International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

Fault detection of NOC Routers in FIFO memory

Ch. Kalpana & M.S. Shyam
[1]M.Tech (VLSI&ES), St. Mary's College of Engineering and Technology, Hyderabad
[2]Assistant Professor, St. Mary's College of Engineering and Technology, Hyderabad

Abstract-*We have proposed an online transparent test technique for first-input first-output (FIFO) buffers and routing logic present within the routers of the NoC infrastructure. Our contributions are as follows. A transparent SOA-MATS++ test generation algorithm has proposed targeting in-field permanent faults developed in SRAM based FIFO memories and it has been utilized to perform online and periodic test of FIFO memory present within the routers of the NoC. In addition, we have also proposed an online test technique for the routing logic that is performed simultaneously with the test of buffers. The proposal involves two ways of utilizing the unused portion of the header flits of the incoming data packets in transporting the test patterns. First, deterministic test patterns for the routing logic generated by Tetramax are placed in the unused fields of the header flit and are transported during the normal cycle. Second, the pseudorandom patterns in the synthetic data traffic used during normal operation and arriving at the routing logic are considered as test patterns. Fault coverage is estimated for either of the two proposals.*

I. INTRODUCTION

FIFO (First In First Out) memories have acquired a larger interest in VLSI design in telecom applications. Such macrocells provide an efficient way to interface two asynchronous systems, buffering data between subsystems operating at different data rates, thus increasing the flexibility in data transmission. In fact, data in FIFO memories are read following the same order in which they have been written, without the need of providing the address of the data. Different approaches have been considered in literature to design FIFO memory cells. In this paper we define an architecture based on a static RAM cell, available in the Italtel library of components. We aim at defining a parametric architecture whose size can be modified with slight modifications to the basic structure. In order to obtain such a result, the control logic has been designed in a modular way, thus defining a flexible structure, whose basic components are described in VHDL. In particular, the control logic is constituted by two subsystems with two independent clock signals, managing the read and write operations, respectively. Performance and area constraints have driven the design optimizations. Transient and permanent faults are two different types of faults that can occur in on-chip networks [1]. Transient faults are temporary and unpredictable. They are often difficult to be detected and corrected. Permanent faults are caused by physical damages such as manufacturing defects and device wear-out. These faults should be recovered or tolerated in a way that the network continues functioning. Routing techniques provide some degrees of fault tolerance in NoCs. They can be categorized into deterministic and adaptive approaches [1]. A deterministic routing algorithm uses a fixed path for each pair of nodes resulting in increased packet latency especially in congested networks. In adaptive routing algorithms, packets are not restricted to a single path when traversing from a source to a destination router. So, they can decrease the probability of routing packets through congested areas and thus improve the performance. In minimal adaptive routing algorithms, the shortest paths are used for transmitting messages between routers. In contrast, performance can severely deteriorate in non-minimal methods due to taking longer paths by packets. Moreover, non-minimal methods are usually more complex with a larger number of virtual channels. Deadlock-free non-minimal routing schemes are the most common approaches to tolerate faults in the network [3]. They are used in order to reroute packets around faults [4]. Some fault-tolerant algorithms are proposed to support special cases of faults, such as one-faulty routers, convex or concave regions. These algorithms either disable the healthy components or require a large number of virtual channels to avoid deadlock.

Dynamic packet fragmentation in networks-on-chip (NoCs) was first introduced in [5] for enhanced virtual channel (VC) utilization and deadlock avoidance in multicast routing. The proposed router exploits fragmentation as a response to error detection and renews state information by reallocating a new VC. Upon fragmentation, trailing flits can then avoid a faulty VC which has corrupted states. The proposed router is evaluated in erroneous environments by fault injection mechanisms. Statistical fault injection (SFI) [6] is applied at each bit of a network link with independent probability. At the various fault rate levels, the proposed router is observed to perform well, gracefully degrading while it shows resilience to link errors without missing any flits. In the intra-router error analysis, the router is observed to have 97% error coverage for datapath elements. Several approaches have been proposed in literature in the context of topological mapping in NoCs [3]. Mapping algorithms are mostly focused on 2D mesh topology which is the most popular topology in NoC design due to its layout efficiency, good electrical properties and simplicity in addressing on-chip resources. Another concern in NoC implementation is selecting an efficient routing strategy while providing freedom from deadlock. The routing algorithm determines the path that each packet follows between a source-destination pair. In the future chip generations, faults will appear with increasing probability due to the susceptibility of shrinking feature sizes to process variability, age-related degradation, crosstalk, and singleevent upsets. A memory unit is a collection of cells that are capable of storing a large quantity of binary information associated with circuits needed to transfer information into and out of a device. The architecture of memory is done in such a way that information can be selectively retrieved from any of its internal locations. Memory tests are used to confirm that each location in a memory device is working.

II. RELATED WORK

A. The Architecture of the FIFO Memory

Different classes of FIFO memories have been presented and realized in literature. The shifting-type FIFO is based on a shift register of n cells in which data move from the input writing port towards the read output port. When a write operation is performed, from the input write port the datum is inserted into the first cell of the register and shifted of one position, at every clock cycle, until the last free cell is reached. A read operation moves the content of the last cell of the register to the output port, while moving simultaneously the remaining data of one position towards the output port. The structure of such a memory is quite simple, but the main drawback is that each datum written must traverse all n cells of the memory before being available for a read operation, with a minimum delay of n clock cycles. The arbitration RAM-type FIFO is based on a static RAM to store data and on the presence of a write address register and a read address register. Each one of such registers is constituted by an m bit counter (m=log2 n). A controller manages the read and write operations with the FIFO mode and decides which address register must be provided to the RAM port. This realization, although simple and inexpensive, does not allow simultaneous read and write operations. It is therefore not suitable for applications for which an high frequency of operation is required. The dual-port RAM-type FIFO is based on a dual port RAM cell to allow simultaneous read and write operations. Two different independent address busses are provided, to manage the read and write operation, respectively. The main difference with the arbitration RAM-type FIFO is the duplication of the address bus, thus allowing better performance in terms of maximum operation frequency. Two implementations can be identified based on the data access mode: ring-address RAM-type FIFO and counter-address RAM-type FIFO. The first solution is characterized by the use of two n bit shift-registers in which each cell is uniquely associated in an ordered way with a memory cell. Such an implementation requires a RAM macrocell without address decoder, since the two registers perform both the function of address generation and decoding. Such addressing mode allows the definition of very fast FIFO memories, but the area occupied increases linearly with the size of the memory, due to the increase of the size of the registers.

The RAM FIFO has been designed with two controllers independently synchronized, which separately manage the read and write operations. The evaluation of the two conditions empty-fifo and full-fifo considering the memory as a circular buffer and observing that the two conditions can be recognized by checking the read and write addresses as well as the number of times the buffer has been traversed. In fact, the empty-fifo condition is verified when the read and write addresses coincide and the buffer has been covered the same number of times. Conversely, The full-fifo condition is verified when the write address is equal to the read address, but it has performed one more cycle with respect to the read address. Obviously, the read address can never be greater than the write address and the addresses can differ for at most one cycle. Therefore the actual number of cycles performed by the read and write addresses is not relevant, the information necessary to evaluate the two conditions concerns the difference between the two cycling of addresses.

B. The Router Architecture of MiCoF

In NoCs, faults may occur in cores (such as processing elements and memory modules), links or routers. When a core is faulty, the connected router and links can continue functioning. When a link is faulty, the approaches similar to BFT-NoC [7] retain the connectivity by a dynamic sharing of surviving channels. Thereby, cores and routers perform functioning normally. The most severe case causes by a faulty router. In this case, not only the connected core cannot send or receive packets, but also the packets from the other cores cannot be transmitted through this router. The most common solution is to reroute packets around the faulty router or faulty region. This may imply a non-minimal routing algorithm based on turn models. However, turn models are very limited to tolerate multiple faulty routers. In sum, when a router is faulty, the corresponding core and links are also tagged as faulty and become out of use. The core cannot start working until the fault is recovered.

C. Fault-Tolerant Flow Control

Now that a preliminary router infrastructure has been established for fault detection and packet reconstruction, a new flow control scheme for fault handling is presented. A conventional VC router, which adopts credit-based flow control, returns a credit when a flit is forwarded to the next router and the corresponding buffer entry is free. The upstream router which receives the credit then sends a next flit. This enhances buffer recycling by signaling to the upstream router as soon as the buffer entry is empty. However, if an error is detected at the downstream router and the buffer entry which held the faulty flit in the current router is replaced by the next flit, the faulty flit cannot be recovered unless retransmission is requested from the source node. So a new flow control scheme, which keeps the sent flit until the safety of the flit is ensured at the downstream router, is necessary.

A fault-tolerant mesh-based NoC architecture with the ability of recovering from single permanent failure is presented in [8]. This method adds a redundant link between each core and one of its neighboring routers, resulting in significant improvement in reliability while has little impact on performance. In this architecture, only one spare router should be selected among all possible alternative ones. This has an influential effect on overall performance in terms of the average response time and reliability of the system. Regarding to this work, in a new fast and optimum algorithm based on performance measurement and extra communication cost is proposed to find the best configuration that also results in a more reliable system.

III. PREREQUISITES OF APPLICATION-SPECIFIC NOC DESIGN

The proposed method comprises two main modules: The routing algorithm determines the path that each packet follows between a source-destination pair. Routing algorithms noticeably affect the cost and performance of NoC parameters i.e. area, power consumption and average message latency. Due to determined sources and destinations in application-specific NoC, minimum-distance routing algorithms are mostly considered in this area which is computed off-line and admissible paths stored into the routing tables. Fault-tolerant routing algorithms should be able to find a path from source to destination in presence of the faults in NoC with a certain degree of tolerance. Many algorithms in this area have been proposed which follow their own optimization aims. To name just a few, the authors propose a novel low-overhead neighbor aware, turn model based fault-tolerant routing scheme (NARCO) for NoCs which combines threshold-based replication in network interfaces, a parameterizable regionbased neighbor awareness in routers, and the odd–even and inverted odd–even turn models.

IV. IMPLEMENTED ALGORITHM

In proposed system to increase the size of the FIFO buffer to 32bit range. 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. 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.

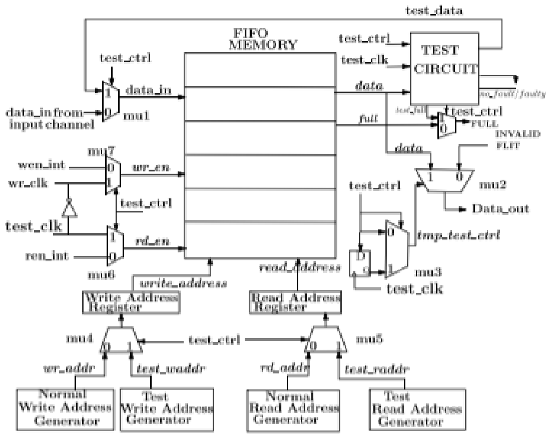
```
Algorithm 1 Transparent SOA-MATS++ Test Algorithm
  Require: N = number of rows of the FIFO memory
   1: $i \leftarrow 0$;                       /* memory address pointer */
   2: while ($i \leq N - 1$) do
   3:   $j \leftarrow 0$;                       /* test cycle counter */
   4:   while ($j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

Fault Coverage of the Proposed Algorithm

The FIFO buffer present in each input channel of an NoC router consists of a SRAM-based FIFO memory of certain depth. During normal operation, data flits arrive through a data_in line of the buffer and are subsequently stored in different locations of the FIFO memory. On request by the neighboring router, the data flits stored are passed on to the output port through the data_out line. Fig. 5(a) shows the FIFO memory with data_in and data_out line. To perform the transparent SOA-MATS++ test on the FIFO buffer, we added a test circuit, few multiplexers and logic gates to the existing hardware. The read and write operations on the FIFO buffer are controlled by the read enable and write enable lines, respectively. The multiplexers mu6 and mu7 select the read and write enable during the normal and test process. During normal operation when the test_ctrl is asserted low, the internal write and read enable lines, wen_int and ren_int, synchronized with the router clock, provide the write and the read enable, respectively.

V. RESULTS

In order to compare the average response time and hardware cost of the reliable architecture and traditional mesh, they have been designed in VHDL and synthesized using Xilinx ISE (on FPGA – Xilinx VitexE).



Hardware implementation of the test process for the FIFO buffers

For our experiments, packets are generated according to a uniform distribution with the rates that extracted from VOPD core graph. As can be seen in Fig. 8, it takes 137,265 cycles to reach all packets to the destination routers in the traditional mesh. As mentioned before, the proposed architecture is able to tolerate one router failure and guarantees the 100 percent packet delivery.

VI. CONCLUSION

In the presented approach, all packets are routed through the shortest paths, maintaining the performance of NoC in the presence of faults. To be able to route packets through the shortest paths, the router architecture is slightly modified. The purpose of this modification is to maintain the connectivity among the surviving routers.

It is realized that the transistor include minimization CMOS doors may enhance the execution, control dispersal, and territory of computerized ICs. In a general perspective, the proposed technique produces efficient switch courses of action very valuable to be investigated by various IC advances in view of switch hypothesis. This paper presented a fault-tolerant NoC router, recovering faulty flits through a link-level retransmission. We demonstrated how a faulty flit is fragmented and retransmitted in a fault-tolerant flow-control scheme, and evaluated the design in various workload environments.

REFERENCES

[1] 'A parametric design of a built-in self test FIFO embedded memory,' Barbagallo S. Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., pp. 221–229,1996.

[2] 'Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip,' Ebrahimi M. Daneshtalab M. Plosila J. and Tenhunen H. Proc. 7th IEEE/ACM Int. Symp. Netw. Chip (NoCS), pp. 1–8, 2013.

[3] 'In-Field Test for Permanent Faults in FIFO Buffers of NoC Routers' Ghoshal B. Sengupta I. Manna K. and Chattopadhyay S. IEEE Trans. VLSI Syst., 2015.

[4] 'Fault-Tolerant Application-Specific Network-on-Chip', Koupaei F.K. Khademzadeh A. and Janidarmian M. proc. WCECS 2011, pp.19-21, 2011.

[5] 'Fault-Tolerant Flow Control in On-Chip Networks' Kang Y.H. Kwon T.J. Draper J. Fourth ACM/IEEE Int. Symp. on Networks-on-Chip (NoCs), 2010.

[6] 'Performance Analysis Of BIST Algorithm For Rams,' Kusuma L. International Journal For Technological Research In Engineering Volume 2, No.7, ISSN (Online): 2347 – 4718, 2015.

[7] D. Fick. A. DeOrio, G. Chen, v. Bertacco, D. Sylverster, D.Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs", in Proc. of DATE, pp. 21-26, 2009.

[8] Ch. Feng, Zh. L, A. Jantsch, J. Li, M. Zhang, "A Reconfigurable Faulttolerant Deflection Routing Algorithm Based on Reinforcement Learning for Network-on-Chip", in Proc. of NoCArc, 2010.