An Effective Approach for the Detection of Soft Errors in FIFO Buffers Using Parity Registers

P. Beautlin Push, Dr. T. Latha

Abstract— The data word that is written in first also comes out first when the memory is read in a First In First Out buffer. First In First Out buffer gets involved in many system operations and they should be protected properly for ensuring reliable operation. This work describes a column parity based fault detection technique for FIFO buffers in single clock domain FIFO. Column parity based fault detection is a low cost technique that involves updation of a parity register to detect faults. This technique is based on the fact that the faults can be detected only if the FIFO gets empty. The requirement to wait for the FIFO to get empty increases the detection latency. The proposed work is based on detecting the fault immediately after the data is read rather than waiting for the FIFO to get empty thus reducing the detection latency and improving the detection rate.

Index Terms— Buffer, Column parity, FIFO, Fault Detection, Latency, Soft Error.

1 Introduction

FIFOs are used in designs to safely pass multi-bit data words from one clock domain to another or to control the flow of data between source and destination side sitting in the same clock domain. First In First Out memories are widely used as building block to buffer data between subsystems operating at different rates [1]. FIFOs are mainly used for interfacing asynchronous designs where the rate of data generated and data consumed is not same. In communications and networking, bridges, routers and switches use FIFO memory to hold data packets that are in route to a given destination. Current technology nodes are more prone to failures due to the complexity involved in them. Static and dynamic variations results in unreliable operation even making the system to collapse. Moreover aging mechanisms such as NBTI, electro migration, time dependent dielectric breakdown degrade devices and cause faults in the field [2].

Errors in memories can be broadly classified as either hard or soft errors. Hard faults known as permanent faults can occur either at manufacture time or in the field [3]. Soft faults known as transient faults arise due to energetic particle strikes, process variations, temperature variations etc. For example SRAM cells are more vulnerable to voltage changes since they are denser and their stability depends on the threshold voltages of transistors. It has been reported that in 12nm one every few thousands SRAM cells will be faulty due to random dopant fluctuation and aging [4]. Faults in FIFO include stuck-at-fault, data retention fault, coupling faults etc and they manifest their existence as bit flips.

Information redundancy is a viable solution for the protection of FIFOs but it comes at the expense of state overhead. The TMR technique [5] is not cost efficient, consumes more area and uses more power. This is a technique of introducing an auxiliary register and writing both buffer and auxiliary register to check if they agree [6] consume more power. Dynamic implementation and verification architecture incurs performance penalty for each error it detects. In [3] the authors used standard horizontal protection schemes to detect multibit errors in cache memories but there exists area, power and performance overheads. In [7] the authors used HVD technique to detect and correct soft errors in semiconductor memories where the increase in word length increases the analysis time. In [8] the authors used matrix code to protect SRAM based memories against multiple bit upsets which can detect multiple faults more than hamming code but less than Reed-Muller code.

In column parity technique, the detection of faults occurred in FIFO buffers can only be accomplished when the FIFO gets empty. This increases the detection latency because it has to wait for the FIFO to get empty. Column parity technique has the advantage of using low power, less area and decreased critical path overhead. The proposed work aims at detecting faults in FIFO buffer immediately after each data is read from FIFO buffer. Hence it is not required to wait for the FIFO to become empty to detect faults. This work has the advantage of detecting the number of bit flips in each data in addition to the location of erroneous data.

2 FIFO Buffers

A FIFO buffer is a special type of buffer. In hardware it is either an array of flip-flops or read/write memory that store data given from one clock domain and on request supplies the same data to other clock domain following the first in first out logic. In First In First Out model, the data are stored in stack basis. The data given as input is inserted in the location pointed by push address pointer. The oldest data is processed first and leaves the buffer first when pointed by the pop address pointer.

The basic elements of FIFO buffer include storage element, read pointer, write pointer, empty and full flags. A buffer is a region of physical memory storage used to temporarily store data while it is being moved from one place to another. Initially read and write pointer of FIFO points to the same location. After each push operation, the write pointer is incremented to point to the next location to write. After each pop operation, the read pointer is incremented to point to the next address space for reading. FIFO buffer can be written when FIFO is not full.

Figure 1 shows the structure of FIFO.

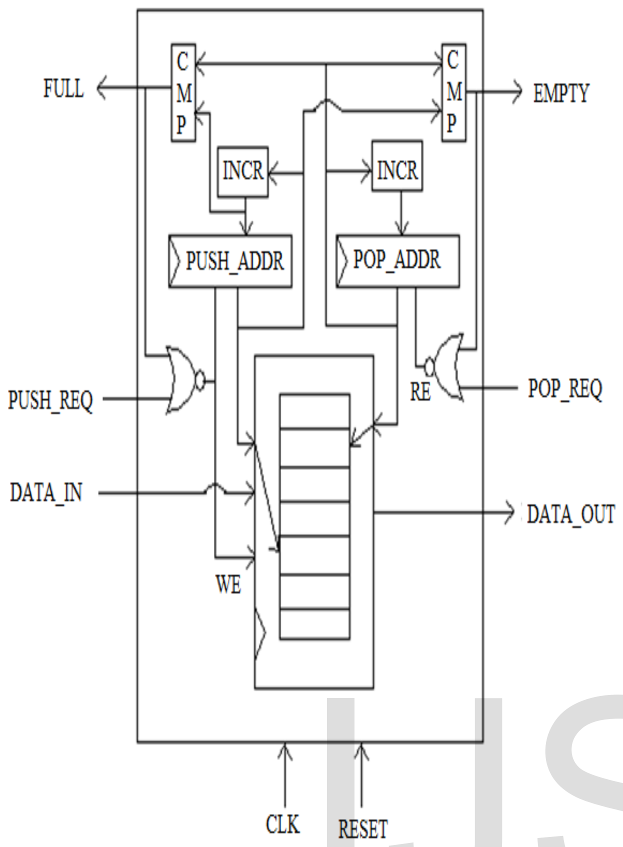

Fig. 1. Parity based FIFO architecture

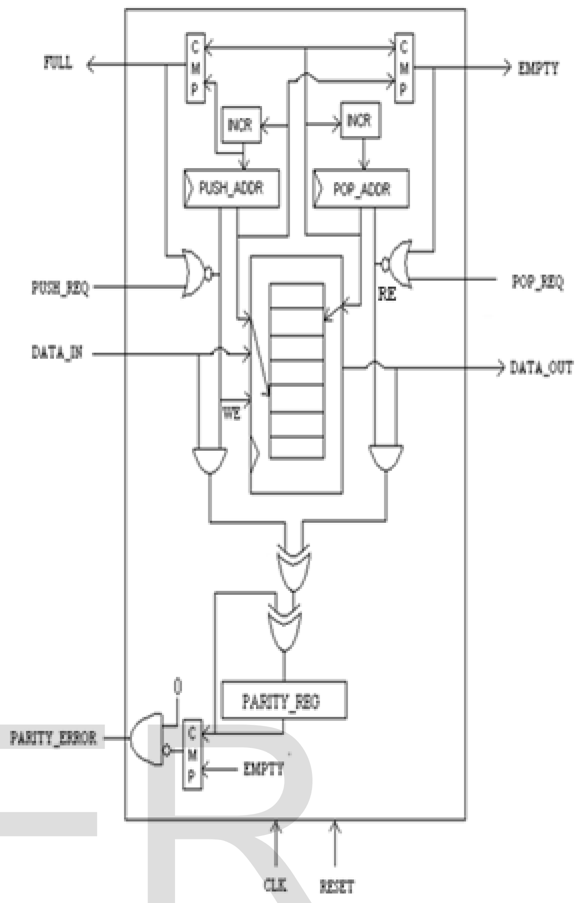

Fig. 2. Column parity based protection mechanism

Full and empty are the signals indicating the status of FIFO. When push address equals pop address, it indicates that FIFO is empty. When push address +1 = pop address, it indicates that FIFO is full. Both Push_Req and full signals are used to enable the WE signal. In addition to this, when the clock signal is asserted the data given as input to FIFO buffer is written onto the memory location pointed by push address. During pop operation, the data item residing in the address pointed by pop pointer is retrieved as data _out (i.e.) the data that goes into the FIFO first comes out of FIFO first thus verifying FIFO operation.

3 Column Paruty Based Fault Detection

Column parity technique uses a single parity register that stores the global parity in column basis. The parity register stores the parity of each column. Normally each parity would need a read before every write, in order to be updated. The parity register has to be updated after each push or pop. After each and every write, the parity register is updated by performing an XOR operation of the value to be written with the parity register value. Figure 2 shows the FIFO buffer augmented with the fault detection mechanism.

At every read we perform an XOR of the read value with the parity register value and update the global parity register. Fault detection in the FIFO buffer can be made only if FIFO gets empty. Whenever the FIFO is empty, the parity register should be zero to indicate that there is no fault in any of the array bits. A non zero value in the parity register indicates the presence of faults in some array bits.

The parity update takes place when a read or write or both happens. The two XOR gates accounts for the case when read and write happens in parallel. In the way the parity is updated faults accumulate in the parity register and when the FIFO gets empty they manifest their existence [4].

4 Proposed Work

Column parity technique has greater detection latency. In addition, it doesnot give detail about the number of bit flips in a particular data. The proposed work is concentrated on minimizing the latency of fault detection by having the parity check after every read operation.

This work has the advantage of detecting the number of bit flips in each data in addition to the location of erroneous data.

Figure 3 below shows the proposed architecture for error detection.

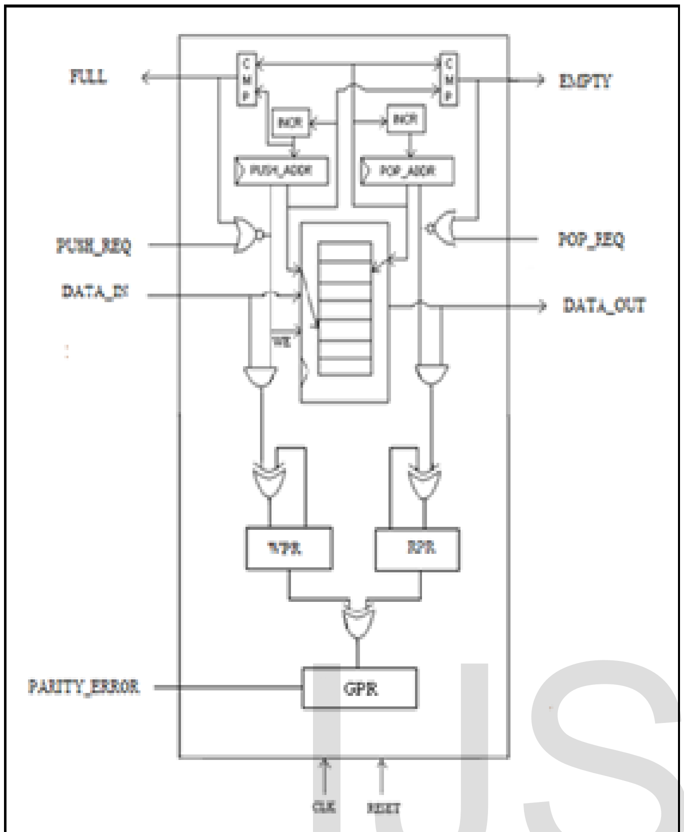

Fig. 3. Proposed architecture for error detection

The error detection can be achieved by using two additional registers namely write parity register (WPR) and read parity register (RPR) in additional to a global parity register (GPR). The process of fault detection is done as follows: After each data is pushed into FIFO, the write parity register is updated by performing XOR operation of the data value to be written with the value in the write parity register. Similarly when it is read, the read parity register is updated by performing XOR operation of the read out value with the value in read parity register. Then the fault can be detected by updating the global parity register using the following equation (1).

$$GPR = WPR \oplus RPR$$

A nonzero value in the GPR is an indication of fault.

5 Results and Discussion

The results are simulated in Active HDL and are discussed below:

5.1 FIFO Operation

Figure 4 shows the simulation result verifying FIFO operation. As soon as the clock is asserted and push_req is low as well as the FIFO is empty the data given as input is written into FIFO in the location pointed by push pointer.

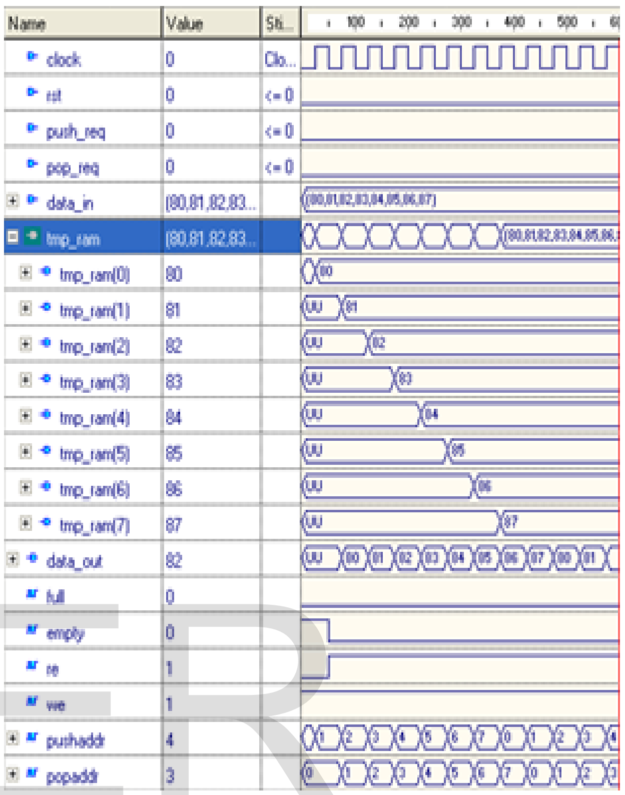

Fig. 4. FIFO operation

After performing a push operation (write), the write pointer is incremented by one to point to the next location to write. During pop operation, the data that is written into FIFO buffer first leaves the FIFO first thus verifying FIFO operation. After each pop operation (pop), the read pointer is incremented to point to the next address space for reading.legible.

5.2 Column Parity Based Fault Detection In FIFO

The parity register is updated each time when a read or write or both happen in parallel. After all the data are read, the parity register shows zero indicating that no faults have occurred in FIFO. A nonzero value in the parity register is an indication of fault. Figure 5 shows the simulation results for the case if there is no fault in any of the array bits.

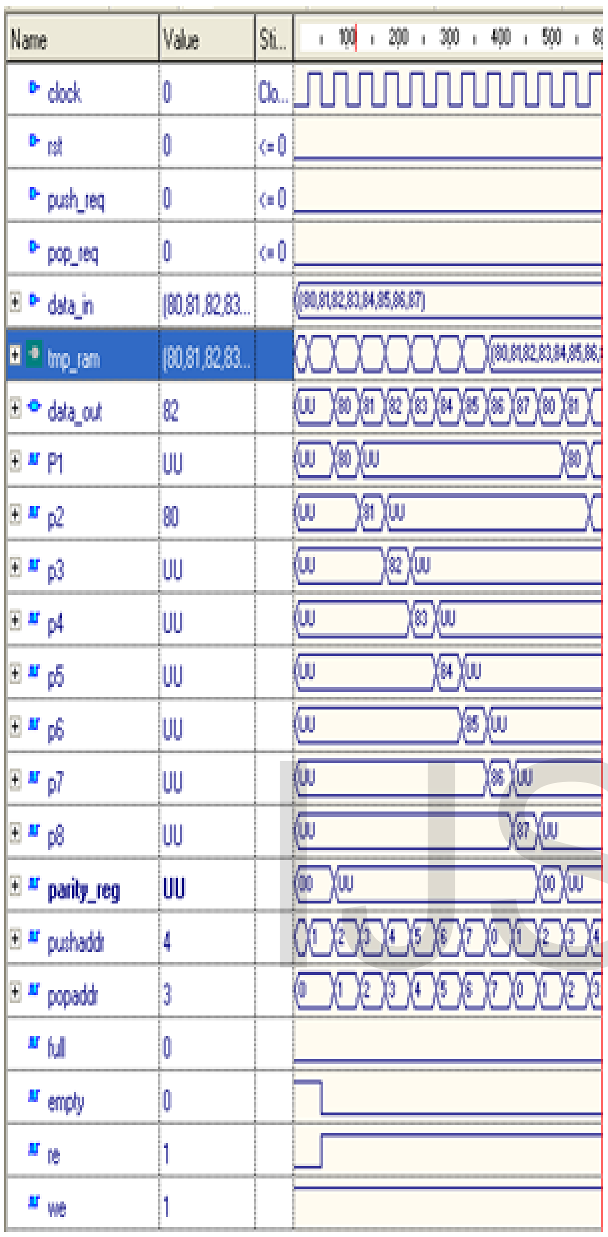

Fig. 5. Simulation result for column parity based fault detection

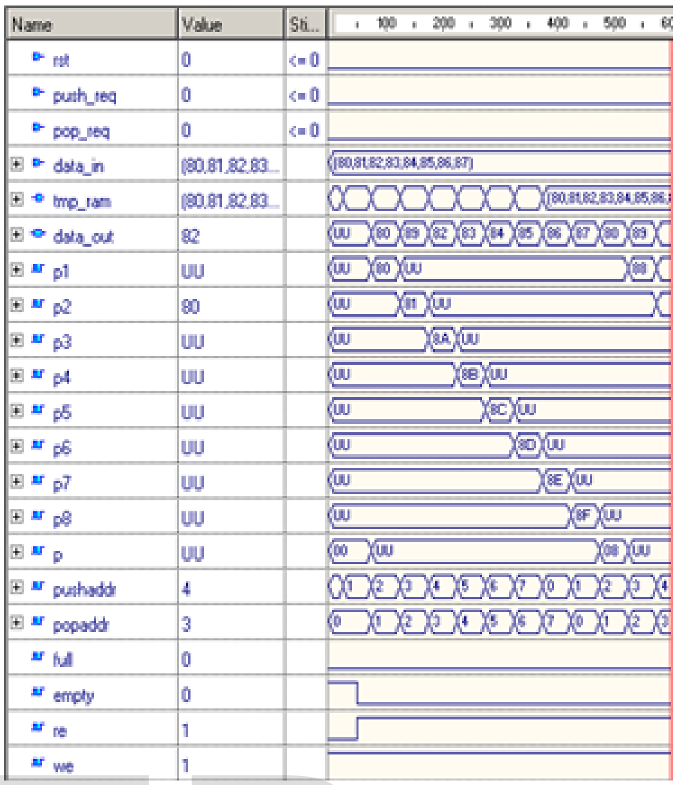

Fig. 6. Simulation result for column parity based fault detection

Figure 6 accounts for the case when errors encountered in some array bits of FIFO. Note that the second data 81 is retrieved as 89 due to the occurrence of soft fault in FIFO.The parity register is indicated by the variable p in Figure 6. After all the data is read, the parity register is checked for error detection. The occurrence of fault is indicated by a nonzero value in the parity register.

6 Conclusion

Fault detection in FIFO buffers using column parity technique has the limitation of greater detection latency since the detection can only be done when the FIFO gets empty and is not able to determine the location of fault. The proposed work can minimize the latency as it does not need to wait for the FIFO to get empty and also identify the location of fault. Results are obtained for performing FIFO operations and column parity based fault detection. This work is to be proceeded to implement the proposed work.

References

[1] Van de Goor, Yervant Zorian "Fault models and tests specific for FIFO functionality," IEEE,1993.
[2] J.Srinivasan, Sarita V. Adve, Pradip Bose "Lifetime reliability: toward an architectural solution," IEEE Micro 25, 2005, pp. 70-80.
[3] Jangwoo Kim, Nikos Hardavellas, ken Mai, Babak Falsafi, James C. Hoe "Multi-bit error tolerant caches using two-dimensional error coding," IEEE, 2007.
[4] Isidoros Sideris, Kiamal Pekmestzi "A column parity based fault detection mechanism for FIFO buffers," Integration, the VLSI journal Vol 46, 2013, pp. 265-279.
[5] R.Henstschke, F.Marques, F.Lima, L.Carro and R.Reis "Analyzing area and performance penality of protecting

different digital modules with hamming code and triple modular redundancy," proc.15th symposium on integrated circuits and system design, 2002.

[6] F.A.Bowler, P.G. Shealy and Ozev S, D.Jsorin "Tolerating hard faults in microprocessor array structures," Proceedings of International Conference on Dependable Systems and Networks, 2004, pp. 51-60.

[7] Shalini Sharma and Vijayakumar "An HVD based error detection and correction of soft errors in semiconductor memories used for space applications," International conference on Devices, Circuits and Systems, 2012, pp 563-567.

[8] Costas Argyrides, Hamid R. Zarandi, Dhiraj K.Pradhan "Multiple upsets tolerance in SRAM memory," IEEE, 2007.