Single Event Upset Tolerance in Flip-Flop Based Microprocessor Cores

Stefanos Valadimas[a], Yiorgos Tsiatouhas,[b,c] Angela Arapoyanni[a] and Adrian Evans[c]

[a]University of Athens, Dept. of Informatics and Telecommunications, 15784 Athens, Greece
[b]University of Ioannina, Dept. of Computer Science, 45110 Ioannina, Greece
[c]TIMA Laboratory, CNRS, 38031 Grenoble, France

***Abstract*—Soft errors due to single event upsets (SEUs) in the flip-flops of a design are of increasing importance in nanometer technology microprocessor cores. In this work, we present a flip-flop oriented soft error detection and correction technique. It exploits a transition detector at the output of the flip-flop for error detection along with an asynchronous local error correction scheme to provide soft error tolerance. Alternatively, a low cost soft error detection scheme is introduced, which shares a transition detector among multiple flip-flops, while error recovery relies on architectural replay. To validate the proposed approach, it has been applied in the design of a 32-bit MIPS microprocessor core using a 90nm CMOS technology.**

Keywords*—*SEUs; Soft error detection and correction; Soft error tolerance.

I. Introduction

Radiation induced transient faults have become a serious reliability threat in nanometer technology integrated circuits and systems. Transistor scaling and lower operating voltages have increased the overall system impact of soft error effects, escalating the soft error rates (SER) [1]. In this context, it is evident that soft error tolerance techniques are becoming necessary to provide robustness and meet system reliability requirements.

Radiation related soft error generation mechanisms are divided into two categories, single event transients (SETs) that affect the combinational logic and single event upsets (SEUs) that affect memory elements (flip-flops and latches) in a design. SETs result in transient pulses that are propagated inside the logic. It is only in the event that a transient pulse reaches the input of a sequential element and is sampled that it becomes an actual soft error. SEUs directly affect the memory elements and change the stored data. The overall effect of SETs is considerably smaller than SEUs [2], [3], due to attenuation and temporal masking phenomena in the propagation of the transient pulse.

Triple modular redundancy (TMR) which is implemented using a voter module is a well known (but high cost) error protection approach. Aiming to reduce the cost, various error detection techniques have been presented in the open literature [4]–[9]. An effective and commonly used scheme is based on a XOR gate comparator [1], [2], [6], [8], [9]. The memory element is modified to incorporate an additional memory element plus a XOR gate. The secondary memory element captures a copy of the data, which may optionally be delayed. In the event of an error the data stored in the two memory elements differ. The XOR gate senses this difference and indicates the presence of the error. An alternative scheme for error tolerance is based on a transition detector [10], [11] to monitor for improper activity at the output of the memory elements and to flag such an invalid transition as an error. In both cases, the local error indication signals are collected through OR trees to generate a global error indication signal. This signal is exploited for error recovery usually by performing architectural replay.

In this work we present a SEU oriented soft error tolerance technique for edge-triggered flip-flop based microprocessor cores. The concept is based on a transition detector for error detection and an asynchronous local error correction mechanism per flip-flop. The paper is organized as follows. In Section II, state of the art in error detection and correction techniques are discussed. Next, in Section III, the proposed soft error tolerance flip-flop is introduced and analyzed, while possible pipeline error recovery mechanisms are presented. In Section IV comparisons with existing flip-flop based solutions are provided. Moreover, simulation results to validate the new scheme are presented by its application to a 32-bit RISC microprocessor core. Finally, Section V concludes this work.

II. State of the Art in Error Tolerance

A number of error tolerance techniques have been proposed for flip-flop and latch based designs. Initially, in the first category, the Razor-I timing error detection and correction technique [8] exploits an additional XOR gate for error detection and a shadow latch for error correction. Soft error protection capabilities of Razor-I are, however, limited. Reduced cost alternatives have been proposed in [12], [13].

The BISER soft error resilience technique [3], [14] uses an extra flip-flop and a C-element per system flip-flop for double sampling and error propagation blocking respectively. This approach can be also applied in latch based designs.

Next, considering latch based designs, the GRAAL architecture has been proposed in [1] and applied to a DSP microprocessor in [15]. The GRAAL concept is based on the XOR comparator for error detection and an additional flip-flop per latch for error correction.

Two error detection techniques for latch based microprocessor designs are analysed in [10] and [16]. The first one uses a transition detector per latch while the second one exploits as earlier the double sampling approach inserting a flip-flop per latch and a XOR comparator. The error recovery mechanism requires clock control to replay errant instructions

This work is supported by the European Social Fund (ESF) and the Operational Program for Educational and Lifelong Learning (NSRF 2007-2013) under the program "THALIS".

at half the clock frequency or utilizes a multiple-issue instruction replay.

In [11] the Razor-II topology is introduced for error tolerance in latch based designs and its application to a 32-bit ARM microprocessor is discussed in [17]. Also in that case a transition detector is used, at the output of the latch, for error detection while error correction is performed through architectural replay.

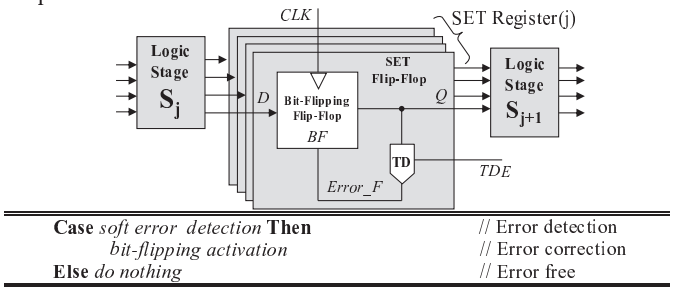
Recently, a latch oriented soft error protection scheme that is based on a linear code for error detection and bit-flipping latches or architectural replay for error correction has been proposed in [18].

III. Soft Error Tolerant Flip-Flop

In this work we deal with soft errors due to SEUs in the flip-flops of a design. The proposed soft error detection and correction technique is based on the fact that after the triggering edge of the clock, plus the propagation delay of the flip-flop ($t_{c\text{-}q}$), the data at its output must remain stable until the next triggering edge of the clock. Thus, if we are able to detect signal transitions at the output of the flip-flop during this time interval, then the related soft error can be corrected by bit-flipping the data stored in the flip-flop.

A. The soft error tolerant (SET) flip-flop

The Soft Error Tolerant (SET) flip-flop concept is presented in Fig. 1. A Transition Detection (TD) unit is exploited and the flip-flop is equipped with an asynchronous bit-flipping option. The TD unit monitors the output (Q) of the flip-flop within a predetermined time window after the triggering edge of the clock *CLK*, which is determined by the *TDE* signal. Within this time window, no signal transitions are expected at the output of the flip-flop. Whenever a signal transition is detected during the monitoring window (caused by a SEU), the TD unit indicates this by raising its output *Error_F*. A correction operation then follows with the *Error_F* signal feeding the bit-flipping activation input (*BF*) of the flip-flop. The erroneous data stored in the flip-flop are thus complemented and the soft error is corrected. During the correction operation the TD unit is deactivated to prevent repeated corrections.



Case *soft error detection* **Then**	// Error detection
bit-flipping activation	// Error correction
Else *do nothing*	// Error free

Fig. 1. The proposed soft error tolerance concept.

B. SET flip-flop implementations

Various bit-flipping flip-flop designs can be considered for the proposed SET flip-flop. Here, we present simple and easily synthesizable implementations, although a custom implementation would be more efficient and will be the subject of future work.

Initially, the bit-flipping scheme proposed by the research team in [13], for timing error tolerance, can be exploited. A multiplexer (MUX) or a XOR gate is inserted at the output of the flip-flop. In the first case, one signal input of the MUX is driven by the flip-flop's output and the other by the complementary value (using a NOT gate). In the second case, one input of the XOR gate is driven by the flip-flop's output. In both cases, the select input of the MUX or the second input of the XOR is driven by the output of the TD unit (*Error_F* signal). Consequently, in case of soft error detection by the TD unit the response of the flip-flop is complemented. However, this scheme requires a memory element inside the TD unit to retain the error indication signal. The drawback is that this memory element increases the silicon area cost.

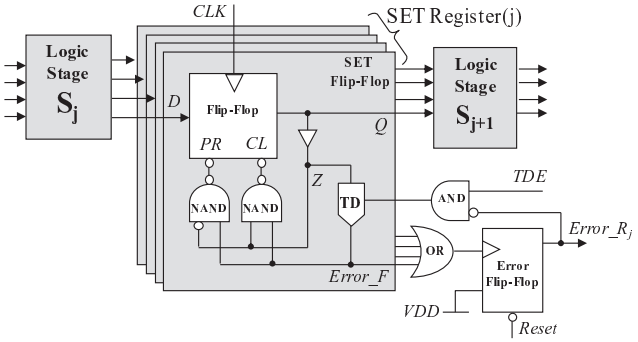


Fig. 2. The proposed SET flip-flop.

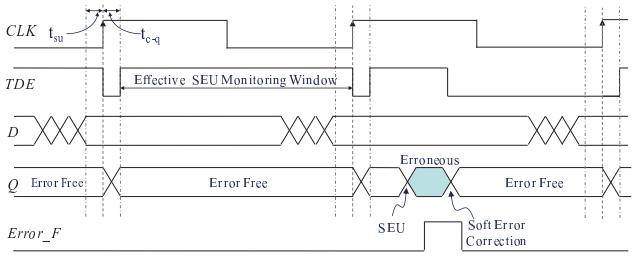


Fig. 3. SET flip-flop timing diagrams.

An alternative design approach, which is shown in Fig. 2, will be used for the evaluation of the proposed concept. It exploits the bit-flipping mechanism (flip-flop preset/clear operation) presented by the research team in [19] for timing error tolerance. In this figure, the SET flip-flop is embedded in a register of the circuit. It consists of the original flip-flop, a TD unit and two NAND gates. A timing diagram for the operation of this SET flip-flop is shown in Fig. 3. The TD unit monitors the output (Q) of the flip-flop within the effective SEU monitoring window, where *TDE* is "high", as shown in Fig. 3. Within the monitoring window, no signal transitions are expected at the output of the flip-flop. However, in case of a signal transition during this window, the TD unit detects it and generates a positive pulse at its output *Error_F*. Then a correction operation is activated. According to the proposed topology, if the final value at the output Q, after the error generation, is a logic "high" then the pulse of the TD unit activates the right NAND gate to clear the output Q to "low", else if the final value at the Q output is a logic "low" then this pulse activates the left NAND gate to preset the output Q to

"high". Consequently, the soft error at the output of the flip-flop is corrected.

The error indication signal *Error_F* is locally captured in the register using the Error flip-flop which provides the register level error indication signal *Error_R*. This signal is exploited to deactivate the TD unit (set TDE to "low") for the rest of the period. Careful design is required to avoid race conditions.

The desired monitoring window for the TD unit starts after the maximum *CLK* to *Q* delay of the flip-flop ($t_{c\text{-}q}$) and ends at the next rising clock edge. This window is determined by the transition detection enable signal *TDE*. When the *TDE* signal is "high", any transition at the *Q* output of the flip-flop will be detected and a pulse will be generated at the output of the TD unit. The *TDE* signal is locally derived from the clock signal by a circuit that is shared among the flip-flops which constitute a register (see Fig. 4).

The buffer, through which the TD unit is fed by the *Q* signal in Fig. 2, is used to minimize the parasitic capacitance load at the output of the flip-flop and compensate the delay of the *TDE* signal generation unit.

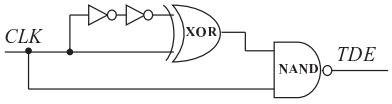


Fig. 4. *TDE* signal generation.

C. The transition detection unit

The proposed SET flip-flop could be implemented using existing TD units in the literature, such as the custom design scheme proposed in [11]. However, for the evaluation of this work the synthesizable TD unit illustrated in Fig. 5 was adopted. It consists of a two input XOR gate and delay elements (inverters or tri-state inverters). The TD unit is enabled for the monitoring operation by the *TDE* signal (active "high"). When the *TDE* signal is "low" any signal at the input of the TD unit arrives concurrently to both inputs of the XOR gate, through the two bottom signal paths that are activated. Thus, no pulse is generated at its output. When the *TDE* signal is "high", the top and the bottom signal paths are activated and consequently there is a delay between the arrivals of the signals at the two inputs of the XOR gate, due to the delay elements inserted in the top path. Thus, the XOR gate generates a pulse with duration equal to the delay inserted in the top path. The pulse width is suitable to permit a preset or clear operation at the flip-flop under monitoring.

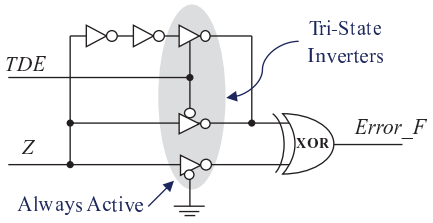


Fig. 5. A Transition Detection (TD) unit.

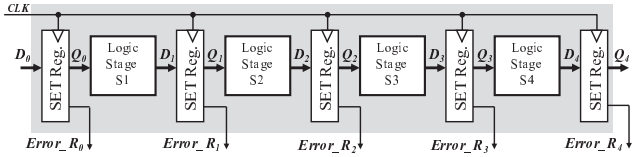


Fig. 6. The pipeline unit of a microprocessor core.

D. Microprocessor core error recovery

Let us consider, without loss of generality, a microprocessor core consisting of a flip-flop based pipeline, as shown in Fig. 6. The stage registers are composed of SET flip-flops. In the error free case, the TD units in the SET flip-flops do not generate any pulse and the core continues its operation in the normal mode. In case of a soft error event, the corresponding TD unit detects it and the proper signal is activated for the local error correction. However, due to the time needed for the detection and the correction of an error in a register, it is not always feasible to ensure that the logic stage that follows will have the required time for correct, (in-time) computation, especially when time critical paths start from the affected flip-flop. Consequently, in the general case, we need a mechanism to guarantee the correct operation of the pipeline. A possible approach to provide the extra time needed for pipeline recovery is to deactivate the clock signal *CLK* for a time duration equal to a single clock cycle (this is identified with the next clock cycle), exploiting existing (low-power oriented) core level clock gating mechanisms.

Initially, suppose that a soft error is detected and corrected at a SET flip-flop in a stage register ($Register_j$) (see Fig. 2 and 6). Thus, the response of the next logic stage S_{j+1} in the current clock cycle may be erroneous due to a possible inability of this stage to provide a correct computation in the remaining time of the clock cycle. In order to extend the computation time of stage S_{j+1} using a clock gating mechanism, an error detection signal is required to activate a clock control circuitry. Thus, in each register an OR tree is used to collect all error indication signals *Error_F* from the flip-flops and generate the register level error indication signal *Error_R* that is captured in the Error flip-flop. Furthermore, all $Error\_R_j$ signals feed the Clock Control unit shown in Fig. 7, where they are collected by a second OR gate to trigger the clock input of a simple flip-flop (Block flip-flop). The data input of this flip-flop is connected to V_{DD} (logic "high"). The Block flip-flop is initialized to "low". In case of an error detection, the generated $Error\_R_j$ signal triggers the Block flip-flop through the OR tree. Thus, the *Block* signal rises to "high" and blocks the system clock *SYS_CLK* from driving the core (global clock gating).

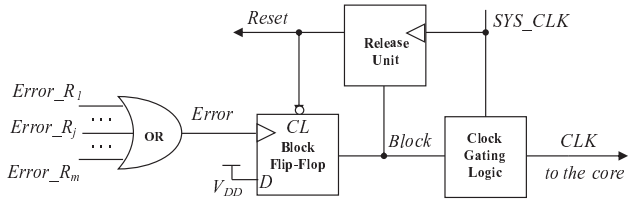


Fig. 7. The Clock Control unit for core level global clock gating.

The *Block* signal is also used to activate the Release unit which resets the Block flip-flop to release the clock signal after the expiration of the next system clock cycle. In addition it resets the Error flip-flops at the stage registers. Actually, the Release unit is a simple "counter" that counts for one system clock cycle after its activation. The operation of the pipeline recovery mechanism is illustrated in Fig. 8. Let us consider that a SEU at the register before the logic stage S3 is responsible for the generation of a soft error at the inputs of this stage. The error is detected and corrected by the corresponding SET flip-flop inside the detection/correction cycle. The pertinent register error indication signal is activated and the clock signal is blocked for a duration equal to the duration of the next clock cycle ("recovery cycle"). The logic stage S3 uses this extra time for correct (in-time) computation. Then the pipeline continues in the normal mode of operation.

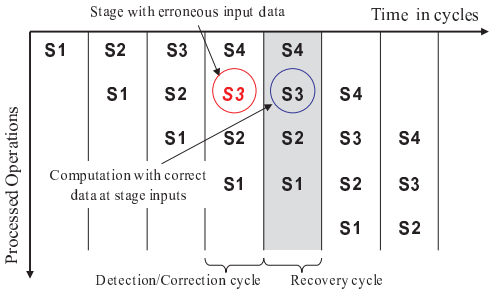


Fig. 8. Pipeline recovery operation.

Note that during the extra computation time provided for pipeline recovery, the stages other than S3, remain inactive retaining at their outputs the correct responses. Moreover, the proposed soft error detection and correction technique can tolerate multiple errors within a clock cycle.

E. Soft error tolerance efficiency

Next we will analyze the effectiveness of the proposed soft error tolerance technique. In Fig. 9 the clock period T of *CLK* is divided into four regions.

- Region A is identical to the worst case response time ($t_{c\text{-}q}$) of the flip-flop. During this time interval the TD unit is inactive so that it is not feasible to provide protection from SEUs.
- Region B is the time window (with duration t_{dc}) where a detected and corrected soft error at the output of the flip-flop does not affect the circuit operation. This means that either the corrected data at the flip-flop output have enough time to propagate inside the subsequent logic stage within the rest clock period (minus the flip-flop's set up time t_{su}) or the error indication signal is capable to block the clock and provide the required time for the propagation of the corrected data inside the subsequent logic stage. The rightmost edge of this area is determined by the latest error that is corrected in the worst case. Consequently, the duration of region B is specified by the delay of the TD unit (t_{TD}), which is dominated by the XOR delay, the propagation delay of the error indication signal through the OR tree (t_{OR}) and the Block flip-flop ($t_{c\text{-}q}$), as well as the propagation delay for the distribution of the clock signal inside the core (t_{pcc}) from the clock control unit to the flip-flops:

$$t_{dc} = T - t_{c\text{-}q} - (t_{TD} + t_{OR} + t_{c\text{-}q} + t_{pcc}) =$$
$$= T - 2t_{c\text{-}q} - t_{TD} - t_{OR} - t_{pcc}$$

- Region C is a time window (with duration t_{up}) where soft errors are detected; however, it may not be feasible to guarantee their correction as we will discuss next.
- Region D is identical to the best case minimum propagation delay (t_{min}) inside the subsequent logic stage minus the hold time of the flip-flop (t_h). During this time interval, it is not feasible for a generated soft error at the output of a flip-flop to propagate through the subsequent logic stage and be captured by the following register at the next triggering edge of the clock *CLK*.

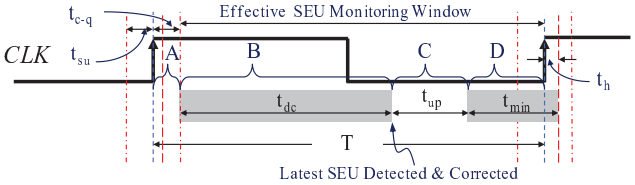


Fig. 9. Soft error tolerance analysis

From the above analysis we distinguish the following two cases in the circuit operation:

a) The minimum propagation delay (t_{min}) is large enough so that the duration of the C region (t_{up}) is zero. The above condition stands when $t_{min} > T + t_h - (t_{dc} + t_{c\text{-}q})$. In that case, the soft errors are either detected and corrected or they are not capable to affect the circuit operation due to temporal masking effects (except those errors that arise inside region A). The SEU protection ratio (SPR) inside the clock period is:

$$SPR = (T - t_{c\text{-}q}) / T$$

b) The minimum propagation delay (t_{min}) is small so that the duration of the C region (t_{up}) is not zero. Under this condition the SPR is:

$$SPR = (T - t_{c\text{-}q} - t_{up}) / T$$
$$= \{T - t_{c\text{-}q} - [T + t_h - t_{min} - (t_{dc} + t_{c\text{-}q})]\} / T =$$
$$= [t_{min} + t_{dc} - t_h] / T$$

However, an SEU inside region C will affect the circuit operation only if the generated soft error has enough time to propagate through the subsequent logic. This means that the soft error is propagated through paths with best case propagation delay (t_{pd}) less than:

$$t_{pd} < T + t_h + t_{SEU} - t_{min}$$

where t_{SEU} is the appearance time of the SEU after the triggering edge of the clock. In addition t_{SEU} must fall inside the C region that is:

$$T - (t_{dc} + t_{c\text{-}q}) < t_{SEU} < T + t_h - t_{min}$$

Under the above circumstances the corresponding soft error although it will be detected and corrected at the output of the flip-flop, it will be also propagated to the output of the subsequent logic stage and captured in the pertinent register. Possible solutions to the above problem are:

1) the propagation delay minimization of the error indication signal (through the OR tree and the Block flip-flop) from SET flip-flops related to fast paths in the logic; this means that for fast logic paths the corresponding error indication signals are propagated through shallow branches of the OR tree while for slow logic paths the corresponding error indication signals are propagated through deep branches of the OR tree, considering that a balanced tree is not a prerequisite for the proposed technique,
2) the padding of the corresponding fast logic paths to increase their propagation delay and

3) the architectural replay of the operations inside the pipeline.

The second approach increases the silicon area cost and the power consumption, while the third solution is characterized by a greater latency for the error correction plus the corresponding replay power consumption. Note that architectural replay is initiated only when both the *Block* signal is active and the clock signal *CLK* is "high" inside the A region of the clock period, which means that the error has been detected but there was not enough time for the error indication signal to block the clock.

F. Low cost soft error detection supported by architectural replay

An alternative low cost soft error detection scheme (without local correction capabilities) is presented next and illustrated in Fig. 10. According to this, a TD unit is shared between a pair of flip-flops (the concept can be extended to a higher number of flip-flops). The TD unit is fed by the output signals of the two flip-flops through a XOR gate. An SEU related transition at the output of a flip-flop, inside the monitoring window as this defined in Fig. 2, will be propagated through the XOR to the input of the TD unit where it will be detected. The local generated error indication signal *Error_FP* of each flip-flop pair is propagated through an OR tree to a control unit which is responsible for architectural replay activation in order to attain pipeline recovery.

Also in this case, the *TDE* signal is properly activated so that nominal transitions inside region A (see Fig. 9) are not detected. Thus, the SEU protection ratio (SPR) inside the clock period is:

$$SPR = (T - t_{c\text{-}q}) / T$$

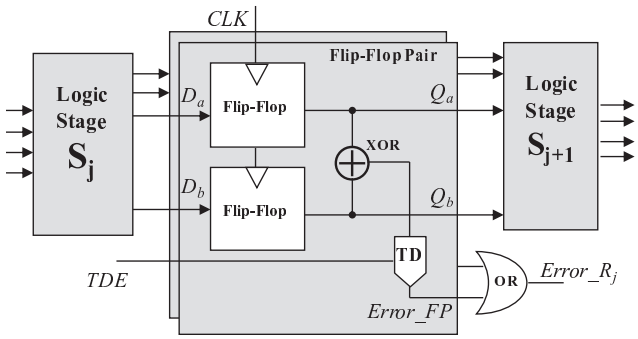


Fig. 10. Low cost soft error detection.

IV. Experimental Results

A. Comparisons

In this section, the area of the proposed SET protection scheme is compared with existing soft error tolerance techniques in the literature. The overhead of each technique is considered by comparing the total area of the protected sequential (including OR tree, transition detectors, etc.) to the area of the original design. In the comparisons that follow the standard cells of the 45nm OpenCellLibrary [20] are used.

The SET flip-flop presented in this work has a similar structure to that of the Razor-II topology [11]. However, Razor-II has been proposed for error detection in latch based designs and it does not provide the option for local (bit level) error correction since it relies on architectural replay.

Furthermore, compared to the low cost soft error detection scheme in Fig. 10, the Razor-II approach has a 37% higher area overhead, assuming that the TD unit in Fig. 5 is used in both designs. This is primarily because we propose to share one TD unit between two sequentials by using an XOR gate.

The BISER [14] is able to locally block the propagation of errors and it has a comparatively lower silicon area than other local correction techniques. However, with technology shrinking, when implementing BISER, it may be necessary to increase the physical separation of the redundant latches in the layout, in order to ensure both latches are not upset by a single particle strike [21].

The LCODE scheme proposed in [18] for latch based designs exploits a linear code for error detection and correction (ECC). In our area comparison, we have used the area of the bit-flipping latch directly as reported in [18]. It is important to note that the temporal window during which LCODE is able to detect and correct errors is limited by the propagation delay of the XOR network for the error detection code. If an upset occurs late in the period of the clock when the latch is opaque, it may be captured in downstream logic before be corrected (or detected) by the ECC circuitry.

TABLE I
NORMALIZED PER-BIT SILICON AREA COMPARISONS

Cell	Area µm²	SET-FF Correct	TMR	LCODE [18]	BISER [14]	SET-FF Detect	Razor-II [11]	DMR	LCODE [18]
		Correct				Detect			
DFF	4.52	0.03	3.00		2.00	1.00		2.00	
DFFRS	6.38	1.00							
LATCH	2.66			0.22			1.00		1.22
BFLATCH	3.19			1.20					
INV	0.53	3.06		0.15	6.00	1.56	3.00		
TINV	1.06	2.00				1.00	2.00		
NAND2	0.79	2.06		1.00		0.03			
AND2	1.06		3.00						
OR2	1.06	1.00		0.22		0.50	1.00	1.00	0.22
OR3	1.33		1.00						
XOR2	1.60	1.03		2.19		1.03	1.00	1.00	2.00
Total (µm²)		14.6	18.1	8.3	12.2	8.62	9.0	11.7	6.7
Overhead (%)		224	300	212	171	91	240	158	151

DMR=double modular redundancy

Table I presents comparisons on the average area per bit, assuming a register width of 32-bits. In all design schemes above, certain additional circuitry is required for each bit while other overheads are amortized across the full register. The percentage overhead is computed with respect to the area of the original sequential element (either flip-flop or latch).

B. Concept validation

The proposed technique was applied in the design of a 32bit MIPS R2000 RISC microprocessor, running at 115MHz frequency, in the 90nm CMOS technology of UMC (V_{DD}=1.2V) using the standard cells of Faraday Technology. The microprocessor consists of a five stages pipeline, the register file (RF), the instruction cache (1KB – 256 instruction words) and the data cache (2KB – 512 instruction words. The pipeline stages are: Instruction Fetch (IF), Instruction Decode (ID) which includes the control logic, Execute (EX), Memory (MEM) and Write-Back (WB). The processor characteristics are presented in Table II. All 466 flip-flops in the design are replaced by SET flip-flops.

TABLE II
MICROPROCESSOR CORE CHARACTERISTICS

Technology	90nm CMOS
Power Supply	1.2V
Total Silicon Area	259066 μm^2
# Flip-Flops	466
Core Frequency	115 MHz @ 1.2V
Core Power	6.65 mW @ 1.2V
I-Cache Size	1KB
D-Cache Size	2KB

The CADENCE platform has been used for the design and simulation of the core (RTL-Compiler for synthesis, Encounter for place and routing and NC-launch for simulation). Fig. 11 presents post-layout timing simulations on the microprocessor design that confirm the operation of the proposed technique. In this figure an injected error is detected and corrected by a preset operation on the corresponding fli-flop. In addition, the *Block* signal is activated at the output of the Block flip-flop to permit clock gating. Thus, the computation time is extended by the duration of a single clock cycle for pipeline recovery. Afterwards, the pipeline proceeds with its normal operation. An error is injected by certain set/reset operations on the target flip-flop.

V. CONCLUSIONS

Soft error tolerance design techniques for flip-flop based microprocessor cores are presented in this work. A transition detector is utilized for soft error detection along with an asynchronous scheme for automatic local (per bit) error correction. Alternatively, a silicon area efficient soft error detection topology is proposed, which shares a transition detection unit across multiple flip-flops. In that case, error recovery is accomplished by exploiting architectural replay techniques.

REFERENCES

[1] M. Nicolaidis, "GRAAL: A New Fault Tolerant Design Paradigm for Mitigating the Flaws of Deep Nanometric Technologies," *IEEE International Test Conference*, p. 4.3, 2007.

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

[3] 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.

[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," *IEEE Int. On-Line Testing Workshop*, pp. 66-70, 1999.

[6] 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.

[7] 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.

[8] 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.

[9] D. Ernst, N-S.Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *International Symposium on Microarchitecture*, pp. 7-18, 2003.

[10] K. Bowman, J.W. Tschanz, S-L. Lu, P.A. Aseron, M.M Khellah, A. Raychowdhury, B.M. Geuskens, C. Tokunaga, C.B. Wilkerson, T. Karnik and V.K De, "A 45nm Resilient Microprocessor Core for Dynamic Variation Tolerance," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 194-208, 2011.

[11] S. Das, C. Tokunaga, S. Pant, W-H. Ma, S. Kalaiselvan, K. Lai, D.M. Bull and D.T. Blaauw, "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32-48, 2009.

[12] 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.

[13] S. Valadimas, Y. Tsiatouhas and A. Arapoyanni, "Timing Error Tolerance in Nanometer ICs," *IEEE International On-Line Testing Symposium*, pp. 283-288, 2010.

[14] M. Zhang, et al, "Sequential Element Design with Built-In Soft Error Resilience," *IEEE Transactions on VLSI Systems*, vol. 14, no. 12, pp. 1368-1378, 2006.

[15] H. Yu, M. Nicolaidis, L. Anghel and N-E. Zergainoh, "Efficient Fault Detection Architecture Design of Latch-based Low Power DSP/MCU Processor," *IEEE European Test Symposium*, pp. 177-183, 2011.

[16] K. Bowman, J.W. Tschanz, N-S. Kim, J.C. Lee, C.B. Wilkerson, S-L. Lu, T. Karnik and V.K De, "Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 49-63, 2009.

[17] D. Bull, S. Das, K. Shivashankar, G.S. Dasika, K. Flautner and D.T. Blaauw, "A Power-Efficient 32 bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 18-31, 2011.

[18] M.E. Imhof and H-J. Wunderlich, "Soft Error Correction in Embedded Storage Elements," *IEEE International On-Line Testing Symposium*, pp. 169-174, 2011.

[19] S. Valadimas, Y. Tsiatouhas and A. Arapoyanni, "Cost and Power Efficient Timing Error Tolerance in Flip-Flop Based Microprocessor Cores," *IEEE European Test Symposium*, pp. 8-13, 2012.

[20] "Nangate Open Cell Library," Available online at: http://www.si2.org.

[21] N. Seifert, V. Ambrose, B. Gill, Q. Shi, R. Allmon, C. Recchia, S. Mukherjee, N. Nassif, J. Krause, J. Pickholtz, and A. Balasubramanian, "On the Radiation-Induced Soft Error Performance of Hardened Sequential Elements in Advanced Bulk CMOS Technologies," *IEEE International Reliability Physics Symposium*, pp. 188 –197, 2010.

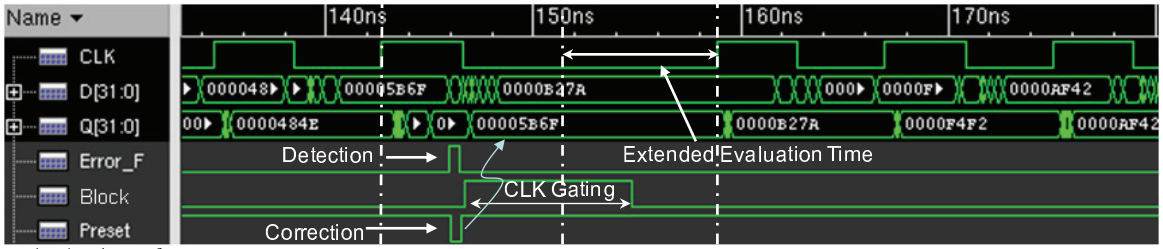


Fig. 11. Simulated waveforms