Deflection-Aware Routing Algorithm in Network on Chip against Soft Errors and Crosstalk Faults

Hadi Zamani‡, Zahra Shirmohammadi§, Ali Jahanshahi‡ ‡University of California, Riverside, CA, USA §Shahid Rajaee Teacher Training University {hzama001, ajaha004}@ucr.edu and shirmohammadi@sru.ac.ir

***Abstract*—Marching into nano-scale technology, probability of soft errors and crosstalk faults has increased by about 6-7 times. Since buffers occupy about 40-90% of the switch area, the probability of soft errors in switches is significant. We propose a deflection-aware routing algorithm (DAR) combined with an information redundancy technique to cover the soft errors and crosstalk faults in the header flow control units (FLIT). We also introduce an interleaving method along with a simple hamming code to tolerate the errors in data and tail FLITs. The proposed methods have been evaluated in both circuit and simulation level through a simulator written in C++, Booksim 2, and Synopsys Design Compiler. The evaluation results show that we can cover the soft errors and crosstalk faults with reasonable power and performance overhead of 3% and 6.5% respectively.**

***Index Terms*—Network on Chip, Soft Error, Crosstalk Fault.**

I. Introduction

As the VLSI technology further scales down, designers are able to integrate a huge numbers of processing cores into a single die. This integration of cores leads to designing many-core systems with new communication requirements. Network on chip (NoC) has become a promising and revolutionary paradigm for the communication of this unprecedented abundance of on-chip cores in the recent past years [1]. In a NoC architecture, a packet is broken down into multiple flow control units called FLITs. These FLITs are sent and received in the communication channels by the means of wires between the cores. Every FLIT arriving in any middle router is temporary stored in an input buffer until resources are freed. During FLIT transmission, accuracy of data transfer can be threatened by serious reliability concerns such as soft errors and crosstalk faults. Theses reliability concerns would become more serious with technology scaling. According to International Technology Road map for Semiconductors (ITRS) [2], most of soft errors are in Multi Bit Upset (MBU) form in the chips fabricated after 2016. Although MBUs can be appeared in the form of different patterns, the probabilities of contiguous MBU patterns would be more than the other MBU patterns [2], [3]. Moreover, data reliability can also be threatened by crosstalk fault which occurs due to coupling and inductance capacitance among adjacent wires of NoC that can result in unwanted voltage glitches, delay, and speed up in rising/falling transitions appearing on the victim wire [4] [5] [6]. To the best of our knowledge, prior research cope with soft error and crosstalk faults independently and mostly focus on the Single Bit Upset (SBU) or few patterns of Multi Bit Upset (MBU), whereas we mainly focus on the Multi Bit Upset (MBU) patterns while covering all SBUs.

In this paper, we propose a deflection-aware routing algorithm (DAR) along with a data redundancy technique to account for soft errors within the header FLIT. Moreover, split interleaving along with hamming code is also used to deal with the faults within the data and header FLITs .

The rest of the paper is organized as follows. An overview of NoC architecture and the motivation is presented in Section II. Section III and IV discusses the proposed methodology and evaluation results respectively. Sections V discusses the literature and finally conclusion remarks are given in Section VI.

II. Background and Motivation

NoC Topology: The way processing elements are connected together is called the NoC topology. The NoC topology determines performance/cost. There are different NoC topologies including star, ring, and mesh to trade performance of the NoC against its cost. In this paper, we target 8X8 2D-mesh topology since it is the most popular NoC topology [1].

NoC Routing Policy: XY routing policy that is a deterministic routing algorithm is chosen in our research which delivers high performance in a 2D-mesh topology [7].

NoC Switching Policy: NoC switching strategy determines data flow through the routers of NoC. There are different types of switching algorithms including Wormhole, store and forward, and virtual cut through. In this paper, we use wormwhole switching algorithm which is widely used in NoCs due to its low latency and small requirements at the nodes.

NoC Packet Format: Each packet is subdivided into FLITs. Each packet contains header FLIT and data FLITs. The header FLIT allows for a flexible source-directed routing scheme and data FLITs contain the information.

NoC Reliability Challenges: Soft errors [8], [9] and crosstalk [10]–[12] are among notable types of errors that occur in NoCs. Technological advances are also compounding the soft errors rates. The rate of soft errors has been increased about 6-7 times from 130nm to 30nm technology [3].

Prior research mainly focus on SBUs [13], [14], whereas ITRS estimates that most of the soft errors occurring in chips manufactured after 2016 will be in the form of multiple-bit upsets [2]. Given that 40-90% of the chip area is occupied with buffers [15], [16], there is a high chance of having MBUs in the buffers. This signifies importance of addressing soft errors, particularly MBUs. Hence, this paper focuses on crosstalk and MBUs as the major challenges towards reliability of NoCs.

III. Proposed Method

Due to importance of the header FLITs, different approaches are considered to deal with faults within the header FLITs and other types of FLITs including data and tail FLITs. Header

978-1-7281-7744-1/21/$31.00 © 2021 IEEE

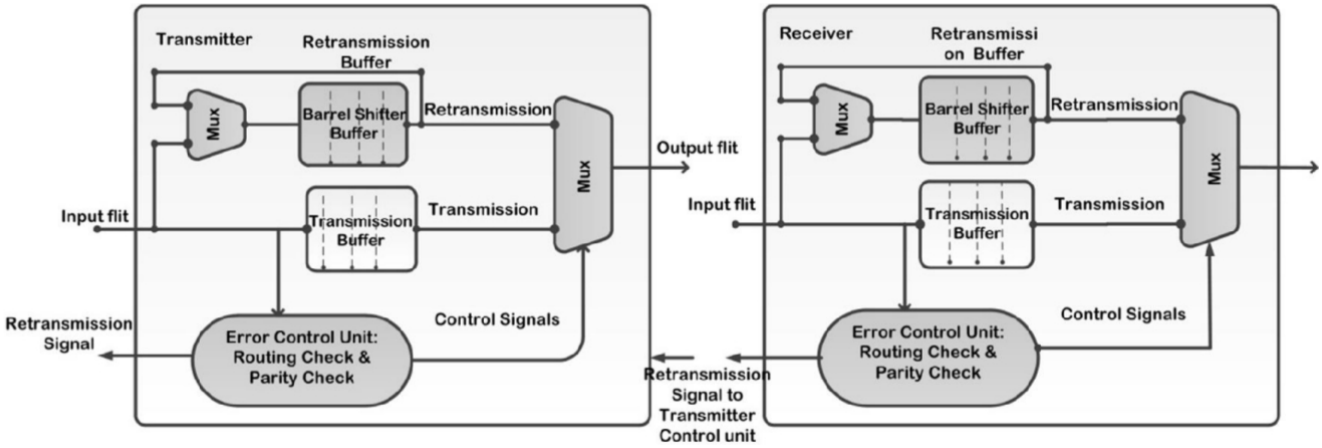

Fig. 1: Header FLIT recovery scheme.

FLITs need more protection compared to the other types of FLITs. This is because, header FLIT is used in the routing algorithm. Any faults could potentially change the routing information which results in a wrong destination.

A. Header FLIT Fault Coverage

To deal with the faults within the header FLIT, we propose DAR algorithm along with an information redundancy technique. Moreover, to address soft error and crosstalk faults within the other FLITs, we introduce an interleaving technique in conjunction with a simple information redundancy technique.

1) Header FLIT Fault Detection: There are two phases dealing with the faults within the header FLITs. *1) fault detection phase* and *2) fault recovery phase*. In detection phase, we propose a detection algorithm called Deflection-Aware Routing (DAR) which is discussed in the following:

Deflection-Aware Routing (DAR): During FLIT transmission between two adjacent nodes, a soft error in the buffer or crosstalk fault in the channel can corrupt the data and potentially lead to a wrong destination. Lets assume node 1 is the current node (source/sender) and node 2 is the next node in the deterministic routing path (next receiver node). We also assume the header FLIT is not faulty during the routing stage in node 1. However, there might be soft errors in the output buffers of node 1 or crosstalk faults in the communication channel between node 1 and node 2 before routing stage in node 2. We already calculated the routing information in node 1 and the header FLIT is on it's way to node 2. If there is a fault in header FLIT between the routing stages in node and node 2, we can not detect it in node 2 anymore and routing in node 2 could lead to a wrong destination. In other words, given faulty header FLIT, if routing is performed in node 2, it could result in a deviation from the main routing path and consequently, the next receiver node might be a wrong destination. To detect such errors which might occur following routing in node 1, we duplicate the routing algorithm. Once in node 1 (local sender) with the correct header FLIT and then, in the node 2 (local receiver) with potentially faulty header FLIT. Hence, we execute the routing algorithm in node 2 assuming that we are still in node 1. If there is any deflection in the routing, node 2, potentially, could not be the correct node anymore. So, by re-calculating routing in node 2 and assuming we are still in node 1, we can detect the error and fault recovery phase would be initiated. However, there is no guarantee that we can detect all errors. To cover the errors which are not detected by DAR, we employ a simple information redundancy scheme which is explained in the following.

Implementation Details: Given the mesh topology of our NoC and being aware of the input port of the received packet, we can calculate the transmitter address. Also, we are aware of the receiver node address. In each node, we execute both main and DAR algorithm. To eliminate performance overhead of duplicate routing, we first execute the main routing to find address of the next node in the routing path. This means, we assume there is no fault in the header FLIT which guarantees lack of performance overhead in no-faulty cases. In the next step, while the header FLIT traverses switches inside the receiver node, we initiate DAR algorithm. Based on input port of the packet at receiver node, we find address of the transmitter node and pass this address to the routing algorithm that is supposed to execute in receiver node. In other words, we duplicate the routing algorithm; one, assuming none-faulty header FLIT in transmitter and second faulty header FLIT in the following receiver node with assumption of being in the previous transmitter node. There are two scenarios. (1) The output of DAR is not equal to the current receiver node which shows the header FLIT is faulty. (2) The output of DAR is equal to address of the current receiver node. Even in case 2, header FLIT might be faulty but we were not able to detect it by DAR. This is because, even with the faulty header FLIT, receiver node is still in the correct routing path but might end up at faulty destinations in the next nodes. According to results shown in Table II, DAR algorithm is more capable of detecting complex fault patterns. It means complex fault patterns have higher chance of leading to a wrong destination.

According to extracted fault patterns shown in table I [3], faults are mostly in the form of two contiguous bits and single bit. Hence, we further employ 2-bit-parity to detect errors that were not detected by DAR. This would be enough for all single and contiguous 2-bit errors. One parity bit is used for even bits and the other parity bit is used for odds bits.

Based on trade-off between performance, power and area

overhead, DAR can be implemented in two different schemes.

Area and power aware approach: To implement the DAR algorithm while considering area and power efficiency, we can use the same router which is used for main routing algorithm. The hardware redundancies will be limited to simple address generator and comparator. Address generator produces address of previous transmitter node based on the input channel of the receiver. Comparator is used to compare address of receiver node and output of DAR algorithm. In this approach, there is no performance overhead with no faults in the system. This is because first, we execute the normal routing algorithm with assumption of no faults and after that we execute the DAR algorithm. If any faults is detected, we drop the header FLIT which is still in the current node and execute the main routing algorithm based on the header FLIT which is buffered in the transmitter node.

Performance-aware approach: In this approach, we employ an extra router which is able to perform DAR simultaneously with the normal routing algorithm. In this mechanism, beside the redundancies which were mentioned in the previous section, we need an extra router which duplicate the area, power overhead of router. Considering the area, power overhead, and negligible performance overhead of previous approach, we employ the main router for DAR as well.

2) Header FLIT Fault Recovery: Due to different fault patterns, which make it difficult to recover the original information, we store a copy of header FLIT in the spare buffer located in the transmitter node until we verify the correctness of the header FLIT in the receiver node. As shown in Figure 1, a spare buffer is used to store the header FLIT. Each time a header FLIT is arrived at a node, it is stored in both main and spare buffers. In the receiver node, we first verify correctness of the header FLIT. If any error is detected, spare FLIT stored in the spare buffer of the sender node, will be requested. During this time, header FLIT should remain in the spare buffer of the sender node. For an ideal NoC, this time is 4 cycles. This is because, routing algorithm, switch traverse, and communication channels crossing take one cycle each and one more cycle is needed for requesting a spare header FLIT. The buffer used in each node is a queue and spare buffer is a barrel shifter register which its length equals to the number of cycles of the interval time (4 cycles). As shown in the Figure 1, in faulty cases, re-transmission signal which is sent from the receiver node, fetches the FLIT from spare buffer which is not faulty. For the spare buffer, we use barrel shifter since we should re-buffer header FLIT in the spare buffer in case there is a fault again.

TABLE I: Ratio of multiple bit upsets in different technologies.

	Multiple Bit Upsets				
Design Rule (nm)	**Total**	**2**	**3**	**4-8**	**8**
180	0.5	0.5	0.0	0.0	less than 0.1
130	1.2	0.8	0.2	0.2	less than 0.1
90	1.9	1.5	0.2	0.2	less than 0.1
65	2.4	2.1	0.1	0.0	less than 0.1
45	2.3	1.9	0.2	0.1	less than 0.1
32	3.1	2.6	0.2	0.3	less than 0.1
22	3.9	3.0	0.2	0.3	0.1

B. Data FLIT Fault Coverage

Since, we are using Wormhole switching, there is only buffer space for one FLIT in each node. Hence, we should check the correctness of the data node by node and there is mechanism to check the correctness of the data FLITs end to end.

Since the error patterns are mostly in contiguous forms, without simplifying the error patterns, we need to use complicated fault detection/correction methods to coverage the data FLIT errors. We introduce an interleaving algorithm that splits the complex error patterns to simple error patterns. Dealing with the simple error patterns is easier than complex error patterns. According to the fault model introduced in [3], We have at most Eight contiguous bits that could be faulty.

As shown in Figure 2, in a typical interleaving approach, order of the input data is changed before sending the data to faulty environment that Multiple Bit Upset (MBU) might happen. For recovering, the data should be de-interleaved again. As shown in Figure 3, since buffers are more error prone, interleaving and De-interleaving is done before and after the buffers respectively. Based on the fault model, we need to cover at most 8-bit contiguous error patterns. So we interleave the data so that every two adjacent bits are stored in the buffer with minimum distance of 8 bits.

Interleaving would not be able to detect or correct the data. It only simplifies error patterns so that dealing with simplified error patterns could be done with simpler encoders and decoders with less performance, power, and area overhead. Without interleaving, we should employ a fault tolerant techniques which are able to recover the Eight contiguous error patterns which adds huge area/power and performance overhead. However, as shown in Figure 3 using interleaving along with a simple encoder such as Hamming encoder/decoder, we can handle simplified error patterns. Beside, using Matlab Simulink, we simulated the coverage capability of our introduced split interleaving along with Hamming codes in presence of different numbers of contiguous faulty bits. As shown in Figure 4, our introduced split interleaving combined with hamming code is able to cover all the contiguous faulty bits, while the well-known interleaving techniques are not suitable for the contiguous faulty bits.

As shown in Figure 3, there are Two phases in data FLIT recovery scheme. First phase: encoding and interleaving data before storing data into the buffer. And Second phase: De-interleaving and decoding the data after reading from the buffer. With this approach we can guarantee we are able to recover all contiguous and non-contiguous faulty patterns with less than maximum number of faults according to the fault model.

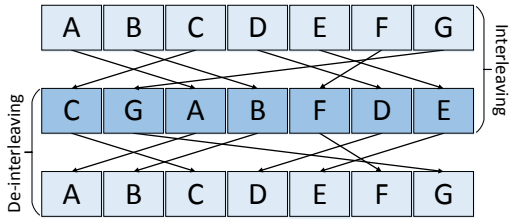

Fig. 2: Random Interleaving Scheme.

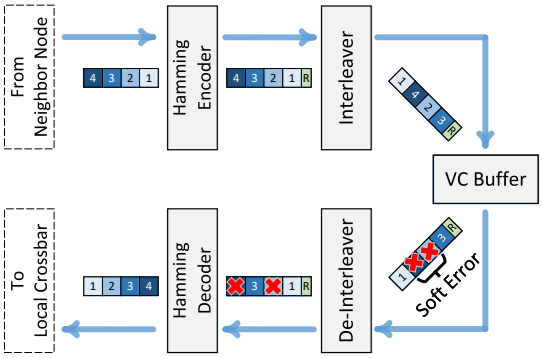

Fig. 3: Encoding/decoding along with Random Interleaving scheme.

The detailed scheme of data FLIT recovery scheme is shown in Figure 3. As illustrated in the figure, Hamming encoder and interleaver is employed before storing the data in virtual channels (buffer) and De-interleaver and Hamming decoder is employed virtual channels.

IV. Evaluation Methodology and Results

Experimental Setup: Deflection-aware routing (DAR) is evaluated in different aspects. To measure the coverage capability of DAR, a simulator is written in C++. The default routing and switching algorithms are considered XY and Wormhole respectively. XY routing algorithm is a deterministic algorithm. In deterministic routing algorithm, given the packet's destination, routing path is unique.

To evaluate performance overhead of hardware redundancies, Booksim 2 is employed [17]. Also, Synopsys Design Compiler is employed to extract the power and area overhead [18].

A. DAR Evaluation

To evaluate fault coverage capability of the DAR algorithm and other simple data redundancy techniques, a simulator is written in C++. In this simulator, network on chip with mesh topology is implemented. The source and destinations of packets are randomly chosen, and any deflection from the main routing path is detected at each hop. In a faulty

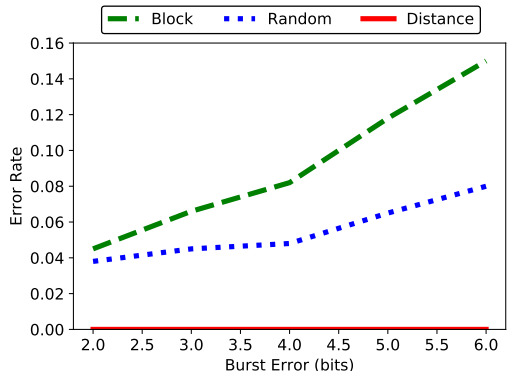

Fig. 4: Fault coverage capability of split interleaving in compared to Block and Random interleaving.

TABLE II: Fault coverage capability of different mechanisms

FT Mechanisms	Number of Injected Faults			
	1	2	3	4-8
DAR (Deflection Aware Routing)	54	67	70	74
One bit parity	100	0	100	40
Two bit parity	100	100	100	99.95
DAR+ One bit parity	100	67	100	55.6
DAR+ Two bit parity	100	100	100	99.98
DAR+ One bit parity (Total)	76.17			
DAR+ Two bit parity (Total)	99.99			

case, as mentioned earlier, there are two situations. 1) The faulty header flit is detected by employing the DAR. 2) The faulty header flit is not detected by DAR. These faults can be detected by employing simple data redundancies. According to Table I, probability of 2-bit faults are more than other faults. Single parity bit can not detect these faults. Since most of soft errors are in contiguous form, with employing 2-bit parity, we can cover most of the faults. According to our simulations, coverage probability of different schemes are estimated. As shown in Table II, DAR combined with two bit parity can detect about 99.99% of soft errors while DAR combined with single bit parity only can detect 76.17% of soft errors. Number of faults are generated according to Table I.

1) Performance Evaluation: We employ the same router which is used for the main routing algorithm. To minimize performance overhead, DAR executes after the main routing algorithm. Since other stages such as address generator and comparing phase are executed simultaneously with the main routing algorithm, they do not add performance overhead and the performance overhead comes from DAR algorithm. In NoCs, Router needs less than one cycle to execute routing algorithm [1]. Based on the main routing algorithm length, we might be able to execute the main routing and DAR in one cycle. We call this situation as merged approach. In merged approach, the DAR performance overhead is merged with the main routing algorithm. In this case, the overhead comes from the encoding/decoding phase which is less than one cycle for NoC running at 1.2 GHz as illustrated in Table IV. Using Booksim simulator, we have extracted the average packet latency in presence of different traffic patterns. As shown in Figure 5, blue dashed line shows the average latency of merged approach which is measured for traffics including Tornado, Uniform, Bitcomp, and Transpose. X-axis denotes injection rate and Y-axis denotes average packet latency. The routing delay, switching delay, virtual channel allocation are considered one cycle. The packet length and virtual channel length are 5 flits and 128 bit respectively. As shown in the Figure 5, there is about less than 3% performance overhead in presence of different injection rates in comparison to the baseline configuration.

In case, if DAR and main routing algorithm need more than one cycle to finish their execution. We call this approach

TABLE III: Area and Delay Overhead

	Delay (ns)	Area Overhead Ratio at Each Node
Address Generator	0.38	0.005
Parity Bit Generator	0.39	0.017
Parity Comparator	0.42	0.011
Address Comparator	0.16	0.003
Multiplexer	0.22	0.025

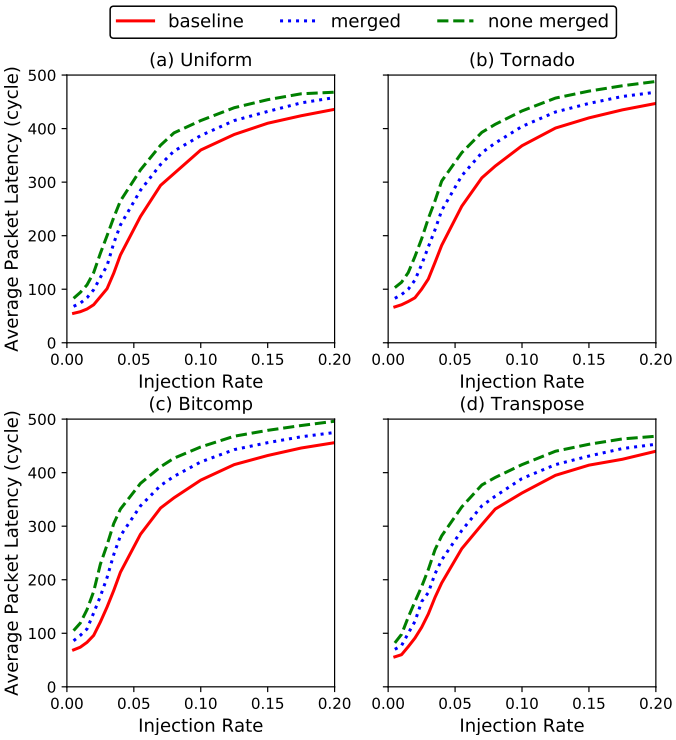

Fig. 5: Performance overhead in baseline vs. merged vs. none-merged approach implementations.

as non-merged. In this case, it adds one more cycle to the critical path. As shown in Figure 5, green dashed line shows the average packet latency in presence of different injection rates for non-merged approach. As shown in the Figure 5, in non-merged implementation, on average, there is about 6.5% performance overhead compared with the baseline.

2) Area and Delay Evaluation: In detection phase, a simple comparator, parity generator and parity detector are used. To evaluate the area, and performance overheads, all added components are implemented in VHDL and synthesized with Synopsys Design Compiler. Delay, and area overheads are measured for 22nm technology. Table III shows these evaluation results for these components. According to the results, address generator, parity comparator and address comparator can overlap with the main routing algorithm and do not imply any performance overhead to NoC pipeline.

TABLE IV: Memory, performance, and power overheads of data FLIT recovery scheme.

Flit Length (bits)	Hamming (m, n)	Delay (cycle)	Ratio of Dynamic Power	Ratio of Static Power
32	(7,4)	0.4333	0.0173	0.0045
40	(9,5)	0.4500	0.0178	0.0055
48	(10,6)	0.4750	0.0185	0.0070
56	(11,7)	0.4833	0.0192	0.0080
64	(12,8)	0.5000	0.0200	0.0095
72	(13,9)	0.5250	0.0210	0.0105
80	(14,10)	0.5417	0.0225	0.0125
88	(15,11)	0.5583	0.0230	0.0135
96	(17,12)	0.6083	0.0242	0.0170
104	(19,13)	0.6167	0.0250	0.0190
112	(19,14)	0.6250	0.0257	0.0195
120	(20,15)	0.6583	0.0262	0.0210
128	(21,16)	0.6917	0.0275	0.0225

B. Data-Flit Recovery Method Evaluation

For data FLITs, split interleaving along with data redundancy technique is exploited to deal with data FLIT errors. Split interleaver guarantees that contiguous faulty patterns are divided into separate faults which can be covered by using simple hamming encoder/decoders.

Assuming B and L as the maximum number of faults and FLIT length, to determine the types of hamming encoder/decoders, we divide the FLIT size to the maximum number of faults. If FLIT size is dividable to number of faults, we need B hamming encoder/decoder, which each of them is able to correct one fault at each data group of FLIT. The length of each group is calculated according to Equation 1.

$$splitted_{group\ length} = \lfloor \frac{L}{B} \rfloor \tag{1}$$

And if FLIT length is not dividable to number of faults, the length of one of hamming encoder/decoders is calculated according to Equation 2 and the length of other hamming encoder/decoders is calculated according to Equation 1.

$$residue_{flit\ length} = L\ mod\ B \tag{2}$$

To find redundancy bits in each group, we use Equation 3. The minimum value for r determines the number of redundancy bits for each hamming encoder/decoder.

$$2^r - r - 1 > N \tag{3}$$

So, hamming codec (N_i+r, N_i) is needed to tolerate one fault at each group of data in which N_i equals to the length calculated according to Equations 2 and 3, and r is the number of redundancy bits.

Area, and performance overheads of this approach is shown in Table IV. To find the area, power and performance overheads, the hamming codecs are implemented in VHDL and synthesized with Synopsys Design Compiler. As shown in Table IV, the performance overhead of the hamming codes are reasonable. For different hamming codes, the maximum performance overhead added to network on chip, is less than 0.83 ns or 0.69 cycle when the NoC is running at 1.2 GHz. But if the frequency is greater than 1.2 GHz and less than about 3.2 GHz, it only adds two cycles to the node pipeline. With these values, we observe the same performance overhead as shown in Figure 5.

V. Related Work

The soft and hard error and their impacts on the reliability and efficiency of the similar many-core architectures are addressed in several researches [19]–[22]. However, the reliability of communications channels in network-on-chips (NoCs) is not thoroughly addressed and improving the NoC reliability has a significant impact on state-of-art multi processor's overall performance and significantly impacts the network cost, including area and power. We can divide the NoC's reliability approaches into Three main categories: 1) Information-based redundancy techniques. 2) Hardware-based redundancy methods. and 3) Routing based methodologies. In information-based redundancy techniques, researches have used different redundancy techniques including parity, SEC-DED [23], and CRC [24]–[26] w.r.t to the fault rate and

application. Also, there have researches that explore reliability-aware design of NoC architecture [27]–[30]. For instance, Dutta Et al [27] introduce changing the router architecture to enable a low cost error correcting code to protect NoC router against single bit upset. Previous researches do not consider MBUs thoroughly and only focus on SBU and particular patterns of MBUs.

Fault tolerant routing based algorithms mostly have been proposed to cope with permanent faults. There are Two kinds of fault tolerant routing algorithms known as stochastic and deterministic routing. For example, in stochastic routing algorithms, a huge amount of packets are transferred through several paths to avoid potential faults. A probabilistic broadcast which is derived from a randomized gossip protocol is introduced by [31]. In stochastic fault tolerant routing algorithms, the bandwidth is wasted due to transfer of redundant packets which degrades the throughput of the NoC. Redundant Random Walk reduces the amount of extra packets by replicating the packets only at the source node [32]. To detect the routing deflection, an adaptive routing algorithm is introduced in [33]. The routing decision is made based on the cost function considering the route length and local fault status. However, the routing decision which is based on the only fault information of the current router, can lead to live-lock.

VI. Conclusion

In this paper, a new deflection-aware routing along with an information redundancy techniques was introduced to maintain the reliability of header FLITs. We also introduced Split interleaving combined with hamming codes to cover the data FLIT errors. Our proposed mechanisms, were evaluated in both circuit level and simulation level. A simulator was written in C++ to calculate the fault coverage capability of header flit fault detection scheme. To find the area, performance, and power overheads of hardware redundancies, we implemented and synthesized them in VHDL and Synopsys Design Compiler respectively. The evaluation results show that the proposed methods achieve high reliability while maintaining performance requirements.

References

[1] S. Kumar, A. Jantsch, J. . Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, 2002.

[2] ITRS, "International technology roadmap for semiconductors," http://www.itrs2.net/itrs-reports.html, May 2015.

[3] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on Neutron-Induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, 2010.

[4] Z. Shirmohammadi, H. Z. Sabzi, and others, "3D-DyCAC: Dynamic numerical-based mechanism for reducing crosstalk faults in 3D ICs," *2017 IEEE International*, 2017.

[5] Z. Shirmohammadi, M. Taali, and H. Z. Sabzi, "InduM: An accurate probablity inductance-based model to predict delay in chips," in *International Conference on Computer and Knowledge Engineering (ICCKE)*, 2019.

[6] Z. Shirmohammadi and H. Z. Sabzi, "DR: Overhead efficient RLC crosstalk avoidance code," *2018 8th International*, 2018.

[7] S. Pasricha and N. Dutt, *On-Chip Communication Architectures: System on Chip Interconnect*. Morgan Kaufmann, Jul. 2010.

[8] M. J. Gadlage, A. H. Roach, A. R. Duncan, A. M. Williams, D. P. Bossev, and M. J. Kay, "Soft errors induced by high-energy electrons," *IEEE Transactions on Device and Materials Reliability*, vol. 17, no. 1, pp. 157–162, 2017.

[9] J.-L. Autran and D. Munteanu, *Soft Errors: From Particles to Circuits*. CRC Press, Dec. 2017.

[10] Z. Shirmohammadi and S. G. Miremadi, "Addressing NoC reliability through an efficient Fibonacci-Based crosstalk avoidance codec design," in *Algorithms and Architectures for Parallel Processing*. Springer International Publishing, 2015, pp. 756–770.

[11] ——, "On designing an efficient numerical-based forbidden pattern free crosstalk avoidance codec for reliable data transfer of nocs," *Microelectron. Reliab.*, vol. 63, pp. 304–313, 2016. [Online]. Available: https://doi.org/10.1016/j.microrel.2016.03.031

[12] ——, "Using binary-reflected gray coding for crosstalk mitigation of network on chip," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*. IEEE, 2013, pp. 81–86.

[13] M. Zhang, Z. Guo, and W. Xu, "An adaptive single event upset (SEU)-Hardened Flip-Flop design," in *2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*. ieeexplore.ieee.org, Jun. 2019, pp. 1–3.

[14] M. Berg, K. LaBel, M. Campola, and M. Xapsos, "Analyzing system on a chip single event upset responses using single event upset data, classical reliability models, and space environment data," 2017.

[15] Jingcao Hu and R. Marculescu, "Dyad - smart routing for networks-on-chip," in *Proceedings. 41st Design Automation Conference, 2004.*, 2004, pp. 260–263.

[16] J. Hu and R. Marculescu, "DyAD," 2004.

[17] Nan Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," pp. 86–96, 2013.

[18] Synopsys, "Rtl design and synthesis," https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test.html, May.

[19] D. Tripathy, A. Abdolrashidi, L. N. Bhuyan, L. Zhou, and D. Wong, "Paver: Locality graph-based thread block scheduling for gpus," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 3, pp. 1–26, 2021.

[20] H. Zamani, Y. Liu, D. Tripathy, L. Bhuyan, and others, "GreenMM: energy efficient GPU matrix multiplication through undervolting," *Proceedings of the ACM*, 2019.

[21] H. Zamani, D. Tripathy, L. Bhuyan, and Z. Chen, "SAOU: safe adaptive overclocking and undervolting for energy-efficient GPU computing," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 205–210.

[22] D. Tripathy, H. Zamani, D. Sahoo, L. N. Bhuyan, and M. Satpathy, "Slumber: static-power management for GPGPU register files," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 109–114.

[23] R. Sharifi and Z. Navabi, "Online profiling for cluster-specific variable rate refreshing in high-density dram systems," in *2017 22nd IEEE European Test Symposium (ETS)*. IEEE, 2017, pp. 1–6.

[24] A. Berman and I. Keidar, "Low-overhead error detection for Networks-on-Chip," 2009.

[25] K. A. S. S. Lakshmi, Kandala Anupama Satya, A. M. Keerthi, K. M. Sri, and M. Vinodhini, "Code with crosstalk avoidance and error correction for network on chip interconnects," 2020.

[26] C.-L. Li, Y.-W. Kim, Y. S. Lee, and T. H. Han, "Power-efficient error-resilient network-on-chip router using selective error correction code scheme," pp. 1368–1370, 2018.

[27] A. Dutta and N. A. Touba, "Reliable Network-on-Chip using a low cost unequal error protection code," in *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, Sep. 2007, pp. 3–11.

[28] M. G. Moghaddam, "Dynamic energy and reliability management in network-on-chip based chip multiprocessors," 2017.

[29] B. Bhowmik, J. K. Deka, and S. Biswas, "Improving reliability in spidergon network on Chip-Microprocessors," 2020.

[30] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das, "Exploring fault-tolerant network-on-chip architectures," in *International Conference on Dependable Systems and Networks (DSN'06)*, 2006, pp. 93–104.

[31] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *ASP-DAC 2003.*, Jan. 2003, pp. 225–232.

[32] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *IEEE Computer Society Annual Symposium on VLSI*.

[33] A. Kohler, G. Schley, and M. Radetzki, "Fault tolerant network on chip switching with graceful performance degradation," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 29, no. 6, pp. 883–896, Jun. 2010.