International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-7, Issue No.-4, July, 2019

High Speed In-Field Test for Faults in FIFO Buffers of NoC Routers

K Sharon Kamal
Electronics and Communication Engineering
M.V.S.R Engineering College
Hyderabad, India
E-mail: ksharonkamal07@gmail.com

V Rajitha Rani
Electronics and Communication Engineering
M.V.S.R Engineering College
Hyderabad, India
E-mail: rajitharani.ece@gmail.com

Abstract: First-in-First-out (FIFO) buffers are getting progressively well-liked as buffer store between subsystems. On continuous usage they result in faults and failure. Over the time the intermittent faults become permanent faults. A test procedure with pipelining is enforced on the information in FIFO buffers. The test detects the faults thereby reducing the delay exploitation through pipelining. The performance of pipelining is measured in terms of delay in conjunction with area by synthesizing the circuit.

Keywords: FIFO Buffers, permanent faults, pipelining, intermittent faults

I. INTRODUCTION

Network-on-chip (NoC) has emerged as a far better communication compared with bus-based communication network for advanced chip styles overcoming the difficulties associated with information measure, signal integrity, and power dissipation [2]. However, like all alternative systems-on-a-chip (SoCs), NoC-based SoCs should even be tested for defects. Testing the NOC infrastructure involves testing routers and interrouter links.

Most of space of the NOC transport medium is occupied by routers, that is preponderantly occupied by FIFO buffers and routing logic. consequently, the possibilities of run-time faults or defects occurring in buffers and logic measure considerably higher compared with the other elements of the NOC. Thus, check method or test for the NOC infrastructure should begin with check of buffers and routing logic of the routers. additionally, the check should be performed sporadically to confirm that no fault gets accumulated.

The occasional run-time useful faults have been one of the key issues throughout testing of deeply scaled CMOS-based reminiscences. The faults are a because of physical effects, like environmental status, aging, and low provide voltage and thus intermittent (nonpermanent indicating device injury or malfunction) in nature [3]. However, these intermittent faults sometimes exhibit a comparatively high prevalence rate and eventually tend to become permanent [3]. Moreover, wear-out of reminiscences conjointly cause intermit- tent faults to become frequent enough to be classified as permanent. Thus, there's a desire for on-line check technique which will find the run-time faults, that intermittent in nature however bit by bit become permanent over time[1].

In this temporary, we've implemented a clear test technique for First-In First-Out (FIFO) buffers among the routers of the NOC infrastructure.

A clear SOA-MATS++ test generation formula has been enforced targeting in-field permanent faults developed in SRAM- based mostly FIFO buffers. additionally, we've conjointly projected the inclusion of pipelining to bring a amendment within the latency.

Pipelining: Pipelining is a method of rising the general process performance of a processor. This field of study approach permits the coincident execution of many directions. Pipelining is clear to the programmer; it exploits similarity at the instruction level by overlapping the execution method of directions. it's analogous to an assembly line wherever employees perform a particular task and pass the partly completed product to consequent employee.

Fault Models: The run-time permanent faults thought-about during this temporary assumed to be intermittent faults, that became permanent over time. Consequently, the fault models thought-about in this temporary square measure that of intermittent faults. the first factors that result in intermittent faults square measure aging effects, like time-dependent dielectric breakdown (TDDB), electromigration, negative bias temperature instability (NBTI), and hot carrier injection (HCI), as mentioned in [4]. TDDB a phenomena wherever the compound beneath the gate material of associate MOSFET degrades over time leading to a short circuit, that square measure shapely as stuck-at-faults [5]. Electromigration reduces interconnect physical phenomenon with passage of time and ends up in open circuit [5]. The open circuits caused by electromigration square measure shapely as stuck-open-faults. NBTI and HCI increase the edge voltage of transistors resulting in decrease in mobility. As a result, the performances of the memory core decreases transportation in read or scan and write failures. The write failures square measure shapely as transition faults, whereas scan failures square measure shapely as read or scan disturb faults [6].

To summarize, the target fault models thought-about for this temporary square measure stuck-at faults. elaborate behavior of those faults is found in [7].

II. LITERATURE SURVEY

As fault tolerance in NOC style has gained importance among analysis community, variety of papers are revealed covering totally different aspects of fault tolerance, like failure mechanisms, fault modeling, diagnosis, and so on. an in depth survey summarizing the analysis add these papers has been provided in [3]. Over the years, researchers have projected variety of Design- For-Testability (DFT) techniques for NOC infrastructure testing (test- ing routers likewise as NOC interconnect) [8] and for NOC based mostly core testing [8]. Built-in self check (BIST)-based techniques have conjointly been used for testing routers likewise as NOC interconnect, such as [9]. A recent paper on NOC and router testing in [10] provides a outline of the DFT techniques utilized for testing NOC interconnects and routers specially.

FIFO buffers in NOC infrastructure square measure massive in variety and unfold everywhere the chip. Consequently, chance of faults is considerably higher for the buffers compared with alternative elements of the router. Each on-line and offline check techniques are projected for check of FIFO buffers in NOC. The proposal in [10] is associate offline check technique (suitable for the detection of producing fault in FIFO buffers) that proposes a shared BIST controller for FIFO buffers. On-line check techniques for the detection of faults in FIFO buffers of NOC routers are projected in [11]. However, the technique considers normal cell-based FIFO buffers, whereas we tend to take into account SRAM-based FIFO styles. Thus, faults thought-about during this temporary square measure totally different from those targeted in [11].

III. ALGORITHM OF TRANSPARENT SOA MAT++

The faults thought-about if applied for SRAMs or DRAMs, is detected by normal March tests [7]. However, if identical set of faults square measure thought-about for SRAM-type FIFOs, March test can not be used directly because of the address restriction in SRAM-type FIFOs mentioned in [12] and therefore we tend to were intended to decide on single-order address MATS test (SOA-MATS++) [12] for the detection of faults thought-about during this temporary. The word-adjusted SOA-MATS++ test is delineate as {( );(ra, wb) ;(rb, wa);(ra)} wherever, a is that the information or data background and b is that the complement of the information or data background. and square measure increasing and decreasing addressing order of memory, severally. suggests that memory addressing is increasing or decreasing. Application of SOA-MATS check to the FIFO involves writing patterns into the FIFO memory and reading them back. As a result, the memory contents square measure destroyed. However, on-line memory check techniques need the restoration of the memory contents once check. Thus, researchers have changed the March tests to clear March test [13] in order that tests is performed while not the need of external information background and therefore the memory contents is remodeled once test. We have therefore reworked the SOA-MATS++ test to trans- parent SOA-MATS++ (TSOA-MATS++) check that will be applied for on-line check of FIFO buffers. The clear SOA-MATS check generated is delineate as{ (rx, w , r ,wx, rx)} .

The operations performed throughout the test represent 3 phases of the test, namely, invert section, restore section, and read section. the primary 2 operations kind a scan write try (rx, wx) representing the invert section wherever the initial content (content before begin of check) of the FIFO buffer location under test (lut) is scan and its complement is written back to same location. The invert section is followed by restore section involving the operations (r , wx), wherever the content of lut square measure scan and reinverted. At this time of the check, the contents of lut are flipped doubly to urge back the first content. The last section, (rx) involves reading the content of lut with none write operation to follow.

Test Algorithm: The recursive interpretation of the transparent SOA-MATS++ test is conferred in algorithm. It describes the stepwise procedure to perform the 3 phases of the transparent SOA-MATS++ test for every location of the FIFO memory. The target location for test is given by the loop index i that varies from 0 to N − 1, wherever N is that the variety of locations within the FIFO memory. In alternative words, i represent the address of the FIFO memory location presently underneath check or test. for every location, the 3 test runs are performed throughout 3 iterations of the loop index j.

Algorithm 1 Transparent SOA-MATS++ Test Algorithm

```
Require: N = number of rows of the FIFO memory
1: i ← 0;                                   /* memory address pointer */
2: while (i ≤ N − 1) do
3:    j ← 0;                                /* test cycle counter */
4:    while (j ≤ 2) do
5:       temp ← read(i);
6:       if (j = 0) then
7:          original ← temp;
8:          write(i, !temp);
9:       else
10:         if (j = 1) then
11:            result ← compare(temp, original);
12:            write(i, !temp);
13:         end if
14:      else
15:         result ← compare(temp, original);
16:      end if
17:      j ← j + 1;
18:   end while
19:   i ← i + 1;
20: end while
```

For a specific FIFO memory location (i), the 1st iteration of j (address run1) performs the invert section, wherever the content of the FIFO location is inverted. The invert check or test section involves reading the content of lut into temporary variable temp and copying it in original. Then, the inverted content of temp is copied into lut. At this time, the content of lut is inversion of content of original.

In the next iteration of j (address run2), the restore section is performed. The content of lut is read into temporary and compared with the content of original. The comparison ought to end in all 1's pattern. However, deviation from the all 1's pattern at any bit position indicates fault at that specific bit position. Next, the inverted content of temporary worker is written back to lut. Thus, the content of lut, that were inverted once the primary iteration get remodeled once after the second. The third iteration of j performs solely a scan operation of lut, wherever the content of lut is scan into temp and compared with the contents of original. At this stage of the check, all 0's pattern within the result signifies fault free location, whereas deviation at any bit position from all 0's pattern suggests that fault at that specific bit position. The last scan operation ensures the detection of faults, that remained undiscovered throughout the sooner 2 check runs. At the tip of the 3 check runs (iterations of j), the loop index i is incremented by one to mark the beginning of check for consequent location.

Example for detecting the fault: The transparent SOA-MATS++ formula is meant for check of stuck-at-fault developed throughout field operation of FIFO buffers. An example of fault detection is shown in Fig. 1. In each the figures, the word size of FIFO memory is assumed to be of four bits.

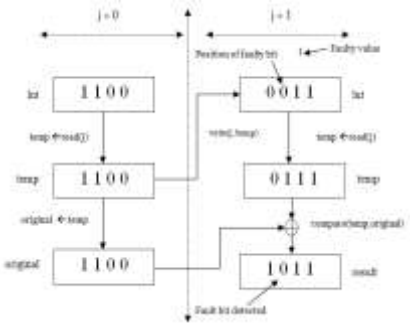

Figure 1. Transparent SOA-MAT++ fault detection

As shown in Fig. 1, assume the information word in lut be 1100. The check cycles begin with the invert section (memory address pointer j with zero value) throughout that the content of location self-addressed is scan into temp and secured within the original. the information written back to lut is that the complement of content of temp. Thus, at the end of the cycle, the information t in temp and original is 1100, whereas lut contains 0011. Assume a stuck-at-1 fault at (MSB-1) position of the word in lut. Thus, rather than storing 0011, it truly stores 0111 and as a result, the stuck-at-fault at the MSB-1 position gets excited.

During the second iteration of j, once lut is readdressed, the information scan into temp is 0111. At this time, the information word in temp and original are compared (bitwise XORed). associate all 1's pattern is predicted as result. Any zero among the pattern would mean a stuck-at fault at that bit position. this example is shown in Fig. 1, wherever the XOR of 1100 and 0111 yields a 0 at the MSB-1 position of the result indicating a stuck-at-fault at the MSB position. However, for cases wherever the initial knowledge for a small amount position is totally different from the faulty bit price, the stuck-at-fault can not be detected for the bit position once the restore section of the check. It therefore needs an additional check cycle to excite such faults.

IV. PIPELINE STRUCTURE AND PERFORMANCE

The pipeline style technique decomposes a consecutive method into many sub processes, known as stages or segments. A stage performs a selected function and produces an intermediate result. It consists of input latch, additionally known as a register or buffer, followed by a process circuit. (A process circuit will be a combinatory or consecutive circuit.) The process circuit of a given stage is connected to the input latch of consecutive stage (see Fig 2).

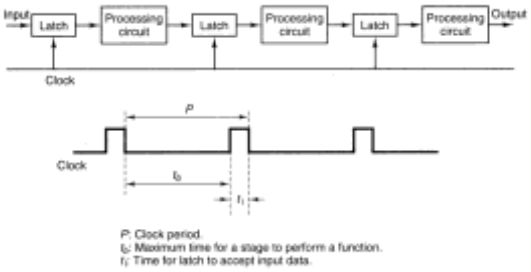

Figure 2. Structure of Pipeline

A clock signal is connected to every input latch. At every clock pulse, each stage transfers its intermediate result to the input latch of consecutive stage. during this method, the ultimate result's created once the computer file have suffered the whole pipeline, finishing one stage per clock pulse. the amount of the clock pulse ought to be massive enough to produce sufficient time for an indication to traverse through the slowest stage, that is named the bottleneck (i.e., the stage needing the longest quantity of your time to complete). additionally, there ought to be enough time for a latch to store its input signals. If the clock's amount, P, is expressed as $P = tb + tl$, then *tb* ought to be bigger than the utmost delay of the

bottleneck stage, and *tl* ought to be sufficient for storing information into a latch.

The ability to overlap stages of a serial method for various input tasks (data or operations) ends up in AN overall theoretical completion time of

$$Tpipe = m*P + (n-1)*P,$$

where n is that the range of input tasks, m is that the range of stages within the pipeline, and P is that the clock amount. The term m*P is that the time needed for the primary input task to urge through the pipeline, and also the term (n-1)*P is that the time needed for the remaining tasks. once the pipeline has been stuffed, it generates AN output on every clock cycle. In alternative words, once the pipeline is loaded, it'll generate output solely as quick as its slowest stage. Even with this limitation, the pipeline can greatly surpass nonpipelined techniques, that need every task to finish before another task's execution sequence begins. To be a lot of specific, once n is massive, a pipelined processor will manufacture output about m times quicker than a nonpipelined processor.

V. IMPLEMENTATION OF THE TEST ON FIFO BUFFERS

Here we present the technique used for implementing the projected transparent SOA-MAT++ test. The buffers operate in 2 modes, the traditional mode and therefore the check or test mode. The FIFO buffers are allowed to be operative in traditional mode for sufficient quantity of your time before initiating their check method. This delay in check initiation provides sufficient time for run-time intermittent faults developed in FIFO buffers to rework into permanent faults. The switch of FIFO buffers from traditional mode to check mode happens once a precise amount of your time while not caring regarding the current state of the FIFO buffer. it's going to be argued that at the moment of switch, the buffer might not be full, and as a result not all locations would be checked throughout the test cycle. However, check initiation once the buffer gets full would cause the subsequent issues.

A check burst involves series of check read and write cycles. Periodic checking of a FIFO buffer permits check of a special set of locations of the FIFO buffer in every test burst. when the buffer is switched to check or test mode, the traditional method gets interrupted. The FIFO memory location presently self-addressed in traditional mode, at the instant of switch, becomes the target location for check. Since traditional operation is interrupted at totally different instants in several check bursts, the locations tested in every burst would vary. Thus, continuation the check bursts for variety times on a FIFO buffer would allow the check of every location as the amount of locations during a FIFO buffer is few. Moreover, periodic testing prevents accumulation of fault within the buffer.

Architecture of the Test Circuit: The FIFO buffer in every input channel of associate NOC router consists of a SRAM-based FIFO memory of sure depth. throughout traditional operation, information arrive through a data_in line of the buffer and are later hold on in several locations of the FIFO memory. for the asking by the neighboring router, the information flits hold on square measure passed on to the output port through the data_out line. Fig. 3 shows the FIFO memory with data_in and data_out line. To perform the transparent SOA-MATS++ check on the FIFO buffer, we tend to supplementary a check circuit, few multiplexers, logic gates and registers to the prevailing hardware, as shown in Fig. 3. The scan and write operations on the FIFO buffer are controlled by the scan change and write change lines, severally. The multiplexers mu6 and mu7 choose the scan and write change throughout the traditional and check method. throughout traditional operation once the test_ctrl is declared low, the internal write and read ensable change lines, wen_int and ren_int, synchronous with the clock, offer the write and therefore the scan change, severally. The scan operations square measure synchronous with the positive edges, whereas the write_clk is obtained by inverting the clock. In check mode (test_ctrl high), the check scan and write addresses square measure generated by check address generators enforced exploitation grey code counters just like the traditional address generation. Muxes m4 and m5 are to choose between traditional addresses and check addresses. The registers are considered to be D-flipflops, as they are used to store the values before and after the circuit process. The addition of registers in between the circuits forms a pipeline. The process circuit of a given stage is connected to the input latch of consecutive stage. A clock signal is connected to every input latch. At every clock pulse, each stage transfers its intermediate result to the input latch of consecutive stage. once the pipeline has been stuffed, it generates AN output on every clock cycle. In alternative words, once the pipeline is loaded, it'll generate output solely as quick as its slowest stage.

Consider the case once the FIFO buffer is in traditional mode with flits being transferred from the memory to the data_out line. once many traditional cycles, the test_ctrl is declared high, switch the buffer to check mode. As long because the buffer is in check mode, no external information is allowed to be written to the buffer, or in alternative words, the buffer is fast for the trial period. As a result, the input file line for the FIFO memory is switched from the external data_in line to check_data line out there from the test circuit. At the switch instant, the flit that was within the method of

being transferred to the data_out line is at the same time scan into the check Circuit. However, a 1 clock cycle delay is formed for the flit to maneuver to the data_out line.

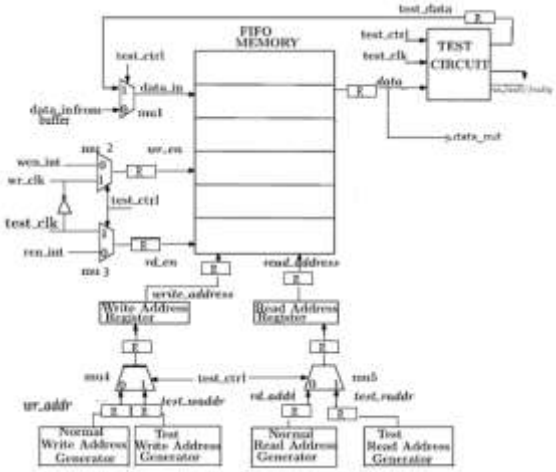

Figure 3. Architecture of the test circuit with the Pipelining method

This delay ensures that the flit isn't lost throughout the switch instant and is correctly received by the router, that requests for it. The flit, that is scan within the check circuit, is hold on during a temporary register temp and therefore the test method begins with this flit.

VI. RESULTS

Simulation results:

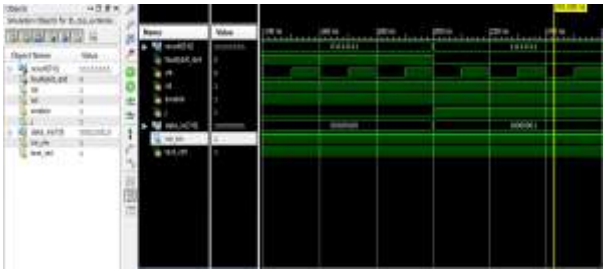

Figure 4. Simulation results for fault detection bit = 0

Figure 5. Simulation results for fault detection bit = 1

Synthesis Results

The proposed architecture for the test circuit has been described in Verilog HDL. The difference in the delay before and after the inclusion of pipelining was found to be 2.877ns less than the original because of adding pipelining. The total area estimate in the test circuit after synthesis has been estimated to be 246 out of 8672 slices. Thus, a significant amount of 55% delay has been reduced.

VII. CONCLUSION

We have implemented the transparent SOA-MATS++ test generation algorithm which will find run-time permanent faults developed in SRAM-based FIFO buffers for eight bits. we've introduced the pipelining technique that has reduced the delay. The simulation results show the reduction in delay and area as 55% and 39% respectively. The transparent test is used to perform periodic check of FIFO memory present among the routers of the NOC. Periodic checking of buffers prevents accumulation of faults and conjointly permits test of every location of the buffer. As a part of future work we can apply the whole test setup to an asynchronous FIFO with multi wide multi depth FIFO for multiple error detection and correction.

REFERENCES

[1] Bibhas Ghoshal, KanchanManna, Santanu Chattopadhyay, and Indranil Sengupta, “In-Field Test for Permanent Faults in FIFO Buffers of NoC Routers” in IEEE Trans. Comput., vol. 24, no. 1,Jan. 2016.

[2] W. J. Dally and B. Towles, “Route packets, not wires: On-chip inter- connection networks,” in Proc. 38th Annu. Design Autom. Conf., 2001, pp. 684–689.

[3] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, “Threshold-based mechanisms to discriminate transient from intermittent faults,” IEEE Trans. Comput., vol. 49, no. 3, pp. 230–245, Mar. 2000.

[4] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, “Methods for fault tolerance in networks-on-chip,” ACM Comput. Surv., vol. 46, no. 1, pp. 1–38, Jul. 2013, Art. ID 8.

[5] S. Ghosh and K. Roy, “Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era,” Proc. IEEE, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.

[6] S. Borri, M. Hage-Hassan, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, “Analysis of dynamic faults in embedded-SRAMs: Implications for memory test,” J. Electron. Test., vol. 21, no. 2, pp. 169–179, Apr. 2005.

[7] M. Bushnell and V. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits (Frontiers in Electronic Testing). New York, NY, USA: Springer-Verlag, 2000.

[8] D. Xiang and Y. Zhang, “Cost-effective power-aware core testing in NoCs based on a new unicast-based multicast scheme,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 30, no. 1, pp. 135–147, Jan. 2011.

[9] K. Petersen and J. Oberg, “Toward a scalable test methodology for 2D-mesh network-on-chips,” in Proc. Design, Autom., Test Eur. Conf. Exhibit., Apr. 2007, pp. 1–6.

[10] C. Grecu, P. Pande, B. Wang, A. Ivanov, and R. Saleh,

"Methodologies and algorithms for testing switch-based NoC interconnects," in Proc. 20th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., Oct. 2005, pp. 238–246.

[11] M. R. Kakoee, V. Bertacco, and L. Benini, "A distributed and topology- agnostic approach for on-line NoC testing," in Proc. 5th ACM/IEEE Int. Symp. Netw. Chip, May 2011, pp. 113–120.

[12] A. J. van de Goor and Y. Zorian, "Functional tests for arbitration SRAM-type FIFOs," in Proc. 1st Asian Test Symp. (ATS), Nov. 1992, pp. 96–101.

[13] M. Nicolaidis, "Theory of transparent BIST for RAMs," IEEE Trans. Comput., vol. 45, no. 10, pp. 1141–1156, Oct. 1996.

[14] D. Xiang and Y. Zhang, "Cost-effective power-aware core testing in NoCs based on a new unicast-based multicast scheme," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 30, no. 1, pp. 135–147, Jan. 2011.

AUTHOR'S BIOGRAPHIES

K Sharon Kamal Pursuing M.E in Embedded Systems and VLSI from Maturi Venkata Subba Rao Engineering College Hyderabad. B.Tech in Electronics and Communication Engineering from G. Narayanamma Institute of Technology and Science, Hyderabad.

V Rajitha Rani Asst. Prof, Maturi Venkata Subba Rao Engineering College Hyderabad. M.Tech, Gokaraju Rangaraju Institute Of Engineering And Technology. Purusing Ph.D from JNTU Hyderabad and the area of research is Analog and Digital VLSI and Fault Tolarance Systems.