# A Pipeline Architecture Incorporating a Low-Cost Error Detection and Correction Mechanism

A. Floros and Y. Tsiatouhas
Dept. of Computer Science
University of Ioannina
Ioannina, Greece
{afloros, tsiatouhas}@cs.uoi.gr

A. Arapoyanni
Dept. of Informatics & Telecom.
University of Athens
Athens, Greece
arapoyanni@di.uoa.gr

Th. Haniotakis
Dept. of Electrical & Computer Eng.
Southern Illinois University
Carbondale, USA
haniotak@siu.edu

***Abstract*— High reliability requirements in many modern applications make soft errors an extremely important design aspect and pose new challenges in nanometer technologies. In addition, timing faults that may escape fabrication tests become a real concern in high complexity, high frequency designs. To confront this situation, a concurrent error detection and correction circuit and technique are presented in this work. Their application in pipeline architectures is analyzed and the pipeline error recovery mechanism is illustrated. The proposed scheme is characterized by low silicon area requirements, compared to earlier approaches, and the need of only a single clock cycle for pipeline recovery.**

## I. INTRODUCTION

As modern CMOS nanometer technologies scale down and the complexity of integrated circuits increases, an ongoing difficulty to achieve adequate reliability levels and keep the cost of testing within acceptable bounds is reported [1-2]. The device size scaling, the increase of the operating frequency and the power supply reduction affect circuit's noise margins and reliability. The probability of transient faults generation increases and many times it is hard to achieve the soft error rate (SER) specifications.

Single event upsets (SEUs) and single event transients (SETs), due to alpha particles and cosmic neutrons, play an important role to transient fault and consequently soft error generation in nanometer IC technologies. In addition, timing related transient faults due to crosstalk, power supply disturbance or ground bounce are known mechanisms for soft error generation. Important problems also arise due to timing faults. The increased path delay deviations, due to process variations, and the manufacturing defects that affect circuit speed may result in timing errors that are not easily detectable (in terms of test cost) in high frequency and high device count ICs. Considering also the huge number of paths in modern circuit designs along with the complexity of testing, it is easy to realize that a significant number of defective ICs may pass the fabrication tests. Furthermore, modern systems running at multiple frequency and voltage levels may suffer from timing errors generated by numerous environmental and process related (and many times data dependent) variabilities that can affect circuit performance [3]. Consequently, concurrent testing techniques for error detection and correction are becoming mandatory in order to achieve acceptable levels of error robustness and meet reliability requirements.

Duplication and triplication techniques and self-checking design [4] are widely used to achieve systems reliability. In addition, soft and/or timing error detection techniques have been proposed in the open literature [5-8] that are based on the temporal nature of the transient faults or the delayed response of timing faults to provide error tolerance using time redundancy.

Soft error protection techniques for special purpose, scan based, microprocessor Flip-Flops have been proposed in [2]. These techniques refer to designs where each system Flip-Flop consists of two distinct Flip-Flops (the main Flip-Flop and the scan Flip-Flop). The scan Flip-Flop is modified to operate as a shadow of the main Flip-Flop, latching the same data. According to the first technique, a C-element and a keeper, based on a cross couple inverter pair, are added at the output of each system Flip-Flop to ensure that any soft error that may affect the main Flip-Flop will not be propagated to the logic stage that follows. This way a 20 times reduction in SER is reported. An alternative technique is based on the use of a XOR gate to compare the outputs of the Flip-Flop pair and detect possible soft errors in the system Flip-Flop. Then, three extra logic gates are exploited to enable the trapping of any error indication signal in the scan Flip-Flop. This error indication signal is shifted out using the existing scan path in order to activate system recovery through re-execution. The main drawback of these techniques is the high silicon area cost even in case of microprocessors where pairs of Flip-Flops are incorporated in the standard design. Furthermore, in the second technique, although the global routing of error signals is reduced reusing the scan facilities, there is an extra high cost in error detection latency.

Recently, a pipeline architecture (named *Razor*) with timing error detection/correction for low power operation of systems exploiting dynamic voltage scaling has been proposed in [3]. Fig. 1 illustrates the Razor Flip-Flop that is the basic element of the architecture in the construction of the stage registers. It consists of the main Flip-Flop, an

This work is co-funded by the European Union in the framework of the program "Pythagoras II" of the Hellenic Ministry of Education (funded by 25% from national sources and 75% from the European Social Fund – ESF).

assistant shadow latch, a multiplexer (MUX) and a XOR gate as comparator. As in [7], a delayed clock *D_CLK*, with respect to the circuit clock *CLK*, is used to control the shadow latch and capture the response of the combinational logic.

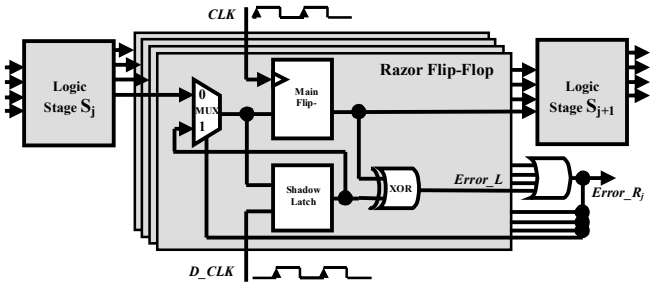


Figure 1. The Razor Flip-Flop

In the error free case both the main Flip-Flop and the shadow latch will capture the same data. The XOR gate compares the outputs of the main Flip-Flop and the shadow latch and remains low while the pipeline operation continues in the normal mode. In case of a delay in the evaluation of the logic stage $S_j$ that exceeds circuit specifications, erroneous data are latched in the main Flip-Flop while the shadow latch will capture the correct (delayed) data, since it operates with a delayed clock. Consequently, the XOR output will rise to high indicating the detection of an error. The generation of a timing error in a clock cycle i+1 at a pipeline stage $S_j$ implies that the data of stage $S_{j+1}$ in the following cycle i+2 are incorrect and must be flushed. This action is quiet easy to be accomplished since the shadow latch contains the correct data without the need to re-execute them through the failing stage. These correct data are injected into the pipeline in the next cycle i+3 allowing stage $S_{j+1}$ to compute the correct responses.

In the Razor architecture two possible approaches for pipeline error recovery have been adopted. The first one is the clock gating where in case of an error detection the entire pipeline stalls by gating the next global clock edge for one cycle. This period is exploited by each stage to re-compute its result using the correct data of the shadow latch. The second approach used in Razor is the *counterflow* pipelining which is based on the namesake processor architecture [9]. Note that in the Razor case there is a high silicon area cost mainly associated with the use of an extra latch for each system Flip-Flop.

In this paper, we present a low-cost pipeline architecture that incorporates concurrent soft and timing error detection and correction capabilities with the minimum error detection latency and a single clock cycle penalty for error recovery. The proposed approach requires only a XOR gate, for error detection, and a multiplexer, in latch configuration for error correction, per system Flip-Flop. Thus, the silicon area requirements are reduced drastically, compared to earlier techniques. The paper is organized as follows. In Section II the proposed (*Blade*) error detection/correction architecture is presented and the pipeline recovery mechanism is analyzed. In Section III experimental results from electrical simulations on a pipeline structure are presented to demonstrate the proposed approach and explore its efficiency. Finally, the conclusions are drawn in Section IV.

## II. THE BLADE PIPELINE ARCHITECTURE

### A. Error detection and correction

Fig. 2a presents the proposed Blade Flip-Flop for the register of a pipeline stage. Compared to the Razor Flip-Flop the shadow latch has been eliminated. Instead, the multiplexer with its output connected to one of its inputs plays the role of the extra memory element (latch), while the XOR gate directly compares the data from the input M and the output Q of the main Flip-Flop.

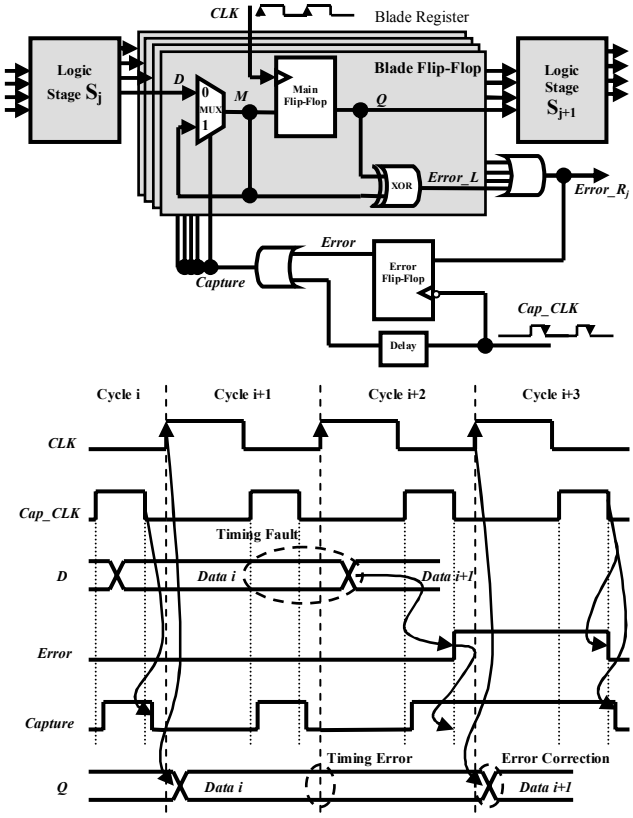


Figure 2: a) The Blade Flip-Flop and b) Blade Flip-Flop operation with a timing fault in cycle i+1 and recovery in cycle i+3.

The memory state of the multiplexer is activated by the *Capture* signal which in the error free case is controlled by the *Cap_CLK* signal, a delayed version of the clock signal *CLK* with a lower duty cycle. An OR gate is used to provide the register error indication signal *Error_*$R_j$ from the local error signals (*Error_L*) of the XOR gates in the Blade Flip-Flops of a register. Finally, the error indication signal *Error_*$R_j$ is captured in a single Flip-Flop (error Flip-Flop) at the falling edge of the *Cap_CLK* signal. When the *Cap_CLK* signal is high the *Capture* signal is activated (turns also to high) and the MUX latch enters the memory state; else the MUX latch is transparent. The time interval that the *Capture*

signal is active must coincide with the time interval where the D inputs of the Blade Flip-Flops, in all stage registers, change values due to an earlier evaluation of the pertinent logic stages according to the circuit specifications. Any signal transition at the D inputs of the Blade Flip-Flops, outside this time interval, is considered as violation of the timing specifications and must be detected. Obviously, the deactivation of the *Capture* signal (its falling edge), and consequently of the *Cap_CLK* signal, must take place before the main Flip-Flop's setup time plus the MUX delay so that valid data are present at the inputs M of the main Flip-Flops at the triggering edge of the clock *CLK*.

In Fig. 2b the operation of the Blade Flip-Flop is presented. In clock cycle i, the response of the logic stage $S_j$ is within the timing specifications of the circuit. Consequently, after the triggering edge of the clock *CLK* both the data input M and the output Q of the main Flip-Flop will carry the same value up to the falling edge of the *Capture* signal. Thus, the XOR output *Error_L* as well as the subsequent signal $Error\_R_j$ will be both zero at the triggering edge of the *Cap_CLK* signal on the error Flip-Flop. In that case, the pipeline's operation remains unaltered (*Error*=low). In the next cycle i+1 a timing fault occurs due to a delayed response of the stage $S_j$. Thus, a timing error is generated at the next triggering edge of the clock *CLK* and a transition occurs at the D input of a Blade Flip-Flop, inside cycle i+2, after this triggering edge and before the activation of the *Capture* signal. Since the MUX latch is transparent during this time interval, this transition passes to the M line. In that case, the comparison by the XOR gate of the MUX latch valid data with the erroneous data of the main Flip-Flop turns the local error signal *Error_L* to high and generates a register error indication signal $Error\_R_j$. Thus, the triggering (falling) edge of the *Cap_CLK* signal captures a high value at the output of the error Flip-Flop. This high value extends the active duration of the *Capture* signal beyond its original falling edge, keeping all MUX latches in the memory state. At this point the error has been detected. In addition, all the MUX latches hold the correct (valid) responses of the $S_j$ logic state for the cycle i+1. The new responses of $S_j$ at the cycle i+2 are blocked at the D inputs of the Blade Flip-Flops. Entering the next cycle i+3, the triggering edge of the clock *CLK* moves the valid data of the MUX latch to the main Flip-Flop and makes them available to the next pipeline stage $S_{j+1}$. Consequently, the error is corrected.

According to the above analysis, if a timing error occurs in a pipeline stage $S_j$ during a particular clock cycle, then the data in the subsequent stage $S_{j+1}$ are incorrect, during the next clock cycle, and must be flushed from the pipeline. However, the MUX latch contains the correct data without the need to re-execute the operation in the $S_j$ stage. Thus, the $S_{j+1}$ stage re-executes the operation using the correct input data with only one-cycle penalty.

The speed penalty introduced by the proposed topology is equal to the delay of the MUX at the input of the Blade Flip-Flop. However, this cost is also present in the case of the Razor topology. Moreover, the silicon area requirements of the Blade Flip-Flop are drastically reduced due to the elimination of the shadow latch used in the Razor Flip-Flop. Finally, taking a closer look at the Razor circuit it is easy to see that the error Flip-Flop must be also present; however in Fig. 1 it is omitted for simplification reasons.

## B. *Pipeline recovery*

In case of error detection a pipeline state recovery action must follow. Fig. 3 illustrates the pipeline recovery mechanism. The event of a timing error in a logic stage (lets say ID) generates an error indication signal *Error_R2* at the following Blade register. This means that the results of the next stage are incorrect (as indicated in Fig. 3b) since its input data are not valid.

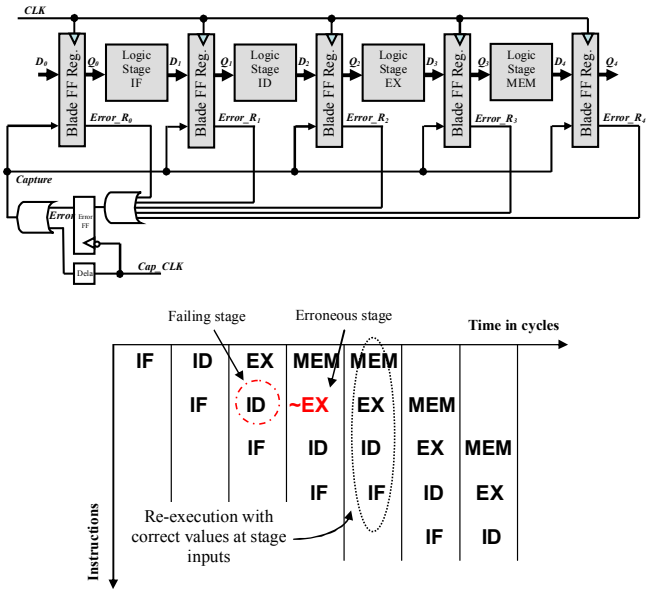


Figure 3. a) Pipeline organization and b) Pipeline recovery

The error indication signal is latched by the error Flip-Flop and the *Capture* signal remains high keeping all the MUX latches of the Blade Flip-Flops in all stage registers in the memory state. Thus, in the next clock cycle every stage is allowed to re-compute its result using the correct data stored in the MUX latches. Note here that there is no need for the failing stage to re-compute the response of the cycle where the failure occurred since the correct responses are already available in the MUX latches that follow. The Blade Flip-Flop based pipeline architecture can tolerate any number of errors in a clock cycle since all stages re-evaluate their results with correct data at their inputs. In case that one or more stages fail in each clock cycle, the pipeline will continue to run at half of the normal speed.

Referring to the analysis of the Blade architecture, there is no need to apply main clock gating to accomplish pipeline recovery. Moreover, although the counterflow pipeline approach can be also applied in the Blade architecture, there is no need to proceed with it. This is due to the fact that the pipeline performance is not affected by the recovery

mechanism since there is not any prohibitive delay in the feedback path from the error indication signal generation to the redirection of the MUXs' inputs. The MUXs in the Blade Flip-Flops of every register are set, by the *Capture* signal, to the memory state independently of the error signal generation. Thus, at the time an error indication signal *Error* is latched in the error Flip-Flop, the *Capture* signal is already active (high) keeping the MUX latches in the memory state. The error indication signal simply extends the active state of the *Capture* signal until the next triggering (falling) edge of the *Cap_CLK* signal, that is for a time duration equal to a clock period. Consequently, the following triggering edge of the clock *CLK* injects the correct data from the MUX latch into the pipeline, allowing the "swerved" instruction to continue. Later instructions inside the pipeline are not flushed and continue to run after recovery. Hence, only a single cycle is required in the Blade architecture for pipeline recovery as it is shown in Fig. 3b.

## III. Design issues – Simulation results

The proposed Blade error detection and correction approach was implemented in a 64-bit four stages pipeline structure, in a 0.18μm CMOS technology ($V_{DD}$=1.8V), with 200MHz clock frequency. The delay of the *Cap_CLK* signal is 2.5ns and its on-time duration is equal to 2ns.

In Fig. 4 simulated waveforms are presented. A timing error is injected at the second stage of the pipeline. As result, a delayed response appears at the D input of the blade Flip-Flop, after the triggering edge of the *CLK*. This response is propagated to the M input of the main Flip-Flop since the MUX latch is transparent (*Capture*=low) in this time interval. Next, the *Capture* signal is activated and the MUX latch captures the correct data on M. The XOR gate detects the difference between M and Q (due to the erroneous data on Q) and raises signal *Error_R2* (not shown) to high. Consequently, the triggering edge of *Cap_CLK* raises the global *Error* signal to high that holds the *Capture* signal active and the MUX latch in the memory state within the next clock cycle. In this cycle the pipeline re-executes the response with the correct data that are available in the MUX latch. After that the pipeline continues its operation with only one cycle penalty. Note that the use of the delayed *Cap_CLK* signal may result in the corruption of the data in the MUX latch due to short paths in the combinational logic. To prevent data corruption a minimum path delay constraint is considered in the design. In order to meet this constraint buffers may be added (like in the Razor case) and/or minimum size plus high-threshold voltage transistors may be used during logic synthesis to slow down fast paths. The minimum path delay constraint is equal to the delay of the *Capture* signal, with respect to the system clock, plus the hold time of the MUX latch. A trade-off arises. A large value for the minimum path delay constraint may increase the number of the required buffers in the design and consequently the silicon area penalty. On the other side, a small value for this delay constraint reduces the error tolerance due to the reduction of the maximum detectable signal delay or transient pulse duration.

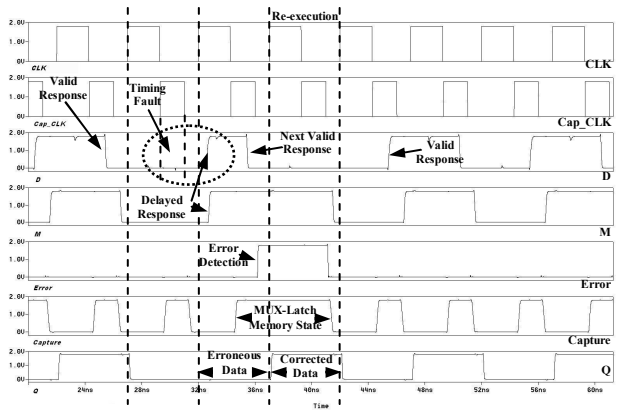


Figure 4: Simulated waveforms from a pipeline stage error detection and correction.

## IV. Conclusions

In this paper we propose an error detection/correction circuit and an architecture for pipeline recovery after an error occurrence. This approach is characterized by low silicon area requirements (lower than earlier approaches in the open literature), small performance penalty (lower or equivalent to this of earlier approaches) and the minimum cost, of only one clock cycle, for pipeline recovery. Although the proposed technique has been presented for Flip-Flop based pipeline architectures it can be also easily adapted in latch based ones. In general, it can be applied to any sequential circuit. Finally, the proposed topology can be used to support architectures that exploit dynamic voltage scaling for low power operation.

## References


[1] R. Wilson and D. Lammers, "Soft Errors Become Hard Truth for Logic," *EE Times*, 3 May 2004 (available at http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=19400052).

[2] S. Mitra, N. Seifert, M. Zhang, Q. Shi and K. S. Kim, "Robust System Design with Built-In Soft-Error Resilience," *IEEE Computer,* vol. 38, no. 2, pp. 43–52, 2005.

[3] T. Austin, D. Blaauw, T. Mudge and K. Flautner, "Making Typical Silicon Matter with Razor," *IEEE Computer,* vol. 37, no. 3, pp. 57–65, 2004.

[4] M. Nicolaidis and Y. Zorian, "On-Line Testing for VLSI – A Compendium of Approaches," *Journal of Electronic Testing: Theory and Applications*, vol. 12, no. 1-2, pp. 7-20, 1998.

[5] C. Metra, R. Degiampietro, M. Favalli and B. Ricco, "Concurrent Detection and Diagnosis Scheme for Transient, Delay and Crosstalk Faults," *5th IEEE On-Line Testing Workshop*, pp. 66-70, 1999.

[6] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," *VLSI Test Symp.,* pp. 86-94, 1999.

[7] L. Anghel and M. Nicolaidis, "Cost Reduction and Evaluation of Temporary Faults Detecting Technique," *Design Automation and Test in Europe Conference*, pp. 591-598, 2000.

[8] S. Matakias, Y. Tsiatouhas, A. Arapoyanni, and Th. Haniotakis, "A Circuit for Concurrent Detection of Soft and Timing Errors in Digital CMOS ICs," *Journal of Electronic Testing: Theory and Applications*, vol. 20, no. 5, pp. 523-531, 2004.

[9] R.F. Sproull, I.E. Sutherland and C.E. Molnar, "The Counterflow Pipeline Processor Architecture," *IEEE Design and Test of Computers*, vol. 11, no. 4, pp. 48-59, 1994.