology size shrinks bellow 100 nanometers, chips become more susceptible to Alpha and Neutron particles [19]. Energy provided by these particles may exceed the electrical threshold that the circuit can tolerate. In such cases, an unwanted bit flip is seen in the logical value stored in the memory element which is called Single Event Upset (SEU). If the number of affected memory cells is more than one, the Multiple Bit Upset (MBU) is occurred. Information redundancy (error detection and correction codes) can be added to the packet traversing NoCs to improve the reliability of packet delivery [7]. Codes can be applied in either switch level or routing level. While switch level methods have lower error detection latency due to their hardware support, routing level methods have lower performance and power consumption overheads.

Robust memory cells which have circuit level redundancies are the other way to enhance the reliability of NoCs against particle strikes [20][21]. Time redundancy can be also used to reach this aim [22] e.g., several samples of a data are voted using delays and voting circuitries; hence the error can be masked.

Reliable routing algorithms utilize redundant packets or redundant paths to improve the reliability of packet delivery. As an example, flood-based fault-tolerant routing algorithms send redundant packets similar to the spreading of a rumor within a large group of friends [20][21]. Whenever a node of NoC receives a new packet, it chooses a subset of its adjacent nodes and sends the packet to them. Destination node then receives multiple copies of a same packet and uses the packet redundancy to tolerate probable data errors occurred to the packet during transmission. Reliable routing algorithms can cope with almost all transient faults (including SEU/MBU).

III. RELATED WORK

Previous sections clear that all of the reliability improvement methods require redundancies to reach their aims. The used redundancy can be in form of information, hardware, time, or combination of these. However, all types of redundancies have negative impact on the performance and/or power of NoC. In this regard, *reconfiguration* can be the proper choice used to minimize the overheads of the reliability improvement methods in NoCs.

Reconfigurable switch architecture has been proposed in [23] which can adaptively control network bandwidth. The switch can incorporate various IPs with different bandwidth requirements in a complex SoC design. According to [23], an additional bus with the crossbar is added to the switch, when the rate of the incoming packet is higher than the rate of the outgoing packet, the bus scheduler detects the congestions and uses the additional bus.

In [13], architecture of a run-time reconfigurable NoC with intelligent nodes is described. In this architecture the network configures itself in the term of routing and switching to maintain the quality of service. Authors of [24] have proposed an environment, called GEZEL, for implementation of a reconfigurable communication network for NoCs. In [14], a fast framework for Virtual Architecture for flexible partial time reconfiguration systems is inspected. In this framework, one can has flexible PTRs that can be used as core in NoC based SoC emulation. An approach to the design space exploration of a configurable NoC is proposed in [25]. In [26], with respect to the essential role of calculation of the router performance in early design phases, analytical model is proposed to improve NoCs functionality. In this paper, NoC routers, NoCs queues, and input/output buffers are inspected. The Markov chain analysis in this paper for mesh-based input queue routers shows that the changing the size of queue improves packet lost probability.

Although several research paper have been proposed in the literature to deal with the concept of reconfiguration in NoCs, to the best of our knowledge there is no work which utilizes the reconfiguration to achieve high level of reliability with low performance and power consumption overhead. By the means of reconfiguration, reliability improvement modules can be put in the clock-gate [27] mode when NoC does not require them. In contrast, when NoC is in a high error rate condition the reliability improvement modules can be turned on.

IV. PROPOSED SWITCH ARCHITECTURE

Reliability Improvement of the Switch

Considering the importance of crosstalk and SEU/MBU faults which are among the main reliability concerns in NoCs [12], the proposed switch architecture employs mechanisms to tackle with these two faults. Duplicated add parity is the used mechanism to guarantee the reliability of flits leaving the switch against crosstalk fault. Figure 2 shows the decoder and encoder of the DAP mechanism. DAP is a simple mechanism trying to prevent $\downarrow\uparrow\downarrow$, $\uparrow\downarrow\uparrow$, $\uparrow-\uparrow$, and $\downarrow-\downarrow$ transition patterns by duplicating wires of the communication channel [18] DAP also adds a one-bit parity to the transmitted flit to increase the hamming distance of consecutive flits to three. The hamming distance of three enables this mechanism to correct one bit, and detect two bit errors. As shown in Figure 2, encoder/decoder of the DAP mechanism are very simple, consequently this mechanism imposes negligible power and area overheads to the switch architecture. In contrast, the DAP mechanism has more than 100% wire overhead which have motivated authors to use this mechanism in a reconfigurable manner.

To improve the reliability of the switch against single and multiple bit upsets happening inside and outside of the switch, flit triplication mechanism is proposed. The flit triplication is considered as an end-to-end flow-control method to improve the reliability of packet delivery, i.e., the source and destination nodes are involved in the flit triplication mechanism. Figure 3 shows how flit triplication can help the destination node to recover the correct flit if a SEU/MBU has occurred in the flit during transmission

Finally, the third mechanism which is used in the reconfigurable switch is the hamming code to protect flits inside the switch. To do this, each newly received flit is coded by a hamming encoder at its arrival time. At the departure time, the flit is decoded and is checked to be correct. In this way, flits are protected against SEU/MBU occurring inside the switch.

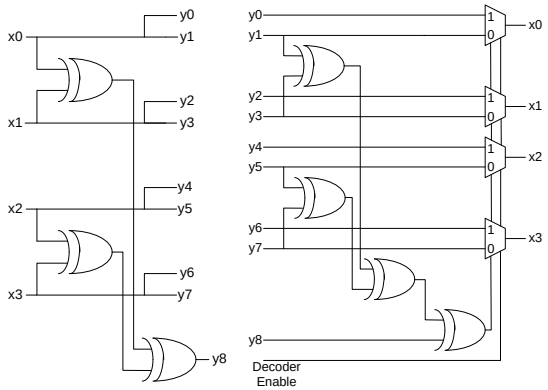

Figure 2. Encoder (a), and decoder (b) of DAP mechanism.

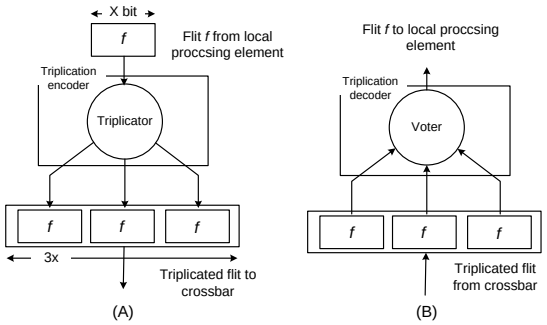

Figure 3. Flit triplication for SEU/MBU tolerance.

The question remained here is that how and when each the three mentioned mechanisms are turned on. In other words, how the current configuration of the switch is determined, configuration controller logic is embedded in the switch which is described in the following subsection.

Configuration Controller

Figure 4 represents the block diagram of the proposed switch architecture. As shown in this figure, the configuration controller logic which is embedded in the switch architecture, determines the current configuration for the switch. DAP and hamming decoder modules which are in the path of output channels are enabled/disabled by a control signal from the configuration controller. Flit triplication module which is in the path of injection and ejection channels (channels which are used by the processing element to inject and to eject its packet to the network respectively), is also under control of configuration controller. Hamming encoder modules which are in the path of input channels, as well, are controlled by the configuration controller. As previously mentioned, hardware of these mechanisms are clock gated when they are disable by the configuration controller to minimize their power overhead.

Configuration controller logic determines the current configuration of the switch based on the number of detected errors. To do this, each flit traversing in the network is equipped with one-bit parity. Flits are examined in every switch to count the number of errors occurred to them. Let us call this number in a typical switch by $E(error)$ which is excepted number of errors. Based on the value of $\frac{\mathrm{E(error)}}{\mathrm{Time}}$,

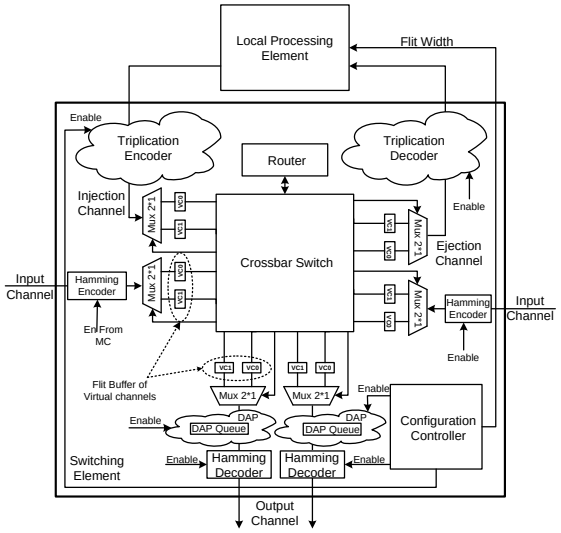

Figure 4. Block diagram of the proposed switch architecture.

which is the rate of error occurrence the switch is configured in one of the modes M_0 to M_3. Functionality of the switch in each of these modes is as follows:

M_0: In this mode all of the mentioned modules are in the state of off i.e., all of them are clock-gated.

M_1: In this mode the DAP mechanism which is intended to protect the switch against crosstalk faults is turned on. To do this, 1) appropriate control signals is sent to the DAP encoder/decoder, 2) appropriate flit width is declared to the local processing element which is (channel width/2)-1.

M_2: In this mode, flit triplication mechanism is on and other mechanisms (DAP and hamming) is off. To do this, 1) appropriate control signals is sent to the flit triplication coders, 2) appropriate flit width is declared to the local processing element which is (channel width/3).

M_3: All of the mentioned mechanisms i.e., DAP, flit triplication, and hamming coders are on to achieve maximum reliability.

Configuration controller logic incorporates five parity generator/checker modules to count the number of errors occurred in each input channel of the switch. Block diagram of the configuration controller is shown in Figure 5. The parity generator/checker modules works in all configurations. The used policy in the configuration controller logic to determine the current mode of the switch (referred as configuration selection algorithm in Figure 5) is shown in Figure 6.

This algorithm calculates the error rate of the switch. The error rate is examined by two predefined thresholds, T_1, T_2 . If the calculated error rate is lower than T_1, a mode with lower reliability level is selected. This means that the current reliability improvement mechanism can be turned off with a low reliability loss. If the error rate is greater than T_1 and lower than T_2, the switch is remained in the current mode. If the error rate is greater than T_2, high error rate condition is detected, consequently the switch is configured in a higher reliability level mode. There are two special cases: 1) lower reliability improvement mode when the switch is in the mode M_0, is the mode M_0 itself, and 2) higher reliability improvement mode when the switch is in the mode M_3, is the mode M_3 itself.

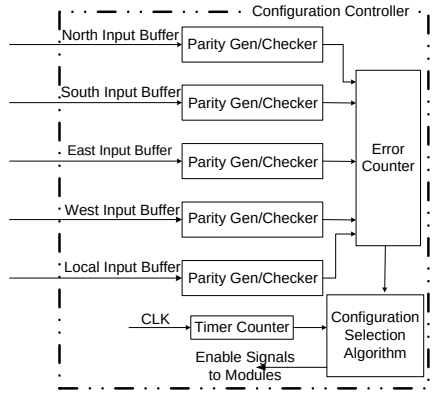

Figure 5. Block diagram of the configuration controller.

```
ConfigurationSelection{
    Erate=E(Error)/time;
    If (Erate<T1)
        Switch( Current mode){
            Case (current mode=M 0);
            Nextmode=M0;
            Case (current mode=M 1);
            Nextmode=M0;
            Case (current mode=M 2);
            Nextmode=M0;
            Case (current mode=M 3);
            Nextmode=M1;
        }
    If (T1<Erate<T2)
        Nextmode=current mode;
    If (Erate>T2)
        Switch( Current mode){
            Case (current mode=M 0);
            Nextmode=M1;
            Case (current mode=M 3);
            Nextmode=M3;
            Case (current mode=M 2);
            Nextmode=M3;
            Case (current mode=M1);
            If (T2<Tt)
                Nextmode=M3;
            Else
                Nextmode=M2;
}}
```
Figure 6. Pseudo code of configuration selection algorithm.

The reconfiguration policy considers the fact that the crosstalk faults are more probable than SEU/MBU faults [11]. Thus in the modeM_0, higher reliability level is the modeM_1. If this reconfiguration does not reduce the rate of error occurrence, mode M_2 is selected by the configuration controller.

V. EVALUATIONS

Experimental evaluation

We implemented the reconfiguration switch architecture in XMulator simulator [28], which is a framework for interconnection and NoC network. This simulator provides the possibility of defining details of hardware and algorithms with performance considerations. XMulator (XML+Simulator) uses new software architecture paradigms (object oriented design) and new programming languages. Representing topologies, parameters and outputs in this simulator is in XML format which makes it flexible. It uses a multi-layered architecture to be coherent with the modern software architecture rules. Each layer provides services to the upper layer(s) and only depends on the lower ones. Listener-based integration solves the problem of two-way dependency which makes the design less extensible. There is a complete, flit-level, event-based, and extensively detailed package for simulation of interconnection networks which enabled us to simulate interconnection networks with arbitrary topology even in the presence of faults.

XMulator uses Orion [29] for calculating the power. Orion is a power-performance interconnection network simulator that provides detailed power characteristics, and performance characteristics, which enables us to make tradeoffs in power-performance at the architectural-level.

Implementation Details

The implemented network is composed of 16 nodes arranged as a 4×4 mesh. We choose mesh topology as is easy to implement. For traffic pattern, uniform distribution destination is used to inject and route the messages to their destinations. In this distribution destination, the source node with an equal probability sends messages to different destination in the network. The switching method is wormhole and message length of 32 flits and flit width of 64 is chosen. Injection and ejection channels have 2 virtual channels. Message generation rate for each node is 0.02 with maximum simulation event 5000000. Moreover, the process feature size and working frequency of the routers is set to 70nm and 250 MHz, respectively in the Orion library. Also the network traffic is synthetic. The fault model applied in this simulation is MBU which represents MBU, SEU and crosstalk fault models.

In the simulation process, we examined different configurations with various reliability levels. The results of different configurations can be seen in following conditions:
C1: error rate is selected in a way that network nodes are 20% of times in mode M_0, 20% in mode M_1, 30% in mode M_2, 30% in mode M_3.
C2: error rate is selected in a way that network nodes are 60% in mode M_0, 20% in mode M_1, 10% in modeM_2, 10% in mode M_3.
C3: 33% in mode M_0, 33% in mode M_1, 33% in modeM_2, 1% in mode M_3.
C4: 40% in mode M_0, 30% in mode M_1, 15% in mode M_2, 15% in mode M_3.

Figure 9 compares the total network latency of *C1*, *C2*, *C3* and *C4* configurations. This diagram shows the effect of the reliability improvement methods on the performance. We can see that by increasing the reliability levels as the process like duplicating in DAP, triplication in TMR and adding the hamming code are forcing to the network we have more latency in the network. In diagram, *C1* and *C2* have more latency because they are more in higher reliable modes $M_2$and M_3. By reducing the level of the reliability, diagrams are nearing to normal network performance even in *C1* that the configuration is more than 60% like the normal mode and 40% reliability improvement; we have the enhancement of performance.

Analytical evaluations

Since the proposed switch architecture is working in four different configurations, Markov chain is used to model the proposed switch architecture. In this regard, configuration modes of the switch are considered as the states of a Markov chain. Rates of transition in the Markov model represent the probability of moving from one configuration to another. Figure 10 shows the used Markov model to calculate the

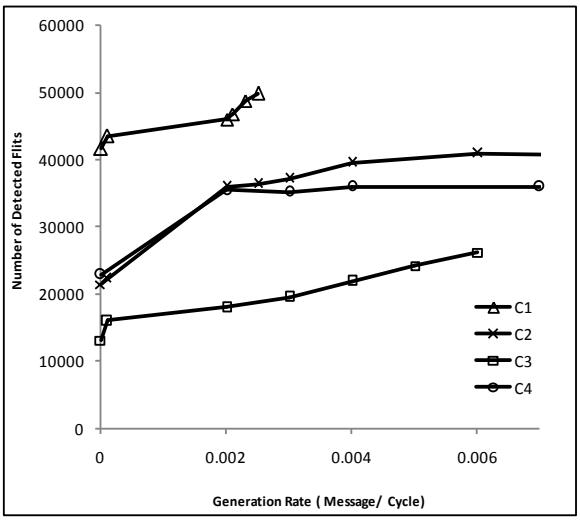

Figure 7. Number of detected faulty flits in different operation modes.

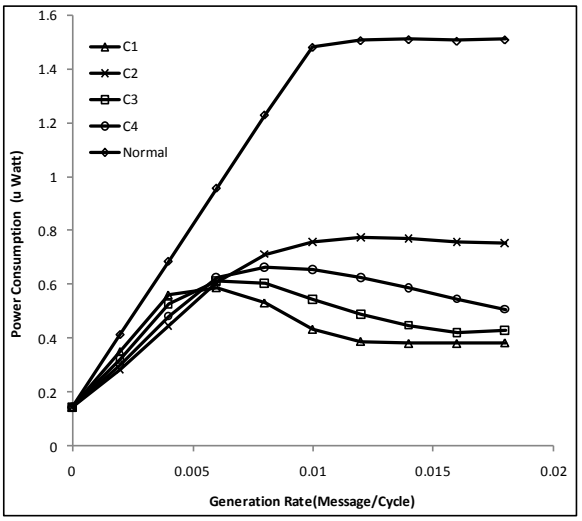

Figure 8. Power Consumption of the proposed architecture in different working conditions versus a normal network.

overall performance, power consumption and reliability of the switch. For the sake of generality the model assumes different transition rates for each state. Also it has been assumed that the reliability, performance and power consumption of each mode M_i is R_{M_i}, Per_{M_i}, Pow_{M_i}. Based in these assumptions, the steady state solution of the model gives the average performance, power consumption and reliability of the switch architecture. The model is defined by the following transition matrix and is solved by the use of Maple tool:

$$\begin{pmatrix} 1-\lambda_1 & \lambda_1 & 0 & 0 \\ \mu_1 & 1-\mu_1-\lambda_3-\lambda_2 & \lambda_3 & \lambda_2 \\ \mu_3 & 0 & 1-\mu_3-\lambda_4 & \lambda_4 \\ 0 & \mu_2 & 0 & 1-\mu_2 \end{pmatrix}.$$

In order to make sense of the overall performance, reliability and power consumption of the proposed switch, Table 1 represents assumption we have made about R_{M_i}, Per_{M_i}, Pow_{M_i} for $i = 0$ to 3. Using these values and the transition rates which are reported in Table 2, the overall performance, reliability and power consumption of the proposed switch are reported in Table 2. Although values of Table 1 and transitions rates of Table 2 are not extracted from a real NoC, they are good approximations when we consider the proposed reliability improvement mechanisms described in Section 4. Examinations show that the model provides a significant speed-up in the evaluation phase of NoC design, experimental evaluation takes about 3 hours while with the analytical model we can have speed up to 10 seconds.

CONCLUSIONS

This paper proposed a reconfigurable switch architecture which is capable of operating in four configurations with different levels of reliability. This is done by the use of configuration controller logic. Widely used reliability improvement mechanisms including DAP, flit triplication and hamming code are used as the different configurations of the switch. The reconfiguration policy is designed in a way which minimizes the overall performance and power consumption overheads of the switch. Evaluation experiments show a noticeable reliability improvement and power saving of at least 20% with a negligible performance overhead.

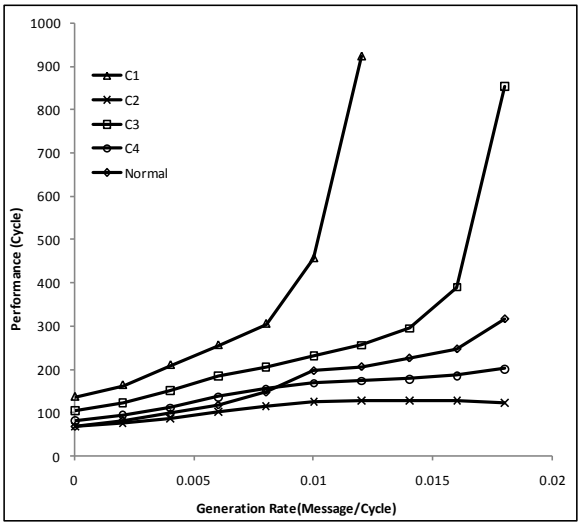

Figure 9. Average packet delivery time for the proposed architecture in different working conditions versus a normal network.

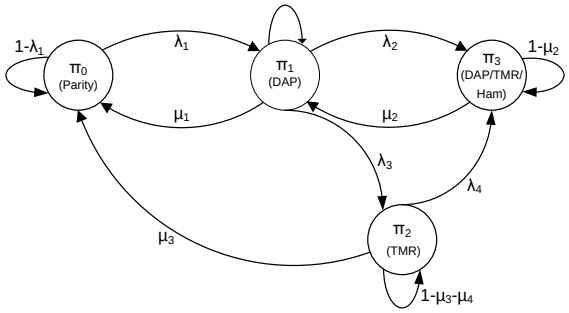

Figure 10. Markov model of the proposed switch architecture.

TABLE 1. RELIABILITY, POWER AND PERFORMANCE FOR EACH MODE OF THE PROPOSED SWITCH.

$Reliability_{\Pi 0}$=50%	$Reliability_{\Pi 1}$=60%	$Reliability_{\Pi 2}$=70%	$Reliability_{\Pi 3}$=80%
$Power_{\Pi 0}$=100nW	$Power_{\Pi 1}$=120nW	$Power_{\Pi 1}$=142nW	$Power_{\Pi 3}$=156. 2nW
$Performance_{\Pi 0}$=90%	$Performance_{\Pi 1}$=80%	$Performance_{\Pi 2}$=70%	$Performance_{\Pi 3}$=60%

TABLE.2. RELIABILITY, POWER AND PERFORMANCE FOR EACH APPLICATION.

	λ_i	μ_i	Reliability	Performance	Power
Application 1	0. 1	0. 1	2.808695652	2.426086957	550.2608696
Application 2	0. 25	0.25	1.272727273	1.272727273	250.5818182
Application 3	0. 3	0. 3	1.111877395	1.151724138	219.2337165

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, 2002, pp. 70–78.

[2] S. Kumar, A. Jantsch, J.P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," *IEEE Symposium on VLSI*, 2002, pp. 117–124.

[3] M. Ali, M. Welzl, and M. Zwicknagl, "Networks on chips: scalable interconnects for future systems on chips."

[4] D.E. Culler, J.P. Singh, and A. Gupta, *Parallel computer architecture: a hardware/software approach*, Morgan Kaufmann Pub, 1999.

[5] L. Benini and G. De Micheli, "Networks on chips: A new paradigm for component-based MPSoC design," *IEEE Computer*, 2002, pp. 70–78.

[6] S. Murali, T. Theocharides, N. Vijaykrishnan, M.J. Irwin, L. Benini, and G. De Micheli, "Analysis of error recovery schemes for networks on chips," *Trans. VLSI Systems*, vol. 8, 2000, pp. 379–391.

[7] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, 2006, p. 1.

[8] A.P. Frantz, F.L. Kastensmidt, L. Carro, and E. Cota, "Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk," *IEEE International Test Conference, 2006. ITC'06*, 2006, pp. 1–9.

[9] K.N. Patel and I.L. Markov, "Error-correction and crosstalk avoidance in DSM busses," *Proceedings of the 2003 international workshop on System-level interconnect prediction*, 2003, pp. 9–14.

[10] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, "Exploring fault-tolerant network-on-chip architectures," *International Conference on Dependable Systems and Networks, 2006. DSN 2006*, 2006, pp. 93–104.

[11] D. Bertozzi, L. Benini, and G. De Micheli, "Low power error resilient encoding for on-chip data buses," *Proceedings of the conference on Design, automation and test in Europe*, 2002, p. 102.

[12] A. Patooghy, M. Fazeli, and S.G. Miremadi, "A low-power and SEU-tolerant switch architecture for Network on Chips," *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing*, 2007, pp. 264–267.

[13] B. Ahmad, A.T. Erdogan, and S. Khawam, "Architecture of a dynamically reconfigurable NoC for adaptive reconfigurable MPSoC," *First NASA/ESA Conference on Adaptive Hardware and Systems, 2006. AHS 2006*, 2006, pp. 405–411.

[14] Y.E. Krasteva, E. de la Torre, and T. Riesgo,"Virtual Architectures for partial runtime reconfigurable systems. Application to Network on Chip based SoC emulation," *34th Annual Conference of IEEE Industrial Electronics, 2008. IECON 2008*, 2008, pp. 2489–2494.

[15] W.J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[16] J. Duato, S. Yalamanchili, and L.M. Ni, *Interconnection networks: An engineering approach*, Morgan Kaufmann, 2003.

[17] S.R. Sridhara and N.R. Shanbhag, "Coding for system-on-chip networks: a unified framework," 2004.

[18] P.P. Pande, A. Ganguly, B. Feero, B. Belzer, and C. Grecu, "Design of Low power & Reliable Networks on Chip through joint crosstalk avoidance and forward error correction coding," *21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2006. DFT'06*, 2006, pp. 466–476.

[19] L. Ciani, M. Catelani, and L. Veltroni, "Fault tolerant techniques to diagnose and mitigate Single Event Upset (SEU) effects on electronic programmable devices," *Proc. of 16th ImEKo TC4 symposium*, 2008.

[20] T. Burd, T. Pering, A. Stratakos, and R. Brodersen,"A dynamic voltage scaled microprocessor system," *2000 IEEE International Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC*, 2000, pp. 294–295.

[21] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, 2006, pp. 155–166.

[22] S. Krishnamohan and N.R. Mahapatra,"A highly-efficient technique for reducing soft errors in static CMOS circuits," 2004.

[23] T. Pionteck, C. Albrecht, and R. Koch, "A dynamically reconfigurable packet-switched network-on-chip," *Proceedings of the Design Automation & Test in Europe Conference*, 2006, p. 38.

[24] M.P. Véstias and H.C. Neto,"Co-synthesis of a configurable SoC platform based on a network on chip architecture," *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, 2006, p. 53.

[25] H. Elmiligi, A.A. Morgan, M.W. El-Kharashi, and F. Gebali, "Performance Analysis of Networks-on-Chip Routers," *International Design and Test Workshop, 2007 2nd*, 2007, pp. 232–236.

[26] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," *13th IEEE International On-Line Testing Symposium, 2007. IOLTS 07*, 2007, pp. 49–56.

[27] J. Rabaey, *Low power design essentials*, Springer Verlag, 2009.

[28] A. Nayebi, S. Meraji, A. Shamaei, and H. Sarbazi-Azad, "Xmulator: A listener-based integrated simulation platform for interconnection networks," *First Asia International Conference on Modelling & Simulation, 2007. AMS'07*, 2007, pp. 128–132.

[29] H.S. Wang, X. Zhu, L.S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings*, 2002, pp. 294–305.