# A Low Cost Method to Tolerate Soft Errors in the NoC Router Control Plane

Changlin Chen, Sorin D. Cotofana
Computer Engineering, Software and Computer Technology
Delft University of Technology, Delft, the Netherlands
Email: {c.chen-2, S.D.Cotofana}@tudelft.nl

***Abstract*—In this paper, we propose a low cost method to tolerate soft errors in the main NoC router functional units, i.e., routing units, Virtual Channel (VC) allocators, and switch allocators. The idea behind our proposal is to utilize the idle routing units at neighboring input ports to do redundant Routing Computation (RC) for the local routing requests, and detect errors in the VC Allocation (VA) and Switch Allocation (SA) results by checking if they are consistent with RC results and are in legal states. If required, soft errors are recovered by redoing the failed procedures and retransmitting the flits. Experimental results on an 8×8 2D NoC indicate that: (i) in the routing units, the proposed method requires 38% more silicon real estate than the Σ & Branch method when XY routing algorithm is used, but it is more general and can be utilized in conjunction with many more routing algorithms; and (ii) in the combined VA/SA units, the proposed method is simpler and more effective than the state of the art methods. When compared with the Triple Modular Redundancy strategy, for similar error detection and correction efficacy, the proposed method can reduce the area and power overhead in routing units by 53% and 38%, respectively, and in combined VA/SA units by 45% and 46%, respectively. The average packet transmission latency increase is less than 5% even if the soft error rate is 0.1/cycle, when compared with that of the baseline router architecture.**

***Keywords*-Networks-on-Chip; Soft error tolerant; Router control plane;**

## I. Introduction

The aggressive technology scaling enables an increasing number of cores to be integrated in one chip. Networks-on-Chip (NoC) are the de facto on chip interconnect strategy, which can meet the performance and power consumption budgets of such designs [1]. However, the smaller transistor dimensions also render the chips less reliable. Due to miniaturization, permanent and soft errors occuring in cores or NoCs are more frequent and in order to prevent application misbehavior one should detect and correct them. In this line of reasoning, in this paper, we focus on tolerating soft errors in NoCs.

NoCs are composed of routers and links. As illustrated in Fig. 1, the components of a router can be partitioned into the datapath group, which mainly consists of memory elements, and the control plane group, which mainly consists of combinational logic circuits [2]. Previous research has been focusing on tolerating soft errors in links [3] and router datapath [4], which are supposed to be more susceptible to soft errors than the router control plane. However, the Soft Error Rate (SER) in nowadays combinational logic circuits is already comparable with that encountered in unprotected memory elements [5]. Consequently, we further concentrate our attention on soft errors occurring in the router control plane.

The main functional units in the router control plane are Routing Units (RUs), Virtual Channel (VC) allocators, and switch allocators. For each received packet, the RU computes the output port, the eligible output VCs at that output port when adaptive routing algorithms, e.g., [6], are used, and the necessary misrouting information when fault tolerant routing algorithms, e.g., [7], are used. The VC allocator chooses one free eligible output VC and assigns it to the packet. Switch allocators decide which flits can be transmitted in the next clock cycle. We note that soft errors in these functional units can lead to deadlock or flits loss during packets transmission.

A soft error occures only if a Single Event Transient (SET) is propagated to an output and latched into a memory element [8]. During the propagation, SETs can be diminished by logical masking, electrical masking, and temporal masking mechanisms [9]. Circuit design methods, e.g., gate resizing [10], circuit rewiring [5], and output multiple sampling [11], have been proposed to exploit these masking mechanisms. However, these methods have the drawbacks of high physical level design effort and high chip area overhead.

In NoC routers, it is inherently required that one output resource can only be assigned to at most one request at a given time. This feature is exploited in [12]–[14] to detect soft errors with low silicon overhead. However, such schemes either have restricted application scope [12], or are still too complicate to allow for practical implementations [13].

In this paper, we propose a low cost method to tolerate soft errors in the main functional units in the router control plane.

By noticing that the RU is usually replicated at each input ports, especially if it is logic based, and it is only used when a head flit arrives at that port, we propose to utilize idle neighboring RUs as well as the local RU to do redundant Routing Computation (RC) for the local routing requests. The proposed method requires 38% more area overhead than the Σ & Branch method proposed in [12] when XY routing algorithm is used on a 2D mesh NoC, but it is applicable in conjunction with many more routing algorithms. When compared with the Triple Modular Redundancy (TMR) strategy, for similar error detection and correction efficacy, our method

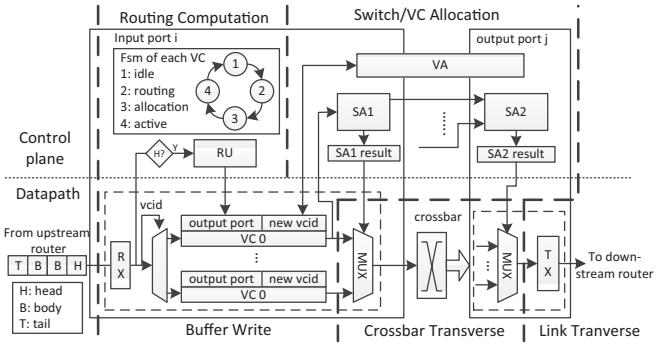



Fig. 1. Typical VC based router architecture.

can reduce the area and power overhead in RU by 53% and 38%, respectively, when Opt_Y routing algorithm [6] is used.

One common requirement for the VC Allocation (VA) and Switch Allocation (SA) results is that they should be consistent with the RC results. Moreover, they must correspond to legal states, i.e., one output resource can maximally be assigned to one request. In view of these, we propose to detect the illegal VA results at the output port side by checking if multiple head flits were transmitted to the same output VC. We expand the method to detect illegal arbiteration results proposed in [12] to detect erroneous SA results and we implement it with simple logic. When compared with the Allocation Comparator (AC) unit in [13], our method is less expensive in terms of area and more reliable. If utilized in the realization of the combined VA/SA units, which are embedding matrix arbiters, the proposed method can reduce the TMR area and power overhead strategy by 40% and 46%, respectively.

The rest of the paper is organized as follows. Section II presents a brief survey of related work. Section III introduces the proposed soft error tolerant method. Section IV evaluates our approach and compares it with tightly related work. Section V concludes the presentation.

## II. Related Work

The most intuitive way to tolerate soft errors in the functional units is TMR, with the obvious drawback of high area and power consumption overhead.

A $\Sigma$ & Branch method is proposed in [12] to detect soft errors in RC results by examining the appearance of forbidden signal patterns in RUs. This method is proved to be efficient to detect erroneous output ports, because each packet can only be transmitted to one output port unless multicast is needed. However, it cannot detect errors in other routing results, e.g., eligible output VCs, which can include multiple existing output VCs.

[13] proposed an Allocation Comparator (AC) unit to detect errors in VA results. The output VC of each input VC is compared with the routing results to check if it is an eligible one, and with that of other input VCs to check if the output VC is occupied already. Considering that an output VC can be assigned to any input VC, such a comparison cannot be implemented with trivial effort.

To detect erroneous SA results, [14] proposed to add a message ID to each flit and check if the flits received by one input VC have the same ID. The drawback of this method is that the link width is increased and the errors are detected in downstream routers. Differently, [13] and [12] check if the SA results are consistent with the RC results and have legal states after they are registered. In this paper, we apply this checking principle to VC based NoC routers and implement it with simple logic.

## III. Tolerate Soft Errors in the Control Plane

As illustrated in Fig. 1, the transmission of a head flit in the router must sequentially go through 3 stages: Routing Computation (RC), VC Allocation (VA) and Switch Allocation (SA), and Crossbar Transverse (CT). An extra Link Transverse (LT) stage is usually required between two neighboring routers. In this paper, we assume that SA and VA are implemented in the same stage, speculatively or combined together, to reduce the pipeline stage number. Body and tail flits just need to go through the SA and CT stages. Each input VC has four states: *idle*, *routing*, *VC allocation*, and *active*. An input VC waits for routing results in the *routing* state and VA results in the *VC allocation* state. After that, it stays in the *active* state until the entire packet is successfully transmitted.

In this section, we assume that the requests to do relative computations, i.e., RC, VA, and SA, are asserted correctly and are well protected against soft errors.

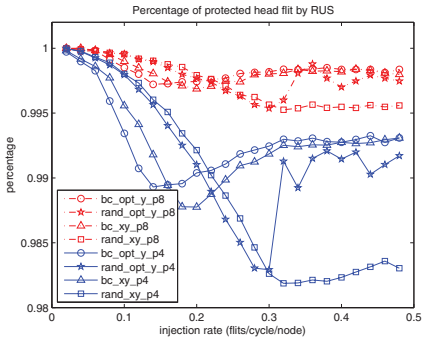
### A. Errors in Routing Units

RUs decide how packets should be transmitted according to the employed routing algorithm. Erroneous RC results can misdirect packets and lead to deadlock or packet loss. Correct RC results are the precondition of correct VA and SA results.

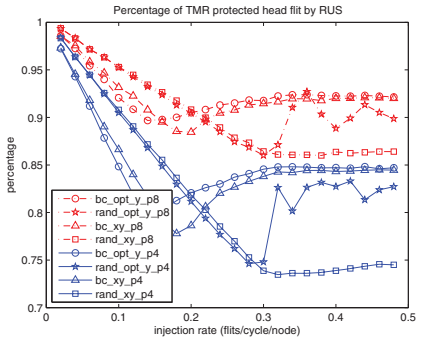
In a typical NoC router design, the RU is replicated at each input port to increase throughput. Each RU deals with local routing requests only, and is solely used when head flits arrive at that port. Assuming that the packet length is $l$, the average RU utilization rate is less than $1/l$, because flits do not arrive in every cycle. This means that RUs are most of the time idle and we propose to utilize idle neighboring RUs, when available, to do redundant routing computation for local routing requests.

The proposed RU Sharing (RUS) method is illustrated in Fig. 2. When a head flit is received at an input port, the routing request is sent to the local RU as well as RUs at two neighboring ports. At each port, the highest priority to utilize the RU is given to the local routing request. When the local routing request is not asserted, the priority is given to a neighboring port, e.g., port $i-1$ in Fig. 2. RC results from two neighboring RUs are compared. The comparison result along with a *valid* signal are sent back to the routing request initiator input port. The *valid* signal indicates whether the RC results are computed for that port.

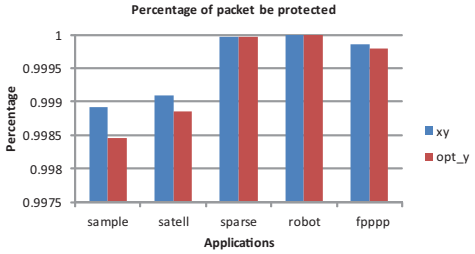
To save silicon cost and limit the critical path length increase in the RC stage, the RC results from neighboring ports are only used to check the correctness of local results. When erroneous RC results are detected, the RC is re-executed for



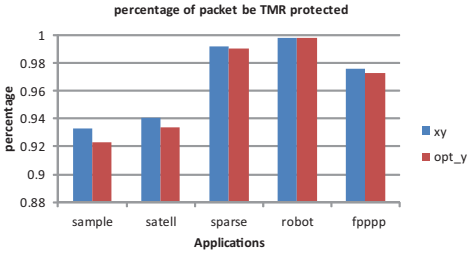

(a) Percentage of protected packets in synthetic traffics;




(b) Percentage of TMR protected packets in synthetic traffics;




(c) Percentage of protected packets in real applications.




(d) Percentage of TMR protected packets in real applications.

Fig. 3. Percentage of packets protected by RUS in synthetic traffics and real applications. Bit compliment (bc) and random (rand) traffic patterns, XY and Opt_y routing algorithms, packets with 8 flits (p8) and 4 flits (p4) are evaluated.

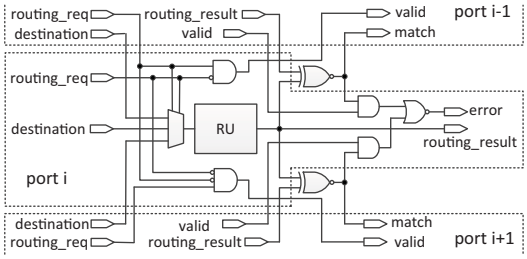



Fig. 2. Routing unit sharing among neighboring input ports.

the packet. We note here that with the overhead of a voter at each port, the correct RC result can be determined when two idle neighboring RUs are available for a routing request.

In practice, depending on the executed application and traffic, situations may occur when both neighboring RUs are occupied and only the local RC results are available at an input port. In this case, the packet has to wait until at least one neighboring RU is available to make sure that the RC results can be checked for correctness.

However, a deadlock situation can happen in this case. When head flits are received at every input port simultaneously, every RU is occupied by the local routing request. In the next cycle, the local routing request at every port is asserted again by these head flits or newly received ones. Thus the deadlock lasts and eventually all the input VCs are occupied and no packet can go further.

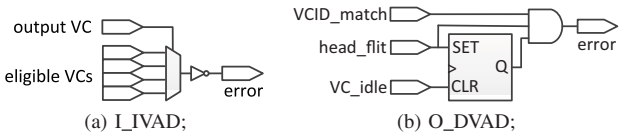
The deadlock can be solved by temporarily prohibiting the routing request of one input port, such that one input port has an idle neighboring RU to utilize. Most probably the routing results from the two RUs match with each other and the port will not assert routing request until a new head flit arrives. In this way after several cycles, every packet gets the correct RC results and enter the next stage.

In Fig. 3, we illustrate the probability that RC results are checked for correctness in various situations in an $8 \times 8$ 2D mesh NoC. We can observe in Fig. 3(a) that for all the evaluated synthetic traffic patterns, more than 98% of the RC results are protected in each router, even if the NoC is saturated. When the traffic load is low, almost every head flit can 'borrow' RUs from neighboring ports. For 5 real applications from [15], our evaluations indicate (see Fig. 3(c)) that more than 99.8% of their packets are protected. Moreover, more than 72% of the packets in synthetic traffics (Fig. 3(b)) and more than 92% of the packets in the applications' traffics (Fig. 3(d)) are TMR alike protected. The statistic results also reveal that the longer the packet length, the higher the probability that the RC results are protected.

Routing results are stored in registers and must be maintained unchanged until the entire packet is successfully transmitted. Otherwise, the next flits will deviate from the path reserved by the head flit. We assume that these registers are protected against soft errors with proper technology, e.g., [16].

### B. Errors in VC Allocators

After the output port of a packet is derived, the VC allocator assigns a free output VC at that output port to the packet. The




(a) I_IVAD; (b) O_DVAD;

Fig. 4. Detect soft errors in VA result.

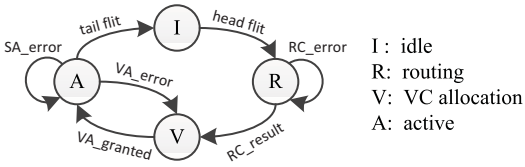



Fig. 5. Changes of input VC states.

VC's index (VCID) can be changed when errors happen in the VA unit. Then the output VC may become: (1) a wrong output VC, which does not exist or is not an eligible VC according to the RC results; or (2) an eligible output VC which is already assigned to another packet.

Errors of the first case can be easily detected by checking if the granted output VC is consistent with the RC results [13]. Assuming that eligible output VCs are indicated by setting the relative bits to '1' in the RC results, the error can be detected by the logic in Fig. 4(a). We name this method as Input Side Invalid VC Allocation Detection (I-IVAD).

In a router which has $p$ physical ports and $v$ VCs at each port, an output VC can be assigned to any of the $pv$ input VCs. It is expensive to check if any two input VCs are granted with the same output VC by doing pairwise comparisons [13]. As it is guaranteed by I-IVAD that each input VC is assigned with an eligible output VC, we propose an Output Side Duplicated VC Allocation Detection (O-DVAD) method (see Fig. 4(b)) to detect type (2) errors.

In O-DVAD, a 1-bit latch is added to each output VC with the initial state '0'. Once a head flit is transmitted to the downstream router, the latch of the relative output VC turns to '1' to indicate that the output VC is occupied. The latch stays in '1' until the output VC returns to the *idle* state. If another head flit is transmitted to the same output VC when the latch is '1', it means that an attempt is made to assign the output VC to multiple packets and the error is detected. The input VC that is related with the error can be determined based on the SA results. The output VC remains under the utilization of the first packet it was assigned to.

I-IVAD and D-DVAD are implemented in the CT stage. The strategy to recover VA errors is presented in detail in Section IV. Similar with the RC results, we assume that VA results are stored in registers protected against soft errors.

### C. Errors in Switch Allocators

Switch allocation is usually divided into the local stage (SA1) at the input side and the global stage (SA2) at the output side (see Fig. 1). An input VC can transmit flits only when it wins both SA stages. The main components in switch allocators are arbiters.

The 4 symptoms of soft errors in SA results are explicitly discussed in [13]: (1) no asserted request is granted in SA1 or SA2, (2) an input VC is allocated with a wrong output port, (3) in the same cycle, multiple input VCs at one input port are granted in SA1, or multiple input ports are granted by SA2 at one output port, or an input port is granted by multiple output ports, and (4) an input VC which does not assert request wins both SA1 and SA2.

An efficient method to detect erroneous results of arbiters is presented in [12]. We apply this method in VC based routers to detect erroneous SA results. Note that the error detection procedure has to be applied to SA1 results, SA2 results, and SA2 grants of each input port.

The SA results correctness is checked in the CT stage after they are registered. Thus soft errors in the registers can also be detected. In case the errors are caused by flipped priority registers in arbiters, the relative arbiters are reset whenever an SA error is detected [12] such that correct results can be obtained by redoing the allocation.

## IV. Error Recovery

To recover from errors in the control plane, the failed procedures must be repeated to generate correct results, and flits have to be retransmitted when necessary. During the recovery, the states of input VCs change as illustrated in Fig. 5.

Errors in RC results (*RC_error*) are detected in the RC stage. When an *RC_error* occurs, the input VC stays in the *routing* state instead of entering the *VC allocation* state. The routing request is reasserted when no new head flit arrives at the input port. Most probably the correct routing results will be derived with one clock cycle overhead. The head flit is not transmitted yet in this stage. So there is no need to do any retransmission.

Both errors in VA results and SA results are detected in the cycle after they are registered, i.e., in the CT stage, such that the critical path in the VA/SA stage is not changed. If an error is detected in the VA result, the input VC returns to the *VC allocation* state from the *active* state. Both VA request and SA request have to be reasserted to compete for a new output VC and flit transmission. The mistakenly allocated output VC needs to be released by the input VC. If an error is detected in SA results, only the SA requests need to be reasserted for flit retransmission. When a VA or SA error is detected, the VA/SA results derived in the same cycle are abandoned. Thus at least two extra clock cycles will be introduced.

In the CT stage, flits are read out from input buffers and are sent to output ports. If a VA or SA error is detected, the relative flit is ignored by the link registers at the output port, and then retransmitted when correct VA and SA results are derived. Because the original flit is still kept in the input buffers at this stage, retransmission buffers are not required.

It is also possible that there is no soft error in the control plane but an error signal is asserted by the error detection logic. Then the error recovery mechanism is triggered, which does not introduce new errors but only increase the flit transmission latency with several cycles.

TABLE I
AREA AND POWER OF SOFT ERROR TOLERANT METHODS FOR RU

| | Area ($\mu m^2$) | | Dyn. Power ($\mu W$) | | Leak. Power ($\mu W$) | |
|---|---|---|---|---|---|---|
| Routing Algorithm | XY | Opt_Y | XY | Opt_Y | XY | Opt_Y |
| baseline | 219.6 / 100% | 638.3 / 100% | 18.0 / 100% | 47.4 / 100% | 0.758 / 100% | 2.174 / 100% |
| RUS | 465.8 / 212% | 1018.4 / 160% | 52.8 / 294% | 95.6 / 200% | 1.768 / 233% | 4.027 / 185% |
| TMR | 774.0 / 352% | 2157.5 / 338% | 60.1 / 335% | 153.1 / 321% | 3.004 / 396% | 8.191 / 376% |
| $\Sigma$ & Branch | 336.6 / 153% | – / – | 25.6 / 143% | – / – | 1.290 / 170% | – / – |

TABLE II
AREA AND POWER OF SOFT ERROR TOLERANT METHODS FOR VA/SA

| | Area ($\mu m^2$) | | Dyn. Power ($mW$) | | Leak. Power ($\mu W$) | |
|---|---|---|---|---|---|---|
| Arbiter type | Matrix | Round-Robin | Matrix | Round-Robin | Matrix | Round-Robin |
| baseline | 10010 / 100% | 6794 / 100% | 2.1 / 100% | 1.5 / 100% | 62.1 / 100% | 44.3 / 100% |
| proposed | 12828 / 128% | 10127 / 149% | 2.5 / 119% | 1.8 / 120% | 81.0 / 130% | 66.7 / 151% |
| TMR | 23351 / 233% | 13824 / 203% | 4.6 / 219% | 2.4 / 160% | 147.7 / 238% | 95.5 / 216% |

## V. EVALUATION

The proposed soft error tolerant approach and the tight related ones are evaluated in the context of a wormhole switched 8×8 2D mesh NoC system [17]. The routers are VC based and implemented according to the architecture in Fig. 1. Each router has 5 physical ports (PC), and each PC has 4 VCs. Note that the proposed method is also applicable in 3D symmetric NoC systems, as the only difference is that the routers have 7 PCs in such a system.

As soft errors happen only when SETs propagate to an output and are captured by registers [8], instead of injecting soft errors into the gates in the functional units, we simulate the symptoms of soft errors by injecting soft errors into the final results, i.e., the RC, VA, and SA results. We assume that: (i) only one error can happen in one router in a cycle, and (ii) the router datapath is error free.

### A. Reliability

Simulations with different soft error rates illustrate that the proposed method can detect and recover all kinds of errors in RC, VA, and SA results, when assumption (i) is satisfied. Thus the NoC system can operate normally unless errors happen in both the functional units under protection and the error detection logic circuits.

Routing Unit Sharing provides TMR alike protection of the RC results in most of the time (see Fig. 3). Thus the RUS reliability for each packet is similar with that of TMR. Although the $\Sigma$ & Branch method [12] is claimed to be more reliable than TMR in the detection of erroneous output ports, it cannot detect errors in other RC results, e.g., eligible output VCs, which are required in sophisticated routing algorithms, e.g., Opt_Y, thus RUS has much larger application scope than $\Sigma$ & Branch.

To detect the erroneous VA results when one output VC is assigned to multiple input VCs simultaneously, the proposed method requires one D-latch and one AND gate for each output VC. In contrast, the AC unit in [13] compares the output VC of each input VC at the input side. Given that in a VC based router, an input VC can be assigned with any output VC at any output port, the AC unit is quite complicate to implement. In the evaluated router architecture, at least 950 XOR gates are required to do the comparison in the AC unit. Assuming that the SER of a D-latch is 10 times higher than that of a gate [2], the SER ratio of O-DVAD to AC unit is $220/(950+X)$, where $X$ is the number of other gates besides XOR gates in an AC unit. Thus the O-DVAD logic is more reliable than the AC unit.

### B. Area and Power Overhead

Silicon area is an important issue that affects the chip reliability as large area overhead implies a higher chance to be hit by high energy particles, hence a higher SER [8].

RUs and combined VC/switch allocators, equipped with different soft error tolerant methods, are implemented at RTL level by using Verilog HDL, and synthesized using the Synopsys Design Compiler with TSMC 65-nm standard cell technology. The area costs and power consumption of different methods are illustrated in Table I and Table II, respectively.

We note that RUS does not increase the number of RUs in the baseline router, and only requires half of the number of the comparators required by the TMR strategy. Thus the RUS implementation cost is much lower than that of TMR, especially when complicated routing algorithms are used. To be specific, the RUS area overhead is 53% less than that of TMR when Opt_Y routing algorithm is used, while the reduction is 40% when XY routing algorithm is used. The RUS power consumption exhibits the same trend. Although the RUS area and power costs are all higher than those of the $\Sigma$ & Branch method [12], it is more general and can be applied in conjunction with many more routing algorithms.

The cost to tolerate soft errors in VC and switch allocators in our method is proportional with the numbers of output ports and VCs in a router, regardless of the allocators implementation details. Table II suggests that the proposed methods reduce the TMR area overhead by 45% and 27% when matrix arbiters and round-robin arbiters are used, respectively. The power consumption of the proposed method is also lower than that of TMR.

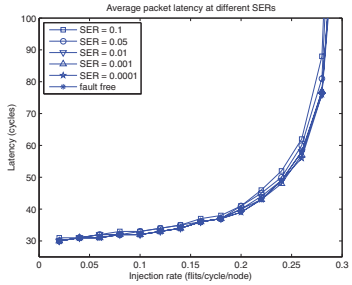



Fig. 6. Average latency at different SERs and FIRs.


Fig. 7. Detected error numbers at different SERs.

TABLE III
PERFORMANCE OF RUS AT DIFFERENT SERS

| SERs | 0.0001 | 0.001 | 0.01 | 0.05 | 0.1 |
|---|---|---|---|---|---|
| Injected errors | 31 | 432 | 4458 | 21966 | 43275 |
| Idle RUs available | 30 | 425 | 4380 | 21659 | 42626 |
| unprotected | 1 | 7 | 78 | 307 | 649 |

### C. System Performance

The average packet transmission latencies at different SERs and different Flit Injection Rates (FIRs) are illustrated in Fig. 6. The error recovery mechanism is triggered when an soft error is detected in RC, VA, or SA results, and is completed in one or two clock cycles. The experimental results indicate that even if the SER is as high as 0.1/cycle in each router, the average packet delivery latency increase is less than 5% when compared with the error free case.

The numbers of detected errors in different functional units are illustrated in Fig. 7. As both RUs and VC allocators work at packet rate, they are supposed to have similar number of errors. However, in the proposed method, the situation that a packet has no available idle neighboring RU is treated in the same way as when an RC_errors happen. As a consequence, the error number in RC results is much higher than that in VA results, but they still have the same increasing trend as the SER increases, because the percentage of packets that cannot be protected by RUS depends on the traffic pattern. SA works at flit rate, thus the number of *SA_errors* is the biggest.

Table III demonstrates that when a head flit has no idle neighboring RUs available, it is necessary to treat such a situation as an RC_error. Otherwise, the absolute number of misdirected packets increases linearly with SER and the execution time increases too. This strategy induces marginal latency overhead as suggested by the results in Fig. 6.

## VI. CONCLUSIONS

In this paper, a low cost method is proposed to tolerate soft errors in the NoC router control plane. Idle routing units are utilized by neighboring input ports to provide redundant routing computation results for each packet. Soft errors in virtual channel and switch allocation results are detected by checking if the results are in legal states. Error recovery mechanism is explicitly discussed. Simulations with different soft error rates on a wormhole switched 2D mesh NoC demonstrate that our method can efficiently detect and recover soft errors in RC, VA, and SA results. In the routing units, the proposed method requires 53% less silicon cost than the TMR method when Opt_Y routing algorithm is used, and it is applicable to many more routing algorithms than the $\Sigma$ & Branch method; in the combined VA/SA units, the proposed method is simpler and more reliable than the state of the art methods. The average packet delivery latency increase is marginal when compared with the error free case.

## REFERENCES

[1] T. Bjerregaard, and S. Mahadevan, "A survey of research and practices of Network-on-Chip," DAC, pp. 684-689, 2001.

[2] W. Dally, and B. Towles, "Principles and Practices of Interconnection Networks," Morgan Kaufmann Publisher Inc., San Francisco, CA, 2003.

[3] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, "A Dynamically Adjusting Gracefully Degrading Link-Level Fault-Tolerant Mechanism for NoCs," IEEE Trans. CAD-ICS, pp.1235-1248, Aug. 2012.

[4] S. Murali, L. Benini, T. Theocharides, N. Vijaykrishnan, M. Irwin, and G. Micheli, "Analysis of Error Recovery Schemes for Networks on Chips," IEEE Design & Test of Computers, pp. 434-442, 2005.

[5] K. Wu, and D. Marculescu, "A Low-Cost, Systematic Methodology for Soft Error Robustness of Logic Circuits," IEEE Trans. VLSI Systems, vol. 21, no. 2, pp. 367-379, Feb. 2013.

[6] L. Schwiebert, and D. Jayasimha, "Optimal Fully Adaptive Wormhole Routing for Meshes," Supercomputing, pp. 782-791, Nov. 1993.

[7] C. Chen, and G. Chiu, "A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults," IEEE Trans. PDS, vol. 12, no. 5, May 2001.

[8] T. Karnik, P. Hazucha, and J. Patel, "Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes," IEEE Trans. DSC, vol. 1, no. 2, pp. 128-143, Apr.-Jun. 2004.

[9] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect to Technology Trends on the Soft Error Rate of Combinational Logic," DSN, pp. 1-10, 2002.

[10] Y. Dhillon, A. Diril, A. Chatterjee, A. Singh, "Analysis and optimization of nanometer CMOS circuits for soft-error tolerance," IEEE Trans. VLSI Systems, vol. 14, no. 5, pp. 514-524, May 2006.

[11] N. Avirneni, and A. Somani, "Low overhead soft error mitigation techniques for high-performance and aggressive designs," DSN, pp. 185-194, Jun. 2009.

[12] Q. Yu, M. Zhang, and P. Ampadu, "Exploiting Inherent Information Redundancy to Manage Transient Errors in NoC Routing Arbitration," NOCS, pp. 105-112, May 2011.

[13] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das, "Exploring Fault-Tolerant Network-on-Chip Architectures," DSN, 2006.

[14] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. Das, "Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective," ANCS, pp. 173-182, Oct. 2005.

[15] W. Liu et al., "A NoC Traffic Suite Based on Real Applications," VLSI (ISVLSI), IEEE Computer Society Annual Symposium on, Jul. 2011.

[16] S. Shirinzadeh, and R.Asli, "A Novel Soft Error Hardened Latch Design in 90nm CMOS," CADS, pp. 60-63, 2010.

[17] Y. Lu, J. McCanny, and S. Sezer, "Exploring virtual-channel architecture in FPGA based Networks-on-Chip," SOCC, pp. 302-307, Sep. 2011.