Timing Error Detection and Correction by Time Dilation

Andreas Floros, Yiorgos Tsiatouhas, Xrysovalantis Kavousianos

Department of Computer Science

University of Ioannina

Abstract—Timing failures in high complexity - high frequency integrated circuits, which are mainly caused by test escapes and environmental as well as operating conditions, are a real concern in nanometer technologies. The Time Dilation design technique supports both on-line (concurrent) error detection/correction and off-line scan testing. It is based on a new scan Flip-Flop and provides multiple error detection and correction at the minimum penalty of one clock cycle delay at the normal circuit operation for each error correction. No extra memory elements are required, like in earlier design approaches in the open literature, reducing drastically the silicon area overhead, while the performance degradation is negligible since no extra circuitry is inserted in the critical paths of a design.

Keywords: On-Line Testing, Concurrent Testing, Timing Errors, Error Detection, Error Correction.

1. CMOS NANOTECHNOLOGIES AND TIMING ERRORS

As modern CMOS technologies scale down in the nanometer era and the complexity of integrated circuits and systems 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 operating frequency increase and the power supply reduction affect circuits' noise margins and reliability. The probability of transient faults generation increases and many times it is hard to achieve error rate specification levels.

Various mechanisms like crosstalk, power supply disturbance or ground bounce have been accused for timing error generation. The increased path delay deviations, due to process variations, and the manufacturing defects that affect circuit speed may also result in timing errors that are not easily detectable (in terms of test cost) in high frequency and/or high device count ICs. The already complex testing process can not sufficiently exercise the huge number of paths in modern circuit designs, and thus it can not effectively screen out all timing related defective ICs. Consequently, a considerable part of defective ICs may escape the fabrication tests. Additionally, and for the same reasons, timing verification turns to be a hard task escalating the probability of timing failures in a design. Furthermore, modern systems running at multiple frequency and voltage levels may suffer from an increased timing error rate due to numerous environmental and process related as well as data dependent variabilities that can affect circuit performance. Besides, *dynamic voltage scaling* (*DVS*) techniques for low power operation that reduces power supply voltage with marginal performance degradation have been proposed in the literature [3]. These exploit timing error detection and correction mechanisms to overcome increased timing error rates. In addition, transistor aging problems significantly impact the performance of nanometer circuits resulting in the appearance of timing errors during their normal lifetime [4-5]. Such an example is the *Negative Bias Temperature Instability* (*NBTI*) induced aging of PMOS transistors which degrades their threshold voltage over time increasing path delays. From the above, it is evident that concurrent on-line testing techniques for timing error detection and correction are becoming mandatory in order to achieve acceptable levels of error robustness and meet reliability requirements.

2. TIMING ERROR DETECTION AND CORRECTION

Timing failures in a combinational logic circuit result in delayed responses at its outputs. As it is shown in Figure 1, in case of a delayed response arrival, after the triggering edge of the system clock *CLK*, the memory element will capture an erroneous value and a timing error is generated.

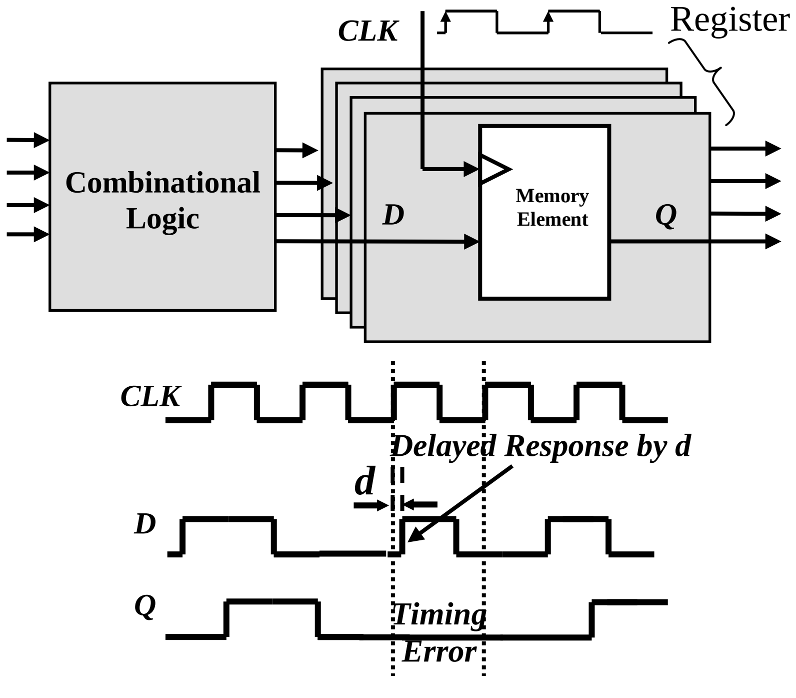


Figure 1. Timing error generation

Various timing error detection techniques have been proposed in the open literature [6-11] that are based on the delayed response of timing faults to provide error tolerance using time redundancy techniques. A well known error detection scheme is based on the use of a comparator that is realized by a simple XOR gate [8-

9, 11]. The monitoring circuitry consists of an additional memory element plus a XOR gate for every memory element (main latch or Flip-Flop) in the design (see Figure 2a). The secondary memory element is clocked by a delayed version of the system clock that feeds the main memory element. This delay is equal to the maximum signal delay (d_{max}) that must be tolerated in order to achieve an acceptable level of timing error rate, plus the setup time of the used memory elements (t_{su}). Thus, the secondary memory element captures a delayed version of the data stored in the main memory element. In the presence of a timing error the stored data in the two memory elements differ, while the secondary memory element holds the correct delayed response of the combinational logic. The XOR gate "compares" the contents of the two memory elements and in case of discrepancy it raises its output to high indicating the error detection. The local error indication signals (*Error_L*) are collected by an OR gate (realized as an OR tree) to generate a global error indication signal (*Error_R*). This signal can be exploited to achieve error tolerance by performing a retry procedure after each error detection. During the retry operation the period of the system clock must be increased to provide the necessary time for correct response evaluation.

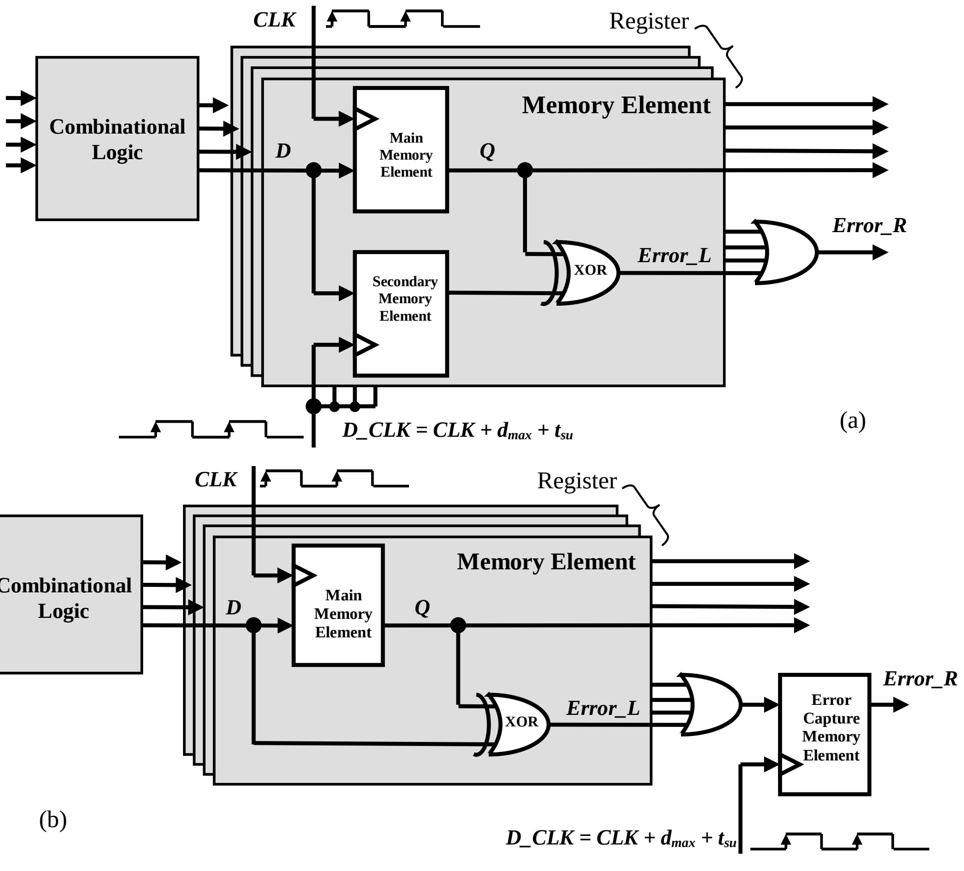


Figure 2. Timing error detection: a) memory duplication and b) cost efficient design

Alternatively, a cost efficient approach is to use only the XOR gate for error detection as it is shown in Figure 2b [8]. The XOR gate compares the data input and output signals of the main memory element for a predefined time period after the triggering edge of the system clock. This time period is also equal to the maximum signal delay that must be tolerated plus the setup time of the memory elements. In case of discrepancy between the two signals, the error indication signal raises at the XOR output. Possible very fast paths with propagation times close to or less than $d_{max}+t_{su}$ must be excluded from the timing error monitoring process, since they will induce false alarms. In general, fast paths with propagation times less than the system clock period minus ($d_{max}+t_{su}$), with a proper tolerance, can be also excluded.

2.1 The Razor Pipeline Architecture

A pipeline architecture (named *Razor*) with timing error detection and correction capabilities, targeting the substantial energy reduction of integrated circuits exploiting dynamic voltage scaling, has been presented in [3]. According to this architecture, the stage registers are constructed using the Razor Flip-Flops. Figure 3 illustrates a Razor Flip-Flop, which consists of the main system Flip-Flop plus an assistant shadow latch, a multiplexer (MUX) and a XOR gate. As discussed earlier, the shadow latch captures, with a proper delay with respect to the main Flip-Flop, the responses of the combinational logic. The XOR gate acts as a comparator and compares the outputs of the main Flip-Flop and the shadow latch.

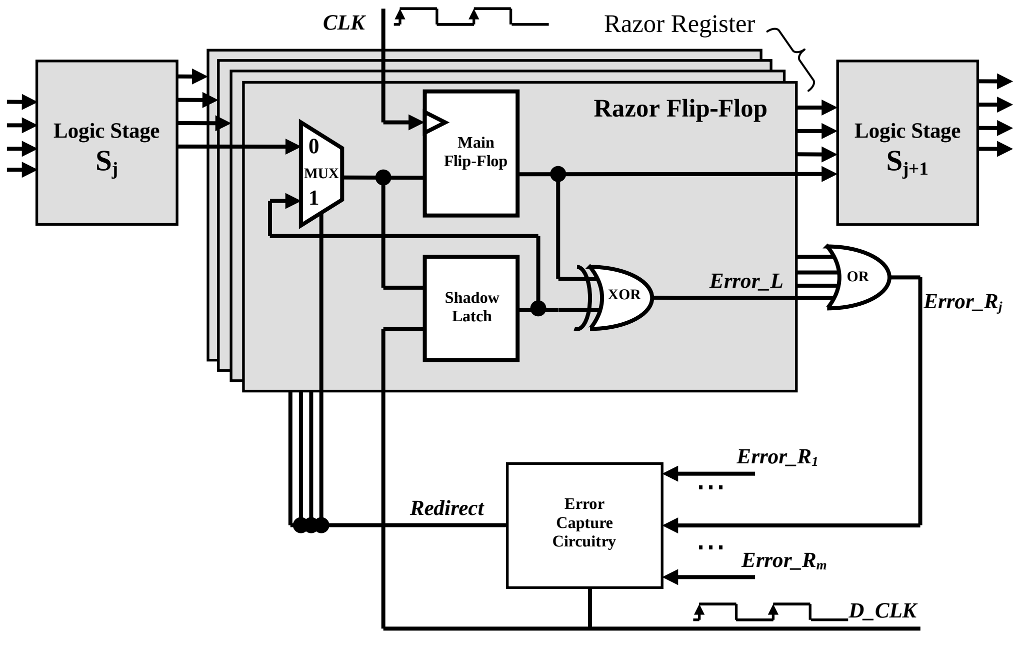
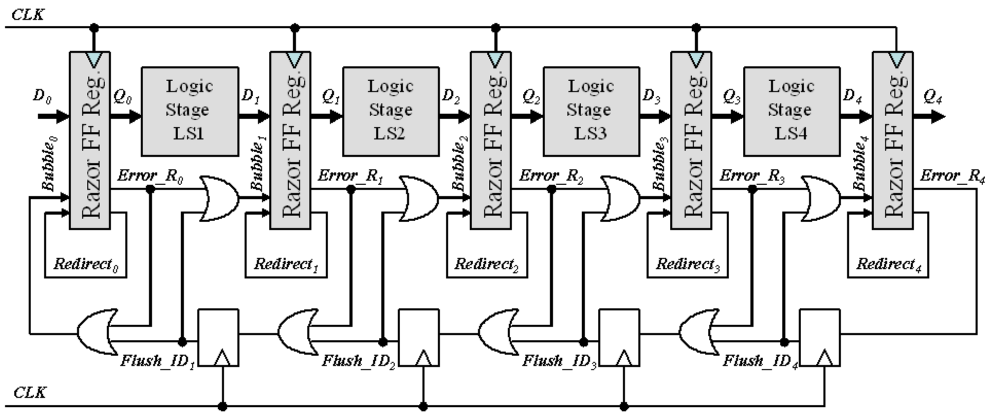
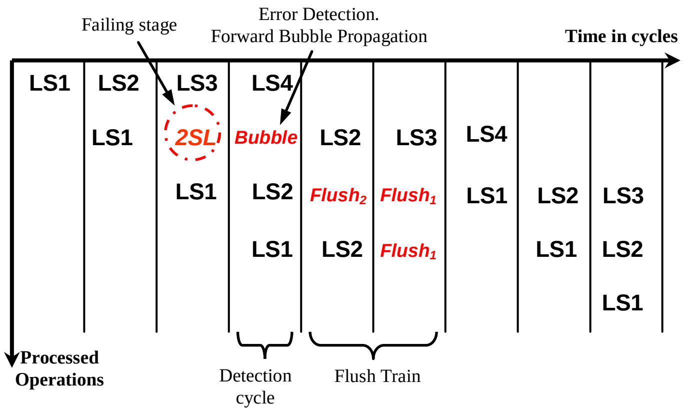


Figure 3. The Razor timing error detection and correction design approach

In the error free case both the main Flip-Flop and the shadow latch will capture the same data. The comparison by the XOR gate provides a low local error indication signal (*Error_L*) and the pipeline continues to operate 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 (*Error_L*) 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 easy to be accomplished since the shadow latch contains the correct data without the need to re-compute them through the failing stage. The local error indication signal *Error_L* activates the register error indication signal $Error\_R_j$ which is captured by the Error Capture Circuitry. This in turn sets the *Redirect* signal to high enabling the shadow latch to feed the main Flip-Flop with the correct data. These are injected into the pipeline in the next cycle (i+3) allowing stage S_{j+1} to compute the correct responses.



(a)



(b)

Figure 4. Razor counterflow recovery: a) pipeline architecture and b) pipeline operation

In the Razor architecture two approaches for pipeline error recovery have been adopted [3]. The first one is the clock gating technique 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 latches. The second approach used in Razor is the *counterflow* pipelining which is based on the namesake processor architecture [12]. This technique is illustrated in Figure 4 and is characterized by negligible timing constraints in the pipeline operation at the expense of few cycles, depending on the pipeline depth, for error recovery. When a register error indication signal is generated, there are two actions that follow. First, a *Bubble* signal is generated to nullify the computation in the following stage. This signal indicates to all subsequent stages that the pipeline slot is empty. Second, a flush train is activated by asserting the ID of the stage generating the error indication signal. In the next cycle the correct data of the corresponding register shadow latches are injected into the pipeline allowing the errant instruction to continue its execution. In parallel, the flush train propagates the ID of the failing stage in the opposite direction to this of the instructions flow. At each stage that the flush train visits, the computation is nullified. When the first stage of the pipeline is reached the pipeline restarts its operation with the instruction that follows the failing one.

The Razor approach suffers from high silicon area cost since for every main Flip-Flop an extra latch, a multiplexer and a XOR gate are required. In addition an extra clock signal is used.

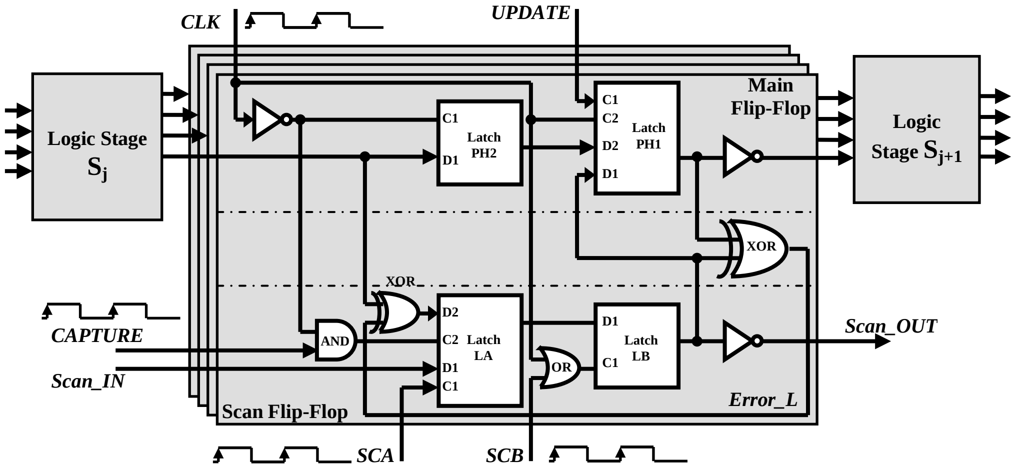


Figure 5. Error trapping scan cell

2.2 Scan Based Error Detection and Correction

Soft error detection and correction techniques for special purpose scan Flip-Flops in microprocessor circuits have been proposed in [1]. These techniques are suitable in designs where each system Flip-Flop consists of a pair of Flip-Flops (i.e. the main Flip-Flop and the scan Flip-Flop as it is shown in Figure 5) and can be also exploited to cover timing errors. The scan Flip-Flop is modified to operate as a shadow of the main Flip-Flop, latching the same data with a proper delay as discussed earlier (signals *CAPTURE* and *SCB* are delayed with respect to the system clock *CLK*). A XOR gate is used to compare the outputs of the Flip-Flop pair and detect possible errors in the system Flip-Flop. Three additional logic gates (a second XOR, an OR and an AND) are used in order to enable the trapping of any error indication signal (*Error_L*) in the pertinent scan Flip-Flop. This error indication is shifted out using the existing scan path in order to activate system recovery through re-execution. The main drawbacks of this technique are: a) the high silicon area cost due to Flip-Flop duplication and the insertion of extra logic gates, b) the performance degradation due to the complexity of the main Flip-Flop, c) the large number of control signals and d) although the global routing of error signals is reduced reusing existing scan facilities, there is a high penalty in error detection latency.

3. THE TIME DILATION SCAN ARCHITECTURE

Recently, a low cost pipeline architecture has been proposed in [13] that is characterized by the ability to detect and correct timing errors. This architecture utilizes only a multiplexer and a XOR gate per system Flip-Flop reducing drastically the silicon area cost, while only a single clock cycle is required for error correction. This technique has been extended in [14] to scan designs forming the *Time Dilation* scan architecture.

Figure 6 illustrates the classical scan register configuration which is based on standard scan Flip-Flops. All scan Flip-Flops are connected together as one or more scan registers. The *Scan_IN* input of a scan Flip-Flop is driven by the Q output of the preceding scan Flip-flop in the shift register. When the *Scan_EN* signal is “high” the circuit is in the scan mode of operation, for testing purposes, and the scan Flip-Flops are driven by the *Scan_IN* inputs, else they are driven by the *D* inputs capturing the response data of the combinational logic.

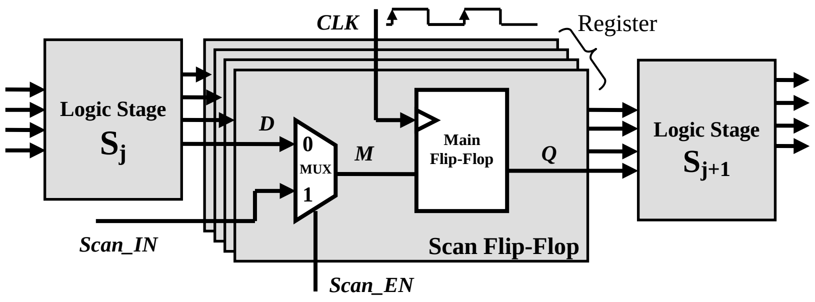


Figure 6. The standard scan Flip-Flop design

3.1 The Time Dilation Scan Flip-Flop

The scan Flip-Flop used in the Time Dilation (TIMED) architecture is presented in Figure 7. The TIMED Flip-Flop provides the capability of error detection and correction by appending only a multiplexer (MUX-B) and a XOR gate in the structure of the standard (main) scan Flip-Flop. This hardware overhead is much lower than this of the next most attractive choice, the Razor topology, where except of the above two cells an additional shadow latch is required. Although we will present for convenience the application of the Time Dilation technique in pipeline architectures, it can be also applied in any sequential circuit design.

When the scan enable signal (*Scan_EN*) is “high” the TIMED Flip-Flop operates like a scan Flip-Flop to support the pertinent off-line testing activity. In the normal mode of operation (*Scan_EN*="low") the TIMED Flip-Flop behaves like an ordinary Flip-Flop enhanced with the ability to detect and correct timing errors. The XOR gate is used to directly compare the data at the *M* input and the *Q* output of the Main Flip-Flop for error detection, while the two multiplexers and the feedback path from the *M* line to the input of the additional MUX-B forms the required memory element (MUX-latch) that holds valid data for error correction.

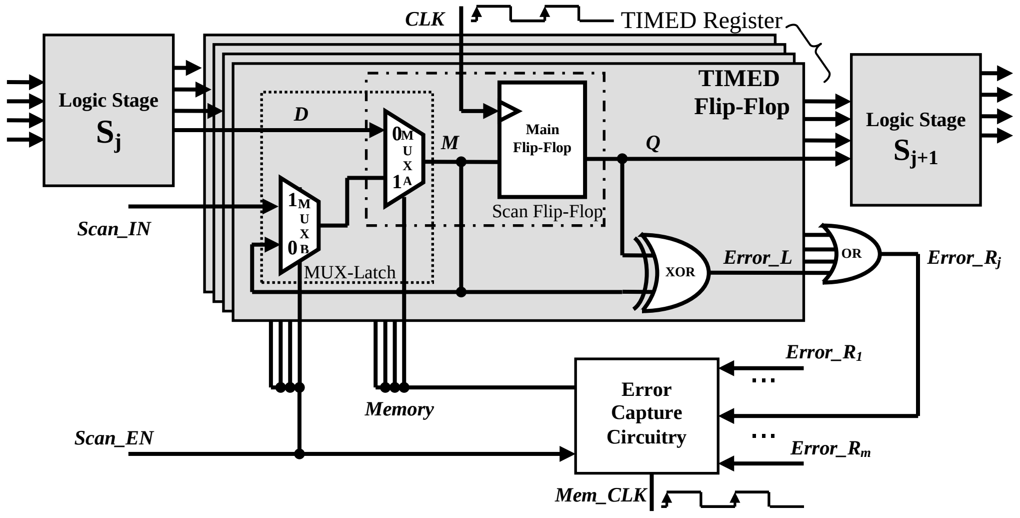


Figure 7. The TIMED Flip-Flop and support circuitry

Briefly, the Time Dilation technique operates as follows. Suppose that a timing error is detected at the inputs of the combinational logic stage S_{j+1}, due to a delayed response of the previous stage S_j. Thus, the response of S_{j+1} will be erroneous and must be corrected. Then, the evaluation time of the circuit is extended by one clock cycle and S_{j+1} is fed with the delayed, but valid, response of S_j that has been captured in the MUX-latch, for error correction.

The MUX-latch is clocked by the *Memory* signal. In the error free case the *Memory* signal is exclusively controlled by the *Mem_CLK* signal, a delayed version of the clock signal *CLK* with a proper duty cycle. When the *Mem_CLK* signal is "high" the *Memory* 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 *Memory* signal is active must coincide with the time interval where new values arrive at the *D* inputs of the TIMED Flip-Flops, in all stage registers, due to an earlier evaluation of the pertinent logic stages according to the circuit specifications. Any signal transition at the *D* inputs of the TIMED Flip-Flops, earlier than the activation time of the *Memory* signal, is considered as violation of the timing specifications and must be detected. Obviously, the deactivation of the *Memory* signal (falling edge), and accordingly of the *Mem_CLK* signal, must occur before the triggering edge of the *CLK* signal and at a time distance at least equal to the delay time of the MUX-A plus the setup time of the Main Flip-Flop.

The XOR gate in the TIMED Flip-Flop detects timing errors and indicates them by setting signal *Error_L* to "high". An OR gate is used to collect the *Error_L* signals and to generate the register error indication signal *Error_R$_j$*. Any register error indication signal is captured by a single Flip-Flop (Error Flip-Flop) triggered by the *Mem_CLK* signal which has been properly delayed. The final error indication signal, *Error,* is used to activate the error correction mechanism.

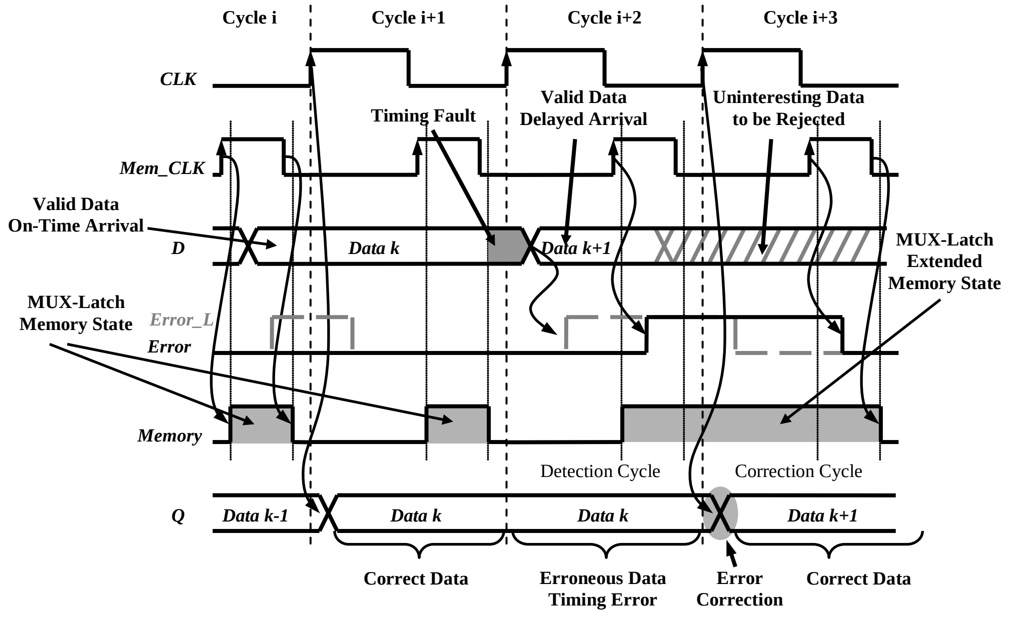


Figure 8. TIMED Flip-Flop operation with a timing error in cycle i+2 and recovery in cycle i+3

3.2 Timing Error Detection and Correction using Time Dilation

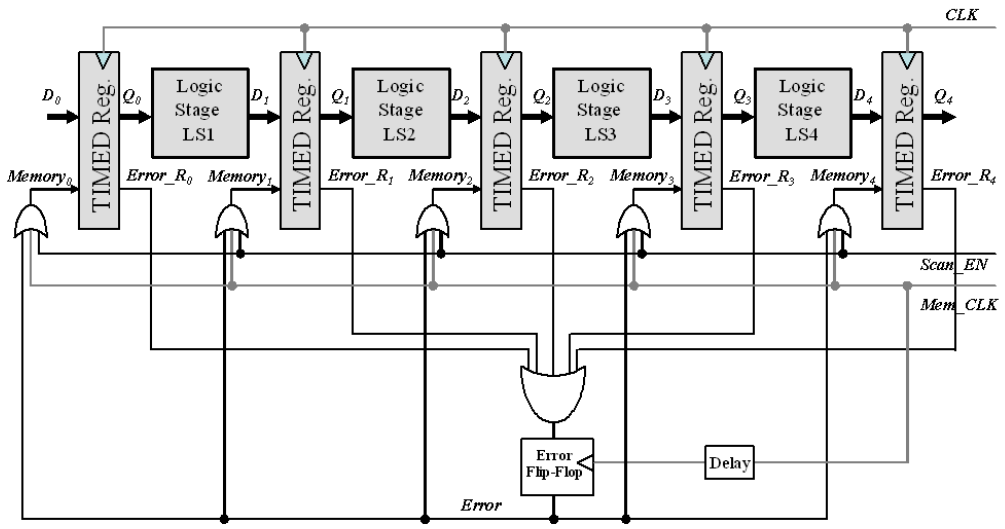
In Figure 8 the operation of the TIMED Flip-Flop is presented. We study the normal mode of operation (not the scan mode) therefore the *Scan_EN* signal is considered always "low". In the ith clock cycle the response of the logic stage S_j is within the timing specifications of the circuit. This means that it occurs during the high state of the *Memory* signal. 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 until the falling edge of the *Memory* signal. Thus, the *Error_L* signal as well as the subsequent *Error_R$_j$* signal will be both zero at the time that the Error Flip-Flop is triggered. In that case, the pipeline's operation remains unaltered (*Error*="low"). In the next cycle (i+1) a timing fault occurs which induce a delayed response of stage S_j. Thus, a timing error is generated at the next triggering edge of the clock *CLK.* The data captured in the TIMED register between the S_j and S_{j+1} stages are erroneous and consequently the response of S_{j+1} stage at the (i+2) cycle will be also erroneous. Moreover, due to the fault, a transition occurs at the *D* input of a TIMED Flip-Flop, inside (i+2) cycle, after the triggering edge and before the activation of the *Memory* signal. Since the MUX-latch is transparent during this time interval, the transition passes to the *M* line. Now the value at the output of the MUX-latch (*M* line) differs from this at the output of the Main Flip-Flop (*Q* line). The first one is the correct response of S_j and the second the erroneous value captured on *Q*. So, the comparison by the XOR gate of the MUX-latch valid data with the erroneous data stored in the Main Flip-Flop sets the local error signal *Error_L* to “high” and generates a register error indication signal *Error_R*$_j$ at the output of the register’s OR gate. Next, the triggering edge of the *Mem_CLK* signal activates the *Memory* signal, setting the MUX-latches in the memory state, and after a proper delay captures the register error indication in the Error Flip-Flop, raising the *Error* signal to “high”. This “high” value will extend the active duration of the *Memory* signal 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 stage for the (i+1) clock cycle. The new responses of the S_j and S_{j+1} logic stages at the (i+2) cycle are blocked at the *D* inputs of the pertinent TIMED Flip-Flops and will be discarded since the response of S_{j+1} is erroneous. Entering the next cycle (i+3), the triggering edge of the clock *CLK* forces the valid data to move from the MUX-latches to the Main Flip-Flops in order to be available to the next pipeline stage S_{j+1}. Consequently, the error is corrected since the logic stage has correct data to perform, inside the (i+3) clock cycle, the failed evaluation of the (i+2) cycle. This is an one cycle penalty for correction. Next, the error indication signals *Error_L*, *Error_R*$_j$ and *Error* turn successively to “low” and the *Memory* signal returns to its routine operation.

According to the above discussion, 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-latches contain the correct data and thus the re-execution of the failed evaluation in the S_j stage is avoided. On the other hand, the S_{j+1} stage re-executes its evaluation using this time the correct input data with only one-cycle penalty in the pipeline operation.

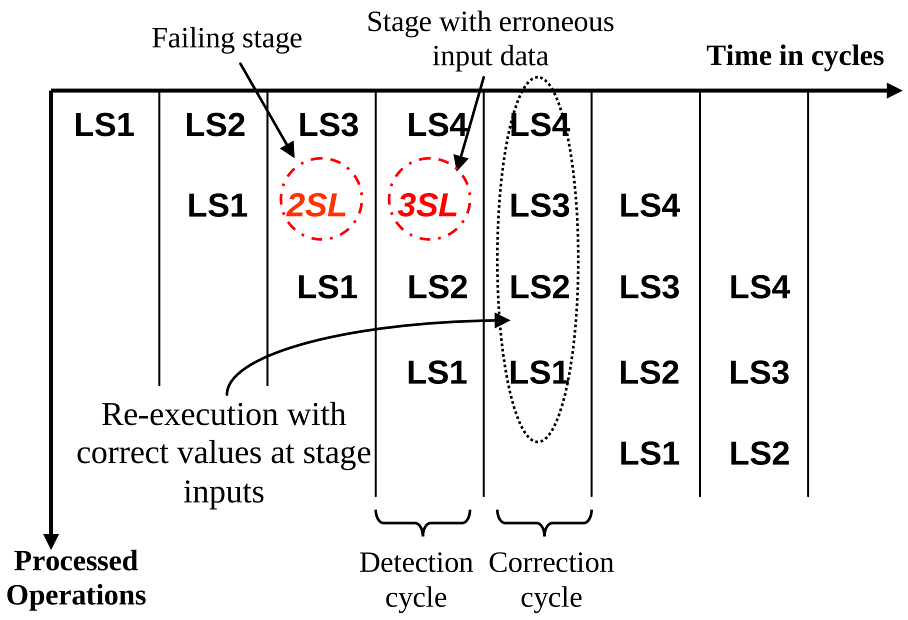
A main characteristic and an advantage of the proposed topology is that no circuitry is inserted in the critical path from the *D* input to the *Q* output of the Flip-Flop or in the distribution path of the clock signal *CLK*. The additional MUX-B is inserted in the scan path which is not critical. A minor performance penalty is introduced by the small parasitic capacitances of the MUX-B and the XOR gate inputs that are driven by the *M* and *Q* signal lines. In addition, note that the silicon overhead of the OR gate at the output of a TIMED register is small (especially when a Domino design style is used), while the rest circuitry (the Error Capture Circuitry) is shared on the whole pipeline and thus its cost is insignificant. The area overhead related to the OR gates and the Error Capture Circuitry is also present in the Razor topology.

3.3 Pipeline Recovery

Every error detection is succeeded by a pipeline state recovery action. Figure 9 illustrates the pipeline recovery mechanism. The event of a timing error in a logic stage (lets say the LS2 stage) generates an error indication signal *Error_R*$_2$ at the following TIMED register. This means that the response of the next stage LS3 at the subsequent clock cycle is incorrect (as indicated in Figure 9b) since its input data are not valid.



(a)



(b)

Figure 9. Time Dilation recovery: a) pipeline organization and b) pipeline operation

The error indication signal is latched by the Error Flip-Flop and the *Memory* signal remains "high" keeping all the MUX-latches of the TIMED Flip-Flops in all stage registers in the memory state. Thus, in the next clock cycle every stage is allowed to re-compute its response using the correct data stored in the MUX-latches. Actually, this seems to be like a "time dilation" in the duration of the failing clock cycle. Note here that there is no need for the failing stage LS2 to re-compute its response in the cycle where the failure occurred since the correct responses are already available in the following MUX-latches. The Time Dilation pipeline architecture can tolerate any number of errors in a clock cycle since all stages re-compute their responses 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 Time Dilation architecture, there is no need to apply main clock gating to accomplish pipeline recovery, neither the Counterflow pipeline design technique [12] as in the Razor case. 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 activation of the memory state of the MUX-latches. The MUX-latches in the TIMED Flip-Flops are set to the memory state, by the *Memory* signal, independently of the generation or not of an error signal. Thus, at the time an error indication signal (*Error*="high") is captured in the Error Flip-Flop, the *Memory* signal is already active ("high") and the MUX-latches are in the memory state. This error indication signal simply extends the active state of the *Memory* signal for one clock period. Consequently, the following triggering edge of the clock *CLK* injects the correct data from the MUX-latches into the pipeline, allowing the "swerved" operation to continue. Later operations inside the pipeline are not flushed and continue to run after recovery. Hence, only a single cycle is required in the Time Dilation architecture for pipeline recovery as it is shown in Figure 9b.

Note that the delay of the *Mem_CLK* signal with respect to the system clock *CLK*, and consequently its duty cycle, must be properly selected to prevent data corruption in the MUX-latches due to possible existence of short paths in the combinational logic. To avoid this, a minimum path delay constraint is considered in the design. In order to meet this constraint in the presence of short paths, gates constructed of minimum size and high-threshold voltage transistors can be used and buffers may be added during logic synthesis (like in Razor [3]) to slow them down. The minimum path delay constraint is equal to the delay of the *Memory* signal with respect to the system clock *CLK*, plus the hold time of the MUX-latch. However, 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.

4. TIME DILATION APPLICATION

The Time Dilation architecture was applied in a 32-bit four stages pipeline datapath, that has been designed in a 90nm CMOS technology (V_{DD}=1V), with 870MHz clock frequency (1150ps period). The TIMED Flip-Flop has been designed in transistor level as a library standard-cell. Since the fastest response of the combinational logic is higher than 400ps, the delay of the *Mem_CLK* signal with respect to *CLK* is set to 300ps and its "on" time duration is equal to 550ps. The extra delay inserted to the *Mem_CLK* signal to drive the Error Flip-Flop is 250ps. Signal delays up to 350ps (30% of the clock cycle) from the triggering edge of the system clock *CLK* can be detected and corrected. The performance penalty introduced in the original scan design with the use of the TIMED Flip-Flop is less than $4^{o}/_{oo}$ and thus it is negligible.

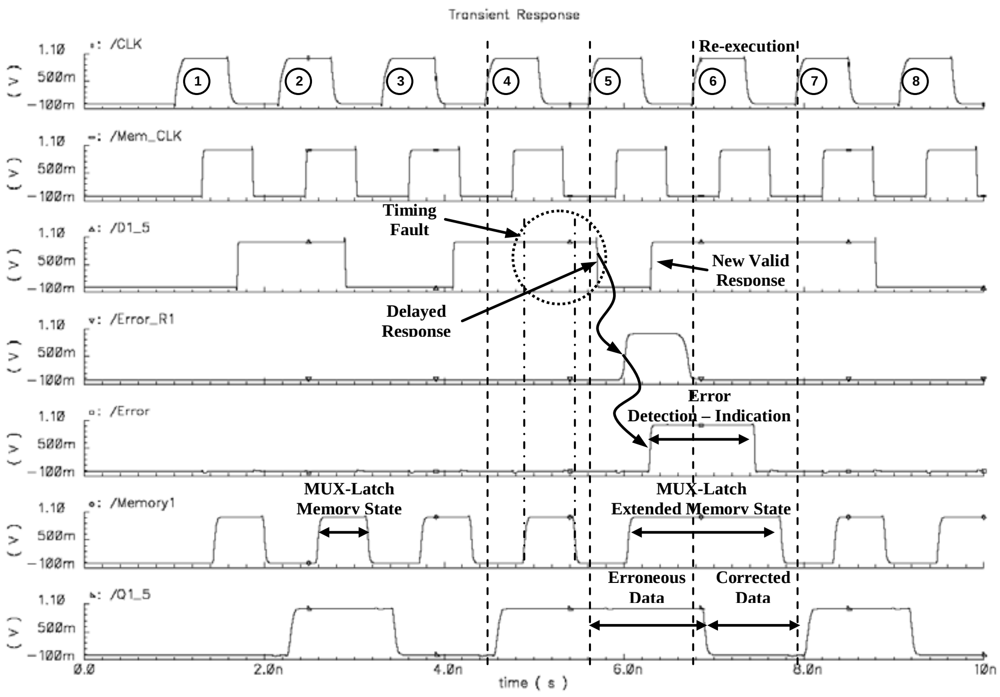


Figure 10. Simulated waveforms from Time Dilation application in a 32-bit pipeline

In Figure 10 electrical simulations using SPECTRE are presented. A timing fault is injected at the first stage of the pipeline during the 4th clock cycle. Consequently, the data captured at the *Q1_5* output of the corresponding TIMED Flip-Flop are erroneous and the same stands for the response of second stage at the 5th cycle. Due to the fault, a delayed response appears at the *D1_5* input of the TIMED Flip-Flop in the 5th cycle, after the triggering edge of *CLK*. This response is propagated to the *M1_5* (not shown) input of the main Flip-Flop since the MUX-latch is transparent (*Memory1*="low") during this time interval. Next, the *Memory1* signal is activated and the MUX-latch captures the correct data on *M1_5*. The XOR gate detects the difference between *M1_5* and *Q1_5* (due to the erroneous data on *Q1_5*) and sets signal *Error_R1* to "high". Consequently, the triggering edge of *Mem_CLK* also forces the global *Error* signal to "high". This extends the memory state of the MUX-latch holding the *Memory1* signal active ("high") within the 6th clock cycle. In this cycle the pipeline re-executes the stage responses with the correct data that are available in the MUX-latches. Thus, the error is corrected and the pipeline proceeds with its normal operation.

5. CONCLUSIONS

Timing error detection and correction techniques are of great importance in today nanometer CMOS technologies. To cope with them, a new scan Flip-Flop design that provides timing error detection/correction capabilities and a pipeline architecture (under the name *Time Dilation*) which exploits this scan Flip-Flop for pipeline recovery after a timing error occurrence, have been proposed. This design approach is characterized by low silicon area requirements (about 24% reduction in Flip-Flop area with respect to *Razor* the most attractive alternative topology), negligible performance penalty and the minimum cost of only one clock cycle for pipeline recovery after each error detection. Although the proposed technique is illustrated for pipeline architectures, it can be applied in general to any sequential circuit.

The Time Dilation technique can be utilized to provide aggressive power reductions in Dynamic Voltage Scaling (DVS) based circuits by tolerating timing errors in critical paths under worst case process and environmental variabilities or the presence of noise sources like di/dt noise in supply voltage and signal crosstalk. Moreover, Time Dilation offers the ability of using more relaxed design constraints or voltage and noise margins to ensure correct operation. Those constraints/margins are inserted to protect a design against uncertainty in circuit model parameters and worst case combination of variabilities. However, such a combination might be very rare or even impossible making this approach overly conservative from the performance point of view and demanding in design effort [3]. With technology scaling, process variations are increased and noise effects are getting more and more serious worsening the required constraints and margins in a design. Time Dilation accounts for both local and global process and temperature variations as well as noise sources that affect timing, eliminating the need to meet severe constraints and apply wide margins to ensure correct operation at a given (desired) performance.

REFERENCES

[1] S. Mitra, N. Seifert, M. Zhang, Q. Shi and K. S. Kim, Robust System Design with Built-In Soft-Error Resilience, *IEEE Computer,* Volume 38, Number 2, pp. 43–52 (2005).

[2] S. Mitra, M. Zhang, S. Waqas, N. Seifert, B. Gill and K-S. Kim, Combinational Logic Soft Error Correction, *IEEE Internatonal Test Conference*, (2006).

[3] T. Austin, D. Blaauw, T. Mudge and K. Flautner, Making Typical Silicon Matter with Razor, *IEEE Computer,* Volume 37, Number 3, pp. 57–65 (2004).

[4] M. Agarwal, B.C. Paul, M. Zhang and S. Mitra, Circuit Failure Prediction and its Application to Transistor Aging, *IEEE VLSI Test Symposium,* pp. 277-284 (2007).

[5] M. Agarwal, V. Balakrishnan, A. Bhuyan, K. Kim, B.C. Paul, W. Wang, B. Yang, Y. Cao and S. Mitra, Optimized Circuit Failure Prediction for Aging: Practicality and Promise, *IEEE International Test Conference*, (2008).

[6] M. Nicolaidis and Y. Zorian, On-Line Testing for VLSI – A Compendium of Approaches, *Journal of Electronic Testing: Theory and Applications*, Volume 12, Number 1-2, pp. 7-20 (1998).

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

[8] Y. Tsiatouhas and Th. Haniotakis, A Zero Aliasing Built-In Self Test Technique for Delay Fault Testing, *IEEE Symposium on Design for Testability of VLSI Systems*, pp. 95-100 (1999).

[9] 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).

[10] 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*, Volume 20, Number 5, pp. 523-531 (2004).

[11] M. Nicolaidis, Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies, *IEEE VLSI Test Symposium*, pp. 86-94 (1999).

[12] R.F. Sproull, I.E. Sutherland and C.E. Molnar, The Counterflow Pipeline Processor Architecture, *IEEE Design and Test of Computers*, Volume 11, Number 4, pp. 48-59 (1994).

[13] A. Floros, Y. Tsiatouhas, A. Arapoyanni and Th. Haniotakis, A Pipeline Architecture Incorporating a Low-Cost Error Detection and Correction Mechanism, *IEEE International Conference on Electronics, Circuits and Systems*, pp. 692-695 (2006).

[14] A. Floros, Y. Tsiatouhas and X. Kavousianos, The Time Dilation Scan Architecture for Timing Error Detection and Correction, *IFIP/IEEE International Conference on Very Large Scale Integration*, pp. 569-574 (2008).