Fault Tolerant FIFO Design for NOC Router using Low Complexity Voter

R.Meena[1], D.L.Jayanthi, M.Tech. [2]
PG Scholar, Assistant Professor,
Department of ECE,
AVS Engineering College, Salem

.

Abstract: The emerging technique for communication with in a large VLSI system is a Network On Chip. The fast scaling of technique there has been susceptible faults in the component of the Network On Chip, thus there is a requirement for technique to maintain circuit reliability. A fault-tolerant NOC (Network-on-chip) should be having the capacity to detect a fault and recover the system to correctly operate and work according to the mapped application. Deflection routing is a promising approach for energy and hardware efficient NOCs. The inherent redundancy of NOCs can be used to tolerate such failures. Redundancy techniques in NOC are implemented widely to increase the reliability, especially the TMR - Triple Modular Redundancy. This project proposes a simple but effective fault tolerant voter circuit for NOC which is more reliable and less expensive. Experimental results demonstrate its improvement over the former TMR structures. Proposed system has been coded in Verilog HDL and simulated using Xilinx 12.1.

Keywords: TMR, VLSI, NOC, Xilinx

I INTRODUCTION

As semiconductor technology improves, the number of processing cores integrated on a single chip has continually increased. More recently, commercial or prototype chips that have 64 or more processing cores have already been produced. To connect such many cores, Network-on- Chips (NoCs) that introduce a packet-switched network structure have been widely employed instead of traditional shared-bus-based on-chip interconnects.

On arbitered-shared-bus, a communication bottleneck appears when more than one component needs to use the bus. It is not scalable when increasing the number of components requiring to use the same bus, which has a limited bandwidth. NoC architectures are similar to those used in parallel computers and System Area Networks (SANs). A 2x2 NoC, which consists on a set of routers interconnected between them. In these networks, source nodes generate packets that include headers as well as data, then routers transfer them through connected links to other routers, and final destination nodes decompose them. Since different packets can be simultaneously transferred on multiple links, bandwidth of NoCs is much larger than that of buses when interconnecting a large number of cores in Multi-Processors Systems on Chip (MPSoCs).

We will refer two routers as *neighbor routers* when they are interconnected directly to each other. Each router on a NoC can send and receive messages from each of their *neighbor routers* or its local cores by using private point to point buses (referred as links). Routers are also responsible to route messages to their *neighbor routers* or *local cores*, or may store them on buffers.

However, the occurrence of system failures may compromise the correct operation of the NoC. Those are due to the increasing design complexity of chips, their need on design reliability due to the increasing integration scale (reducing the size of transistors induces variations and irregularities on them), and finally also signal errors due to the reduction on their voltage work level (reducing the threshold between logic levels).

Thus, it becomes necessary to provide fault-tolerance features, that were once exclusive to critical systems, and now become necessary even in consumer electronics. In recent years, various techniques have been proposed for providing fault tolerate on NOCs. This paper will undertake a study of existing techniques, indicating which faults they support, their strengths and weaknesses. Use of NoCs in future systems.

II. BASIC CONCEPTS ON FAULT TOLERANCE

A. Fault Models

Faults can be classified as *permanent* and *transient* depending on its duration along the time. A transient fault disappears after a short interval of time. For example, it can produce a bit-flip which corrupts the header or the payload of a packet. For such type of faults, error control can be implemented at either link level or end-to-end level.

Routers at the ends of a link work together to deal with transient faults. Each router stores and checks the incoming flit before forwarding it to the next router. Alternatively, error control can be implemented at the end-to-end level, that is, at the end nodes themselves. Transient faults are usually modeled with a Bit-Error Rate.

On the other hand, *permanent faults* do not disappear over the time. For instance, permanent faults can be related to permanent damages of circuits or wires.

B. Basic Concepts

The original definition of dependability is the ability to deliver service that can justifiably be trusted. We will refer to the threats to this system dependability as *faults*. The important concepts of fault tolerance and system failure are defined as:

A fault is tolerated when the system will keep working correctly in its presence. Alternatively, we will refer as system failure when the system cannot deliver service on presence of faults, i.e. fault is not being tolerated.

The solutions to avoid threats to the system dependability become into a system failure, can be classified as:

i) Fault prevention means to prevent the occurrence or introduction of faults;

ii) Fault tolerance means to avoid service failures in the presence of faults;

iii) Fault removal means to reduce the number and severity of faults, and

iv) Fault forecasting means to estimate the present number, future incidence, and the consequences of faults

In this paper we will focus only on describing different proposed techniques for fault tolerance. The last term we define in this section, is the *fault tolerance degree* for characterizing the fault tolerant capabilities of the proposed mechanisms, and for comparison purposes between different fault tolerant techniques. Fault tolerance degree corresponds to the maximum number of faulty components in the network that can be tolerated simultaneously by the system (the network is still working providing the same connectivity, that is, there is at least a path for every source-destination pair).

III. EXISTING SCHEMES

The algorithmic interpretation of the transparent SOA-MATS++ test is presented in Algorithm 1. It describes the step-by-step procedure to perform the three phases of the transparent SOA-MATS++ test for each location of the FIFO memory. The target location for test is given by the loop index i that varies from 0 to $N - 1$, where N is the number of locations in the FIFO memory. In other words, I represents the address of the FIFO memory location presently under test. For each location, the three test runs are performed during three iterations of the loop index j.

For a particular FIFO memory location (present value of i), the first iteration of j (address run1) performs the invert phase, where the content of the FIFO location is inverted. The invert test phase involves reading the content of *lut* into a temporary variable *temp* and then backing it up in *original*. Then, the inverted content of *temp* is written back to *lut*. At this point, the content of *lut* is inversion of content of *original*.

In the next iteration of j (address run2), the restore phase is performed. The content of *lut* is reread into *temp* and compared with the content of *original*. The comparison should result in all 1's pattern. However, deviation from the all 1's pattern at any bit position indicates fault at that particular bit position. Next, the inverted content of *temp* is written back to *lut*. Thus, the content of *lut*, which were inverted after the first iteration get restored after the second.

The third iteration of *j* performs only a read operation of *lut*, where the content of *lut* is read into *temp* and compared with the contents of *original*. At this stage of the test, all 0's pattern in the result signifies fault free location, while deviation at any bit position from all 0's pattern means fault at that particular bit position. The last read operation ensures the detection of faults, which remained undetected during the earlier two test runs. At the end of the three test runs (iterations of *j*), the loop index *i* is incremented by one to mark the start of test for the next location.

Fault Coverage of the Algorithm

The transparent SOA-MATS++ algorithm is intended for test of stuck-at fault, transition fault, and read disturb fault fault tests developed during field operation of FIFO memories. The fault coverage of the algorithm is shown in Fig. 1. In both the figures, the word size of FIFO memory is assumed to be of 4 bits. The text in italics against the arrows indicates the operation performed, while the text in bold font corresponds to the variables used in Algorithm 1.

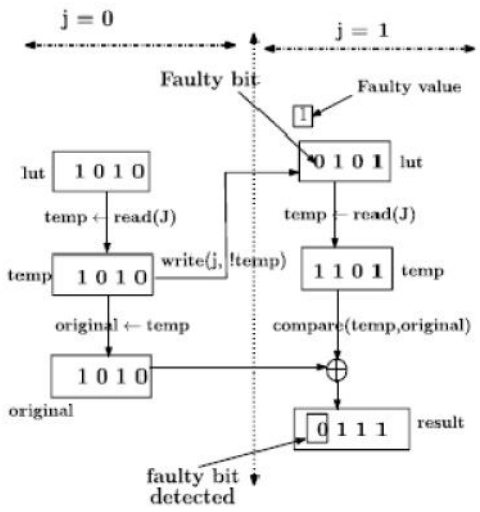

Fig.1. algorithm of SOA-MATS++

As shown in Fig.1, assume the data word present in *lut* be 1010. The test cycles begin with the invert phase (memory address pointer *j* with 0 value) during which the content of location addressed is read into *temp* and then backed up in the *original*. The data written back to *lut* is the complement of content of *temp*. Thus, at the end of the cycle, the data present in *temp* and *original* is 1010, while *lut* contains 0101. Assume a stuck-at-1 fault at the most significant bit (MSB) position of the word stored in *lut*. Thus, instead of storing 0101, it actually stores 1101 and as a result, the stuck-at-fault at the MSB gets excited.

During the second iteration of *j*, when *lut* is readdressed, the data read into *temp* is 1101. At this point, the data present in *temp* and *original* are compared (bitwise XORed). An all 1's pattern is expected as result. Any 0 within the pattern would mean a stuck-at fault at that bit position. This situation is shown in Fig, where the XOR of 1010 and 1101 yields a 0 at the MSB position of the *result* indicating a stuck-at-fault at the MSB position. However, for cases where the initial data for a bit position is different from the faulty bit value, the stuck-at-fault cannot be detected for the bit position after the restore phase of the test. It thus requires one more test cycle to excite such faults.

IV. PROPOSED SYSTEM

Nanoelectronic systems are now more and more prone to faults and defects, permanent or transient. Redundancy techniques are implemented widely to increase the reliability, especially the TMR - Triple Modular Redundancy. However, many researchers assume that the voter is perfect and this may not be true. This paper proposes a simple but effective faulttolerant voter circuit which is more reliable and less expensive.

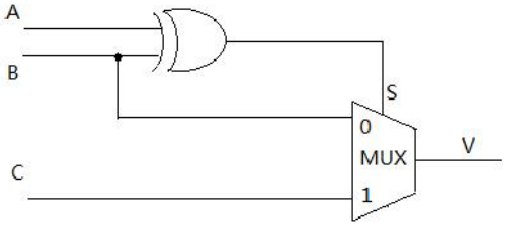

Fig.2 proposed voter

In this work we propose a new scheme for majority voting presented in figure 6. This circuit is based on few logic blocks, an exclusive or gate and a multiplexer. We can see that this structure is fault-tolerant following the analysis below:

a) A fault occurs on the voter: The unique internal node *S* is stuck to a fault. Due to the single-fault model, the output is correct, independent of the value *S* (*A* = *B* = *C*)

b) A fault occurs on one of the modules M:

• Case 1 (*C* is faulty): the logic signals *A* and *B* are the same and *S* = *A* *B* = 0. So, the multiplexer's output will be signal *B* which represents the majority value.

• Case 2 (*A* or *B* is faulty): the logic signals *A* and *B* are different and the majority must be the logic value given by *AC* + *BC*, which corresponds to logic value

C. According to the scheme, this relation is respected because *S* = *A* *B* = 1.

The proposed voter structure shows a better performance than the previous schemes. Compared to Kshirsagar's voter, obviously it consumes less area overhead together with power dissipation and also there is less delay (and consequently also better than the classical structure), and these parameters are all important characteristics for a voter. These features and improvements come directly from the simplicity of the architecture.

IV. EXPERIMENTAL RESULTS

The proposed circuit are simulated and synthesized by using modelsim and xilinx12.1 which occurs low area than the existing. The experimental results are given in Table2 and the simulation results of layout and the waveforms are shown in the fig.3 and fig.4 shows the RTL schematic of proposed system. Then the synthesis result of the proposed are shown in fig.6.

Fig. 3 Simulation Result

s.no	Parameter	Existing	Proposed
1	Slice	19	11
2	lut	36	21
3	Flip flop	13	12

Table 1: comparison table

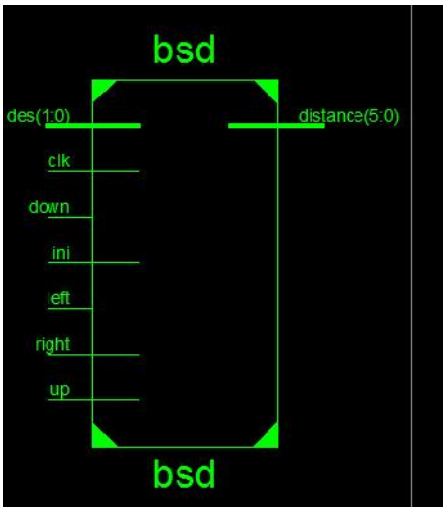

Fig.4 RTL schematic of proposed system

V. PERFORMANCE ANALYSIS

The Figure given below is shown that there is a considerable reduction based on no of transistors and the performance chart has been shown below in fig.5

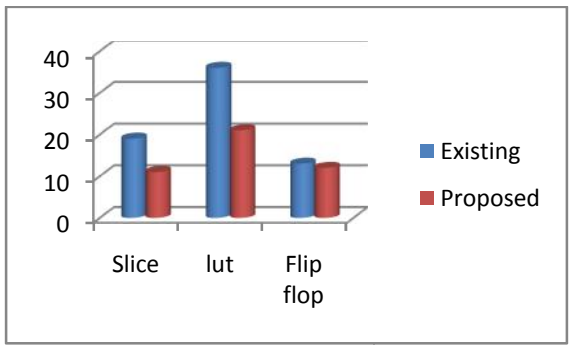

Fig.5 performance chart

VI. CONCLUSION

This work presented a digital voter circuit to be used in TMR schemes for NOC router. This novel approach has a simple architecture, saves area, has a low power consumption and less propagation delays. It is also a robust single fault solution that exceeds the reliability of current solutions in presence of permanent faults occurred in NOC components. Future work could expand to how to formulate an algorithm to design a voter circuit for NMR systems extends beyond TMR which has the same property as this.

REFERENCES

[1] U. Afzaa , " FPGA-based Design of a Self-Checking TMR Voter",2017 .

[2] P. K. Balasubramanian , "A Fault Tolerance Improved Majority Voter for TMR System Architectures" ,2016 .

[3] J. F. T. Olano , "Exploring the Use of Multiple Modular Redundancies for Masking Accumulated Faults in SRAM-based FPGAs",2014.

[4] Y. Thonnart, P. Vivet, and F. Clermidy, "A fully-asynchronous low-power framework for GALS NoC integration," in Proc. Design, Autom. Test Europe Conf. Exhib., Dresden, Germany, 2013., pp. 33–38.

[5] X. Y. Li and O. Hammami, "An automatic design flow for data parallel and pipelined signal processing applications on embedded multiprocessor with NoC: Application to cryptography," Int. J. Reconfig. Comput., p. 631490, Jan. 2013.

[6] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault modeling and simulation for crosstalk in system-on-chip interconnects," Proc. IEEE/ACM Int. Conf. Comput. -Aided Des., pp. 297–303, 2013.

[7] S. Y. Jiang, Y. Liu, J. B. Luo, and H. Cheng, "Study of fault-tolerant routing algorithm of NoC based on 2D-mesh topology," in IEEE Int. Conf. Appl. Supercond. Electromagn. Devices, Beijing, China, 2013, pp. 189–193.

[8] L. P. Carloni, P. Pande, and Y. Xie, "fs," in Proc. 3rd ACM/IEEE Int. Symp. Netw. Chip, New York, NY, USA, 2011, pp. 93–102.

[9] D. Li et al., "FiConn: Using backup port for server interconnection in data centers," in Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, 2010, pp. 2276–2285.

[10] S. R. Hasan, N. Bélanger, Y. Savaria, and M.O. Ahmad, "Crosstalk glitch propagation modeling for asynchronous interfaces in globally asynchronous locally synchronous systems," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 8, pp. 2020–2031, Aug. 2010.

[11] A. T. Clements, M. F. Kaashoek, and N. Zeldovich, "Scalable address spaces using RCU balanced trees," in Proc. 17th Int. Conf. Architectural Support Programm. Lang. Oper. Syst., London, U.K., 2010, pp. 199–210.

[12] V. Tipparaju, E. Aprà, W. K. Yu, and J. S. Vetter, "Enabling a highly-scalable global address space model for petascale computing," in Proc. 7th ACM Int. Conf. Comput. Frontiers, Bertinoro, Italy, 2009, pp. 207–216.

[13] A. K. Singh, J. G. Wu, A. Prakash, and T. Srikanthan, "Efficient heuristics for minimizing communication overhead in NoC-based heterogeneous MPSoC patforms," in Proc. IEEE/IFIP Int. Symp. Rapid Syst. Prototyping, Paris, France, 2009, pp. 55–60.

[14] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 28, no. 1, pp. 3–22, Jan. 2009.

