INTEGRATION, the VLSI journal 46 (2013) 265–279

Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A column parity based fault detection mechanism for FIFO buffers

Isidoros Sideris *, Kiamal Pekmestzi

School of Electrical and Computer Engineering, National Technical University of Athens, 9 Heroon Polytechneiou, Athens 15780, Greece

ARTICLE INFO

Article history:
Received 27 September 2011
Received in revised form
18 January 2012
Accepted 28 March 2012
Available online 10 April 2012

Keywords:
FIFO
Reliability
Fault detection
Column parity
Dynamic verification

ABSTRACT

This paper presents a low cost fault detection mechanism for FIFO buffers. The scheme is based on column parity maintenance in a single register, which is updated by monitoring the values written to and read from the FIFO memory array. A non-zero column parity when the FIFO is empty, constitutes an indication of fault, and this property is exploited for fault detection. The technique has gains in area, power and critical path delay, at the expense of (1) greater detection latency, due to the need for the FIFO to become empty in order to assert a violation and (2) worse Silent Data Corruption (SDC) rate.

1. Introduction

First In First Out (FIFO) memories are used for buffering and flow control and are indispensable parts of almost every system. They are widely used in on chip communication fabric, high speed communication protocol implementations, image/video accelerators [1], multiple channel DMA controllers and coprocessor interfaces (e.g. Xilinx FSL [2]). What is more, they are extensively used between different clock domains for synchronization. Modern multicore processors require different clock domains controlled by Dynamic Voltage and Frequency Scaling (DVFS) mechanisms to meet the power requirements (maximum Thermal Design Power—TDP) and communication between them is ensured using FIFOs [3]. Globally Asynchronous Locally Synchronous (GALS) systems [4] extensively use FIFOs. The 48 core Intel IA-32 chip consists of a lot of FIFOs for communication [5].

Since FIFOs intervene in many system operations, they should be protected properly to ensure reliable operation. This becomes more imperative in current and future technology nodes, in which system failures are becoming more and more dominant. Static and dynamic variations [6,7] result in unreliable operation. Moreover, aging mechanisms [8] such as NBTI [9], electromigration, time dependent dielectric breakdown degrade devices and wires during system's lifetime and cause faults in the field. Furthermore, soft errors [10] and other types of transient faults affect reliability significantly. Thus, dependability in vital system operations is of utmost importance.

Operation in lower voltage, which is more than required to meet the power constraints, exacerbates devices reliability and process variation related problems appear more intense. SRAM cells seem to be much more vulnerable than logic and flip flops [11], since they are more dense, and their stability greatly depends on the asymmetry of the threshold voltages of their transistors. In [11] has been reported that in 12 nm one every few thousands SRAM cells will be faulty (due to Random Dopant Fluctuations and aging). In [12] the dependence of SRAM cell probability of failure (p_{fail}) on voltage is shown for 32 nm technology nodes.

Results show that even in small memory arrays, faults due to process variation/aging will occur. On the other hand, soft error resiliency should be ensured. Results also show that combinational logic is less vulnerable to process variation/aging induced faults (unless the time constraints are very tight). Thus, in this paper we more focus on array related faults.

All these effects result from scaling. Thus it would be wise to spend as less resources as possible for protection, otherwise we could result in canceling the effects of scaling. Information redundancy is a viable solution for protection in FIFOs, but it comes at the expense of state overhead and combinational circuit path for encoding/decoding. Even byte parity, which is the simplest fault detection technique, imposes a 12.5% state overhead, which is not negligible. What is more, it imposes critical path (8–9 input XORs in encoding/decoding) and energy overheads.

In this paper we propose a low cost fault detection technique for FIFO buffers. It is based on the update of a global parity register, which stores global parity in a column basis, requiring only a flip flop and two XOR gates per column. The fault detection

* Corresponding author. Tel.: +30 2107723653.
E-mail addresses: isidoros@microlab.ntua.gr (I. Sideris), pekmes@microlab.ntua.gr (K. Pekmestzi).

0167-9260/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2012.03.004

is based on the fact that when the FIFO becomes empty, the accumulated parity of the data read will be the same with that of the written and this register should be zero. A non-zero value is an indication of fault, and this property is exploited for fault detection. All faults encountered between two empty states accumulate and are effectively indicated in the column parity register.

The requirement to wait for the FIFO to become empty, however, increases the detection latency. Thus, we trade off detection latency for area, power and critical path overhead reduction.

Due to the unbounded detection latency of the mechanism, it can be more easily applied in systems with backward error recovery (BER) schemes or just as a health monitoring mechanism. The technique is easily applicable also in hardware peripherals which take input data in communication bursts and cannot accept new data unless they have processed them.

Besides the errors in memory arrays, this scheme can detect even addressing problems and metastability related problems in dual clock domain FIFOs.

In particular, the contributions of this paper are the following:

- A low cost fault detection mechanism for FIFOs is proposed. It allows for low cost error detection, at the expense of greater latency. It eliminates the state overhead of standard horizontal parity codes, while keeping combinational area low and decreasing critical path overheads and power consumption.
- The mechanism was modeled and synthesized using Verilog HDL and its area, power and delay overheads were evaluated thoroughly, and compared to standard parity protection schemes. Single clock and dual clock domain FIFOs were considered.
- The mechanism's fault coverage is analytically estimated.

The remainder of the paper is organized as follows. Section 2 introduces the proposed mechanism and mentions its capabilities and the involved overheads. Section 3 presents experimental results regarding the implementation of the proposed technique in 90 nm ASIC technology, as well as detection latency results in some example systems. Section 4 presents results regarding the fault coverage of the mechanism. Finally, Section 5 lists the related work and Section 6 concludes the paper.

2. Protection mechanism

The protection technique is evaluated both for single clock and dual clock domain FIFOs. First we describe how the FIFO operates (in each case) and then we present how the proposed protection mechanism is incorporated and how it differs from other protection schemes.

2.1. Single clock domain FIFO

2.1.1. Operation

The FIFO memories are implemented using a dual port memory and two pointers delimiting a sliding window, in which the stored entries are kept. In Fig. 1 the pointers *push_addr* and *pop_addr* point to the next location to write and the next location to read, respectively. Upon each access, the respective pointer is incremented by 1 (modulo N for N entries FIFO). When both pointers point to the same location the FIFO is empty, while if *push_addr*+1 equals *pop_addr*, then the FIFO is full.

Fig. 1 shows the internals of a single clock domain FIFO and its operation. It has two interfaces, one for the producer (*push_req_n*, *data_in*) and one for the consumer (*pop_req_n*, *data_out*). The producer asserts (low) the *push_req_n* signal and at the next positive edge of the clock (*clk*) the data is written into the FIFO. *The push_addr* pointer is incremented by 1, to point to the next location to write. Fig. 1(b) shows a scenario where the FIFO was empty and after the push the empty signal was deasserted. Similarly, the consumer asserts (low) the *pop_req_n* signal and at the next positive edge of the clock it can latch the data (assuming an asynchronous read port interface—in case we use synchronous SRAM for the read port, then the output is available at the next positive edge of the clock). The *pop_addr* pointer is incremented in the next positive clock edge. Fig. 1(c) shows a scenario where the FIFO was full and after the pop the *full* signal was deasserted.

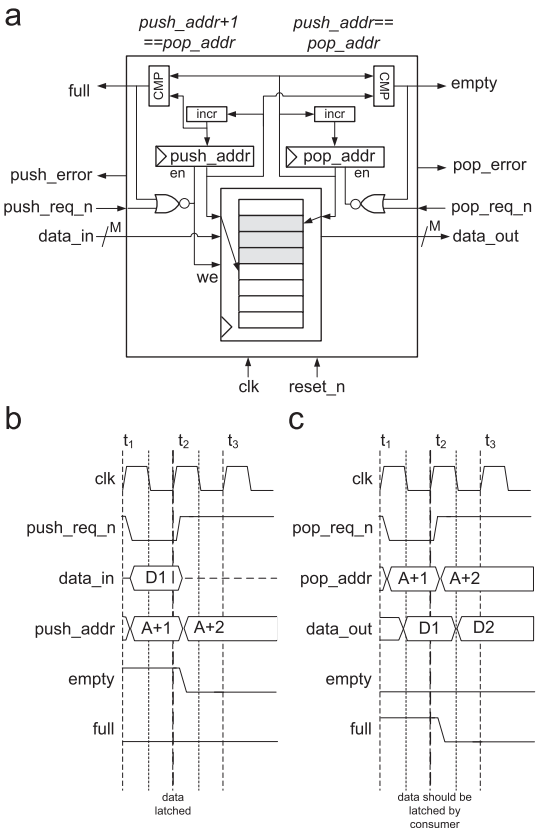

Fig. 1. FIFO operation (single clock).

The error outputs *push_error* and *pop_error* (shown in Fig. 1) are raised when a push is requested and the FIFO is full, or a pop is requested and the FIFO is empty, respectively.

2.1.2. Protection mechanism

Fig. 2 shows the FIFO of Fig. 1 augmented with the proposed protection mechanism. The mechanism maintains one single parity register for the whole memory array, effectively storing the parity of each column. Normally such parity would need a read before every write, in order to be updated, which could be a significant overhead. However, we make use of the FIFO operation and we update the parity register as follows: At every write we perform a xor of the value to be written with the parity register value and we store the new parity (*parity_reg*=*data_in xor parity_reg*). At every read we perform a xor of the read value with value of the *parity_register* (*parity_reg*=*data_out xor parity_reg*). Thus, we effectively store the parity of the active window. Whenever this window is empty (the FIFO is empty), the parity register should be zero, otherwise an error has occurred in some array bits. Fig. 2(a) shows how this algorithm is incorporated in