A New Hybrid Fault-Tolerant Architecture for Digital CMOS Circuits and Systems

Tran, Duc A.; Virazel, Arnaud; Bosio, Alberto; Dilillo, Luigi; Girard, Patrick; Pravossoudovich, Serge; Wunderlich, Hans-Joachim

Journal of Electronic Testing: Theory and Applications (JETTA) Vol. 30(4) 8 June 2014

url: http://link.springer.com/article/10.1007/s10836-014-5459-3

doi: http://dx.doi.org/10.1007/s10836-014-5459-3%20

Abstract: This paper presents a new hybrid fault-tolerant architecture for robustness improvement of digital CMOS circuits and systems. It targets all kinds of errors in combinational part of logic circuits and thus, can be combined with advanced SEU protection techniques for sequential elements while reducing the power consumption. The proposed architecture combines different types of redundancies: information redundancy for error detection, temporal redundancy for soft error correction and hardware redundancy for hard error correction. Moreover, it uses a pseudo-dynamic comparator for SET and timing errors detection. Besides, the proposed method also aims to reduce power consumption of fault-tolerant architectures while keeping a comparable area overhead compared to existing solutions. Results on the largest ISCAS'85 and ITC'99 benchmark circuits show that our approach has an area cost of about 3% to 6% with a power consumption saving of about 33% compared to TMR architectures.

A New Hybrid Fault-Tolerant Architecture for Digital CMOS Circuits and Systems

D. A. Tran[1] A. Virazel[1] A. Bosio[1] L. Dilillo[1] P. Girard[1] S. Pravossoudovich[1] H. –J. Wunderlich[2]

[1] LIRMM – University of Montpellier / CNRS
Montpellier, France
tran@lirmm.fr
virazel@lirmm.fr
bosio@lirmm.fr
dilillo@lirmm.fr
girard@lirmm.fr
pravo@lirmm.fr

[2] Institute of Computer Architecture and Computer Engineering
University of Stuttgart, Germany
wu@iti.uni-stuttgart.de

This paper presents a new hybrid fault-tolerant architecture for robustness improvement of digital CMOS circuits and systems. It targets all kinds of errors in combinational part of logic circuits and thus, can be combined with advanced SEU protection techniques for sequential elements while reducing the power consumption. The proposed architecture combines different types of redundancies: information redundancy for error detection, temporal redundancy for soft error correction and hardware redundancy for hard error correction. Moreover, it uses a pseudo-dynamic comparator for SET and timing errors detection. Besides, the proposed method also aims to reduce power consumption of fault-tolerant architectures while keeping a comparable area overhead compared to existing solutions. Results on the largest ISCAS'85 and ITC'99 benchmark circuits show that our approach has an area cost of about 3% to 6% with a power consumption saving of about 33% compared to TMR architectures.

I. INTRODUCTION

Digital systems have transitioned from specialized applications to ubiquitous mass products. Today, a consumer mobile device may contain a processor with higher computing power than a super computer in the early 1990s. This revolution has been conducted by the evolution of Complementary Metal Oxide Semiconductor (CMOS) technology, which allows the production of smaller and cheaper Integrated Circuits (IC) with higher performance and lower power consumption. While supporting the need for competitive mass products, this evolution also influences the reliability of digital systems [1]. Increasing occurrence rate of faults and errors during manufacturing processes and circuit lifetime makes robustness one of the upcoming key requirements in many application areas, including safety critical applications and mass products.

Different factors are responsible for transient and permanent faults that affect robustness of digital circuits and systems. These faults may induce errors and cause digital systems to fail. First of all, smaller devices are more difficult to fabricate. This leads to a higher rate of manufacturing defects and a lower manufacturing yield. Furthermore, if these defects are not detected during production test, they may cause hard errors during ICs' operation. Secondly, defect free ICs may suffer from Process-Voltage-Temperature (PVT) variations which are responsible to timing errors. During their lifetime, ICs are also affected by interference phenomena. Smaller transistors are more vulnerable to radiation effects, which cause soft errors in both memories and logic circuits (single event-upsets SEUs affecting sequential elements and single event-transients SETs affecting combinational logic). Finally, aging phenomena are responsible to intermittent faults, which may also cause hard and timing error at the end of circuits' lifetime.

Robustness improvement of digital circuits and systems is getting more and more difficult for every introduced CMOS technology node because treating faults at physical level by adjusting manufacturing process parameters is no longer feasible. Therefore, fault-tolerance techniques, which deal with faults at design level, have become essential to fulfill the required robustness of future digital CMOS circuits and systems. These techniques employ information, timing and hardware redundancies to guarantee correct operations despite the presence of faults [2].

In the memory part of digital systems, the use of fault-tolerant techniques has been proven necessary and efficient. Information (error detection and correction codes, [2]) and hardware (spare memory words, columns and cells [3, 4]) redundancies are generally employed to deal with both transient and permanent faults in memories. However, fault-tolerance in random logic circuits of digital systems remains a challenge. These circuits are combined of combinational logic and sequential elements such as latches and flip-flops. Different techniques have been proposed to protect the sequential part from hard errors and/or SEUs [5, 6, 7, 8, 9]. Some of these techniques are also efficient for SETs and timing errors that arrived at the combinational part of logic circuits [7, 8]. Besides, in order to protect this part from hard errors, Triple Modular Redundancy (TMR) architecture has been proven to be an efficient method [10, 11]. Although each type of error has several corresponding solutions, combining all these techniques for robustness improvement may require very high level of redundancy and thus, may not be applicable in mass products.

Besides fault-tolerance capability, area overhead is the main optimization criterion. In [10, 11], authors have introduced manufacturing yield enhancement as a new goal. Moreover, power consumption is also a rising issue in advanced CMOS technology nodes. As fault-tolerance becomes necessary in mass products, limiting power consumption increase of these techniques is one of the key factors in digital design.

Given the new requirements for fault-tolerance, the principle of a hybrid fault-tolerant architecture that targets robustness, manufacturing yield and power consumption of logic circuits is presented in [12]. Combining information, timing and hardware redundancy, this solution targets both hard errors and SETs tolerance in the combinational part of logic circuits. Similar to a TMR solution, this architecture consists of implementing three times the combinational part of the original logic circuit. However, only two of them are running in parallel. A Finite State Machine (FSM) is used to select the running two combinational logic modules. It changes the architecture configuration with respect to the error detection made by a static comparator. It has been proven that this architecture allows30% power consumption reduction compared to TMR architecture while keeping similar area overhead. However, it is vulnerable to SETs, which causes short duration glitches at circuit outputs, as well as timing errors. Furthermore, depending on original logic circuits, additional hardware may cause significant delay overhead and thus, degrade system performance.

To resolve different drawbacks stated above, in this paper, we propose a novel hybrid fault-tolerant architecture based on the principle presented in [12] to tolerate transient and permanent faults that may affect combinational logic networks [ref1, ref2]. In order to tolerate timing errors as well as to improve SETs detection/correction, a pseudo-dynamic comparator is employed [13]. Besides, by removing this comparator from the data path of the architecture, we obtain a lower delay overhead compared to previous solution. Moreover, the new hybrid fault-tolerant architecture can be combined with advanced SEUs fault-tolerance methods that require lower area overhead such as [9]. Finally, simulation results on ISCAS'85 and ITC'99 benchmark circuits show that the new approach still has negligible area overhead compared to TMR structure while helps reducing more than 30% of power consumption.

The rest of this paper is organized as follows. Section II briefly reviews existing fault-tolerance methods for both combinational and sequential parts of logic circuits. It also details different drawback of the previous hybrid fault-tolerant architecture proposed in [12]. Section III provides the principle as well as implementation detailed of the new hybrid fault-tolerant architecture. Simulation results that show fault-tolerance capability together with area and power consumption evaluation of the new architecture are presented in Section IV. Finally, Section V concludes the paper and provides some perspectives.

II. State of The Art

A. Fault-tolerance in logic circuits

Many works have studied the protection of sequential elements in logic circuits from hard errors and SEU. The most prominent techniques include TMR, BISER, Razor, GRAAL, Razor II and information redundancy [5, 6, 7, 8, 9, 14]. TMR and BISER techniques employ hardware redundancy to mask hard errors and/or SEU at latch and flip-flop outputs. Razor, GRAAL and Razor II methods [7, 8, 14] introduce timing redundancy into the protection scheme that allows not only area overhead reduction but also protects the combinational logic part from SET and timing errors caused by Dynamic Voltage Scaling (DVS) process. While other techniques protect sequential elements at bit-level, in [9], authors propose the use of linear codes to enhance these elements at register level. Significant silicon area can be saved using this technique but SETs and timing errors must be treated separately.

While there are several fault-tolerant techniques for sequential elements, protecting combinational part of logic circuits from both transient and permanent errors remain a challenge. Hardware redundancy such as TMR is one of the best-known techniques to deal with hard error and to improve manufacturing yield [10, 11]. Unfortunately, this architecture is vulnerable to SETs and timing errors, which is the rising issue in advanced CMOS technology nodes [15, 16]. One solution is to combine TMR with others techniques such as Razor or Razor II. However, this method may require area and power consumption overhead.

Various solutions using fault-tolerant techniques for robustness improvement of microprocessor core have also been proposed. The method presented in [ref3] target only transient fault protection in combinational logic parts with the help of STEM cell technique. Gupta *et al.* in [ref4] proposed a software based reconfiguration scheme for permanent fault protection. Techniques that offer both transient and permanent fault protection have also been proposed in the literature but they generally rely on slow recovery mechanisms thus are not suitable for highly interactive applications; like the method presented in [ref5] uses instruction re-execution and pipeline flushing to mitigate transient fault effects and in [ref6] a built-in self-test (BIST) is run to detect the presence of permanent fault in case of error detection.

B. Hybrid fault-tolerant architecture

In order to protect the combinational part of logic circuits, we have previously presented the principle of an architecture that detects both permanent and transient errors and corrects them via re-computation [12]. This architecture, called "initial hybrid-fault tolerant architecture" in the rest of this paper, is presented in Fig.1.

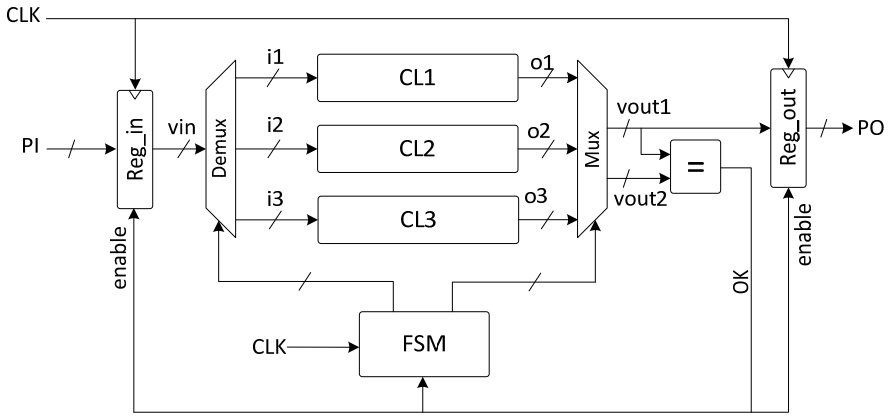


Fig.1. The initial hybrid fault-tolerant architecture [12]

In Fig.1, the combinational part of original logic circuit is implemented three times (CL1, CL2 and CL3). Only two combinational logics (CLs) are running in parallel at any moments while the third one is put on standby state. The configuration is selected by an input demultiplexer (Demux) and an output multiplexer (Mux), which are controlled by a Finite State Machine (FSM). Outputs of the two running circuits (*vout1*, *vout2*) are compared by a static comparator that provides an *OK* signal. This signal controls enable inputs of input and output registers (Reg_in, Reg_out). If *vout1* and *vout2* are different, due to a hard error or a SET, then both registers are disabled. Consequently, at the next clock (*CLK*) period, the previous input vector stored in Reg_in is re-computed. Depending on the current state of the FSM, the configuration is in order to deal with permanent and/or transient faults.

Compared to TMR method, the initial hybrid fault-tolerant architecture has several advantages. First of all, it may lead to about 30% dynamic power saving because only two out of three CLs are running in parallel. Besides, the area overhead of roughly 3% is negligible. In term of fault-tolerant capability, this architecture is also able to detect and correct both hard errors and SETs at combinational logic outputs.

However, the initial hybrid fault-tolerant architecture may suffer from different drawbacks. Firstly, it requires a higher computational time compared to the original logic circuit. While placing the comparator before Reg_out allows re-computation process to take only one *CLK* cycle, it creates an additional delay in the data path between Reg_in and Reg_out. Secondly, the static comparator may filter out small glitches caused by SET at *vout1* [13]. Therefore, it may reduce SET detection capability of the architecture. Finally, the comparator delay may cause *OK* signal to alert error detection after *CLK* capture edge. This makes the initial hybrid fault-tolerant architecture vulnerable to timing errors as well as to SETs that arrive just before CLK edge.Fig.2 shows an example where a timing error may be captured by the output register.

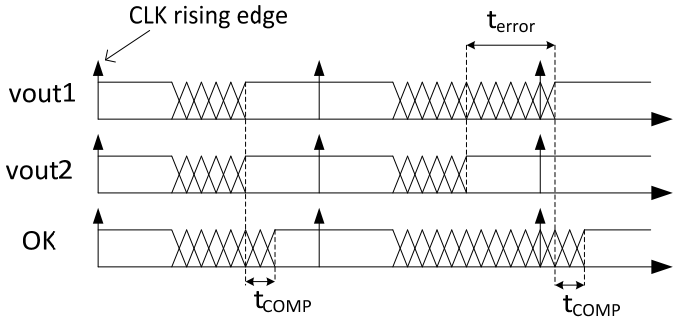


Fig.2. Timing error issue of the previous hybrid fault-tolerant architecture

Fig.2 shows waveforms of *vout1*, *vout2* and *OK* signals during two CLK cycles. In the first *CLK* cycle, the architecture works correctly. There is a comparator delay t_{Comp} between the moment where *vout1* and *vout2* become stable and the moment where *OK* turns to constant logic-1 (no error detected). Note that during t_{Comp} period, *OK* may oscillate between logic-0 and logic-1. In this period, *OK* stabilizes before *CLK* edge allowing Reg_out to capture the fault-free output *vout1*. In the second period, a timing error occurs and causes a delay t_{error} at *vout1*. This makes both *vout1* and *OK* unstable during CLK edge. Consequently, the error may be captured because Reg_out is not correctly disabled.

One solution to improve the fault-tolerance capability of the initial hybrid fault-tolerant architecture with regards to SETs and timing errors is to use enhanced flip-flops such as Razor and Razor II for the output register. However, these methods may require significant area overhead. In this paper, we propose a new hybrid fault-tolerant architecture that adds SETs and timing errors tolerance capability without using bit-level fault-tolerance techniques. Consequently, it can be combined with register-level fault-tolerance techniques such as proposed in [9] in order to protect sequential elements from SEUs with optimum area overhead.

III. The New Hybrid Fault-Tolerant Architecture

Our goal is to improve the initial hybrid fault-tolerant architecture so that all kinds of faults in the combinational part of logic circuits (*i.e.* hard, SET and timing errors) can be tolerated. Besides, area overhead and power consumption advantages must be persevered in the new solution.

A. Overview

Fig.3 shows the structural overview of the new hybrid fault-tolerant architecture.

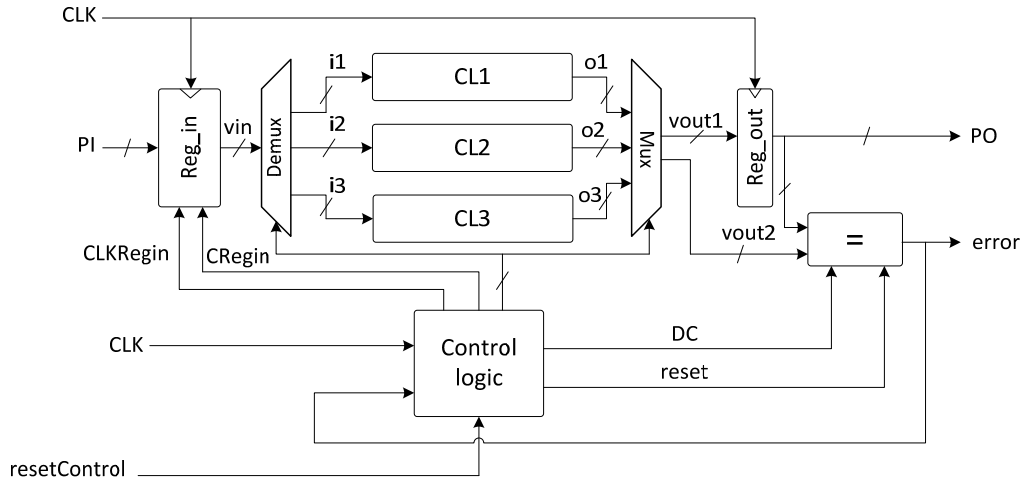


Fig.3. The new hybrid fault-tolerant architecture

Similar to the initial hybrid fault-tolerant method (Fig.2), the new architecture also employs three copies of combinational logic module (CL1, CL2 and CL3). The same input demultiplexer (Demux) and output multiplexer (Mux) are used to select two

running CLs and to put the third CL on standby mode during normal operations. Note that the proposed approach is fully scalable since it consists in triplicating the logic part of the circuit/system we want to protect against transient and permanent faults.

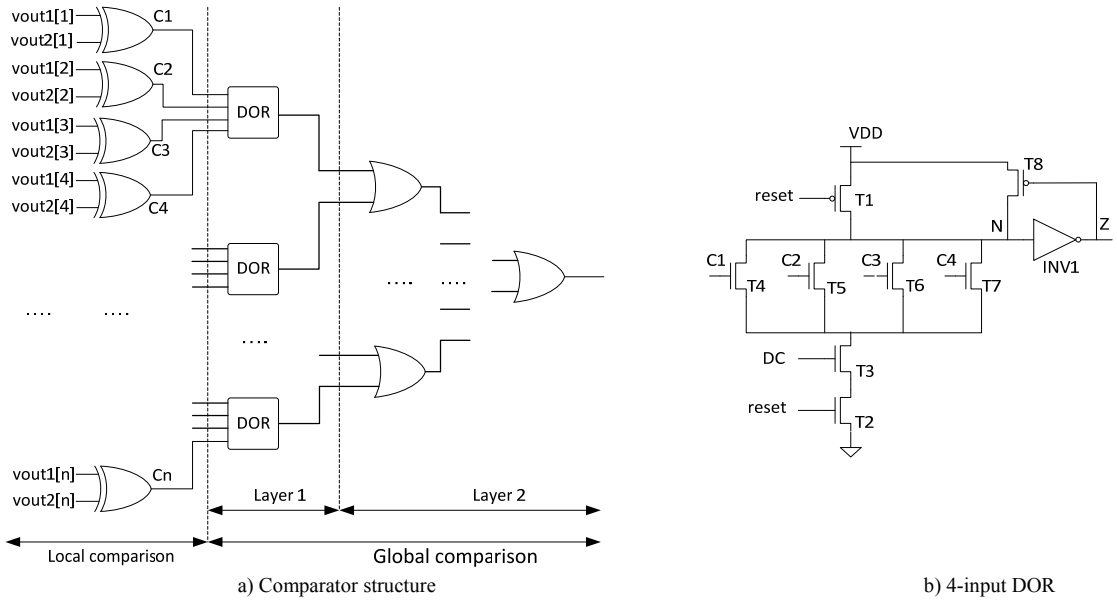
In the new architecture, error detection is also done by comparing outputs of the two running CLs. However, a pseudo-dynamic comparator replaces the static comparator in order to improve SET detection [13]. Furthermore, the pseudo-dynamic comparator is removed from data path between input (Reg_in) and output (Reg_out) registers. This configuration, which allows not only SET and timing errors detection but also smaller computation delays, is discussed in sub-section B.

As the pseudo-comparator is placed after output register Reg_out, error occurrences can only be detected after the CLK capture edge. However, a new input vector is also released by input register Reg_in at the same CLK edge. Consequently, the re-computation process cannot be triggered by using *error* signal to disable Reg_in and Reg_out for one clock cycle. Moreover, the previously affected input vector is not available for re-computation after error detection. One possible solution for these problems consists of storing each input vector until its corresponding output is proven correct by the comparator. In case of error detection, this memorized input vector is released to perform re-computation. This error correction scheme, which takes two CLK cycles, is controlled by a modified input register detailed in sub-section C.

The new hybrid fault-tolerant is driven by a control logic module, which is divided in two parts. The first part consists of a FSM that controls different configurations of the architecture, *i.e.* decides which two CLs are running in parallel. The second part provides control signals *DC* and *reset* for the pseudo-dynamic comparator as well as *CRegin* and *CLKRegin* for Reg_in. The control logic module can be reset by an external input *resetControl*. Timing constraints and implementation of this module are the subject of sub-section.

B. Error detection

In order to improve SET detection, as well as to enable timing error detection capability, we propose the use of pseudo-dynamic comparators studied in [13]. The structure of such comparators is illustrated in Fig.4.



a) Comparator structure b) 4-input DOR

Fig.4. Pseudo-dynamic comparator [13]

The pseudo-dynamic comparator has a similar structure compared to a static comparator. It is also combined of two stages: the Local comparison stage, which is made of static 2-input XOR gates and the Global comparison stage, which consists of an OR-tree. The only difference is that the first OR layer in the Global comparison stage is made of dynamic OR (DOR) gates. Owing to the presented architecture, beside comparison function, the pseudo-dynamic comparator also allows glitch detections at its input vectors during a defined period. This period, called the comparison window, is defined by the high phase of a *DC* signal. Note that once a mismatch between two input vectors is detected during the comparison window, the comparator output will remain at logic-1 until it is reinitialized by a *reset* signal.

As shown in Fig.3, the pseudo dynamic comparator is used to detect mismatches between *vout2* and captured output *PO*. Therefore, the comparison window is placed after the *CLK* capture edge, during hold time of CLs. If due to hard, SET or timing errors, *vout1* is not correctly captured then there will be mismatches between *PO* and *vout2*. Consequently, all errors at *vout1* are

detected. This situation is illustrated in Fig.5-a where the captured value *PO* is O2* instead of O2. In other hand, errors at *vout2* can also be detected as wrong logic value O2* (hard error, Fig.5-b) or invalidated transitions (SET and timing error, Fig.5-c and d) during the comparison window.

There are two constraints for implementation of the error detection scheme presented above. Firstly, as error detection is done after *CLK* rising edge, re-computation can only be triggered at next CLK edge. Therefore, error correction process requires two instead of one CLK cycle in the initial hybrid fault-tolerant architecture. Secondly, in normal operations *vout2* must be held constant during the comparison window. Consequently, short path delays of CLs must be constrained to cover this timing period. This constraint is similar to that applied for CLs used in Razor technique [7]. One solution consists in adding buffers in short paths of CLs to increase their delay.

C. *Re-computation*

As previously discussed, one solution to enable error correction by re-computation for the new hybrid fault-tolerant architecture consists in using additional memories to preserve each input data until the corresponding computation result has been proven correct. For this purpose, we propose to use a modified D flip-flop for the input register Reg_in. The flip-flop structure is presented in Fig.6-a.

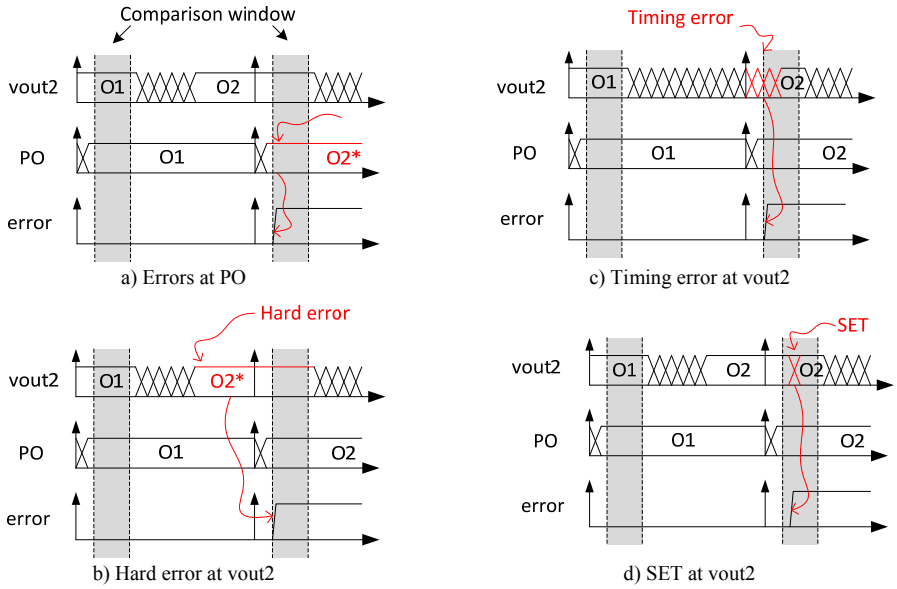


Fig.5. Pseudo-dynamic comparator functioning

InFig.6-a, a low level sensitive D latch (DLL) is used to store previous input in *Qm* when the original D flip-flop (DFF) has captured a new value *Q*. The time period during which *Qm* is maintained in DLL is defined by a new clock signal *CLKRegin* and a control signal *CRegin*. The last signal also controls a 2:1 multiplexer, which decides whether the new data *D* or the memorized value *Qm* will be passed to Q at the next *CLK* rising edge.

Additional signals *CLKRegin* and *CRegin* are driven by the control logic module of the new hybrid fault-tolerant architecture. Examples of this control in fault-free case and during error occurrence are illustrated in Fig.6-b and c.

In the fault-free case (Fig.6-b), *CRegin* remains at logic-0. Hence, the multiplexer always selects data input *D*. Consequently, at *CLK* rising edge ($t=t_{period}$), the main flip-flop DFF captures new data from *D* and hence, *Q* switches from D1 to D2. Before this moment, the prior data D1 has been captured by DLL during its transparent window (low phase of *CLKRegin)* from $t_{CLKRegin-}$ to $t_{CLKRegin+}$. This data is stored in DLL as long as *CLKRegin* remains logic-1.

In the situation of Fig.6-c, a fault becomes active while CLs are computing D0. The erroneous result is then detected during the next CLK period, between t=0 and $t=t_{period}$. CRegin will turn to logic-1 during this period so that the *enable* input of DLL is also kept at high level. Consequently, D1 is not captured during the next CLKRegin low phase from $t_{CLKRegin-}$ to $t_{CLKRegin+}$. Therefore, data D0 will be maintained by DLL. At the next CLK rising edge ($t=t_{period}$), the multiplexer will select this data for re-computation.

Using the modified D flip-flop, the new hybrid fault-tolerant architecture requires two CLK cycles for re-computation. The first cycle consists of error detection and affected input restoration from shadow latch DLLs while the second cycle is for re-computation of this vector. Note that inFig.6-c, when the error is detected, input *D* does not change from D1 to D2 and hence, D1

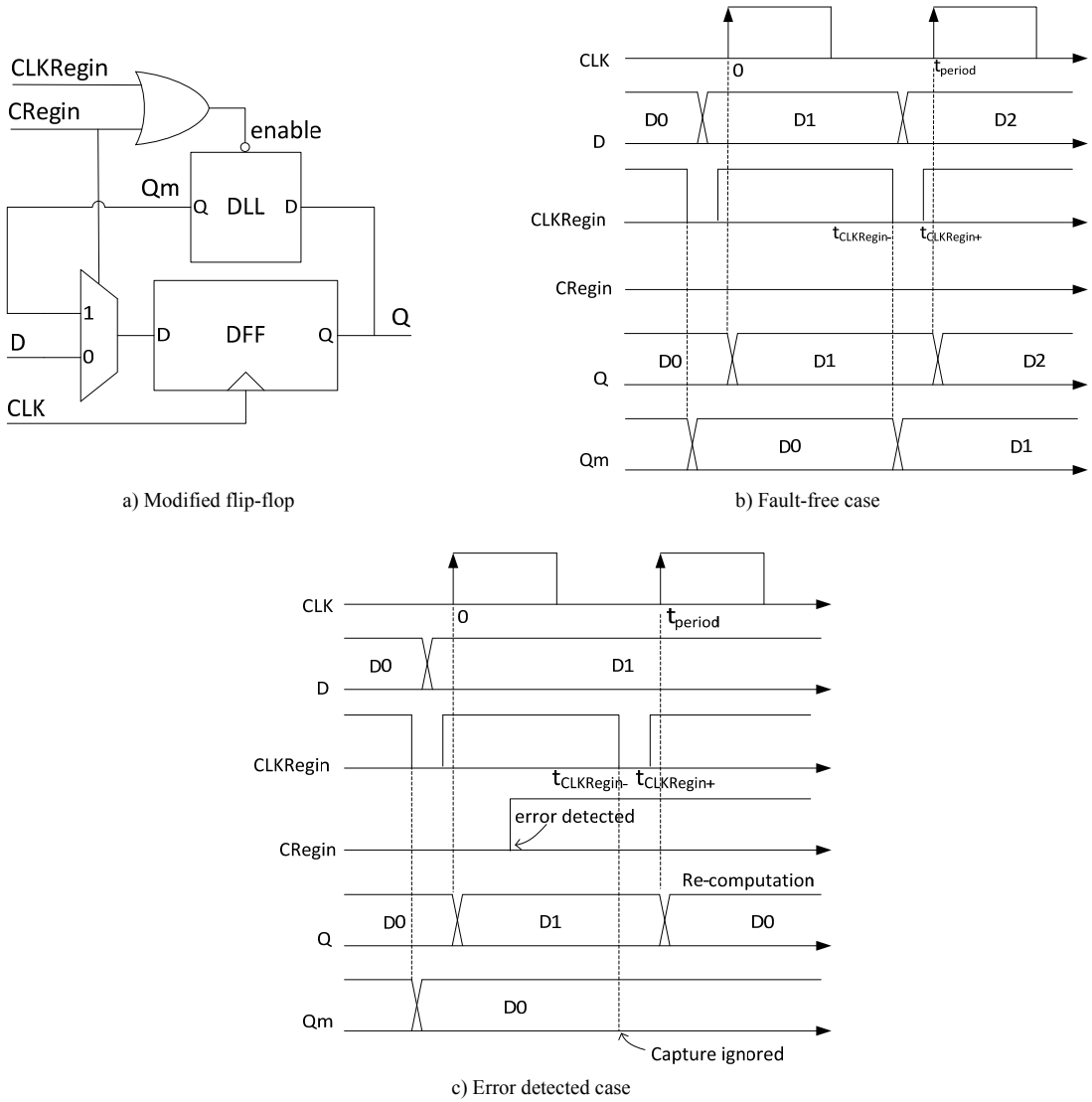
remains available after the error correction process. This must be controlled at system level and will not be discussed in this paper.

D. Control logic module

In the new hybrid fault-tolerant architecture, the control logic module is composed of two sub-modules. The first sub-module is used to re-configure the architecture via Mux and Demux modules (Fig.3). For this module, we can re-use FSM module of the initial hybrid fault-tolerant architecture [12]. However, as the error detection scheme has been modified, FSM will not be driven by *CLK* and *error* but *CLKRegin* and *CRegin* signals. At each *CLKRegin* falling edge, FSM will change state depending on the value of *CRegin*. The re-configuration corresponds to a fault-free situation if *CRegin* is at logic-0, and to an "error detected" situation otherwise.

CRegin and *CLKRegin* signals are generated by the second sub-module of the control logic module. This sub-module also provides control signals *DC* and *reset* for the pseudo-dynamic comparator.

Different timing constraints must be applied for the control signals to guarantee correct operations of the new hybrid fault-tolerant architecture. These constraints are illustrated in Fig.7.



a) Modified flip-flop

b) Fault-free case

c) Error detected case

Fig.6. Modified D flip-flop for re-computation

First of all, the comparison window during which *PO* and *vout2* are compared must be set when both signals are stable. Consequently, at the beginning of each *CLK* period, *DC* signal must turn to logic-1 after delay clock-to-Q t_{dff} of Reg_out. Then,

DC must return back to logic-0 before *vout2* starts switching value by the end of *CLs* computation time t_{CL}. These conditions are presented in Fig.7-a where red shades correspond to delays that must be respected between signal transitions.

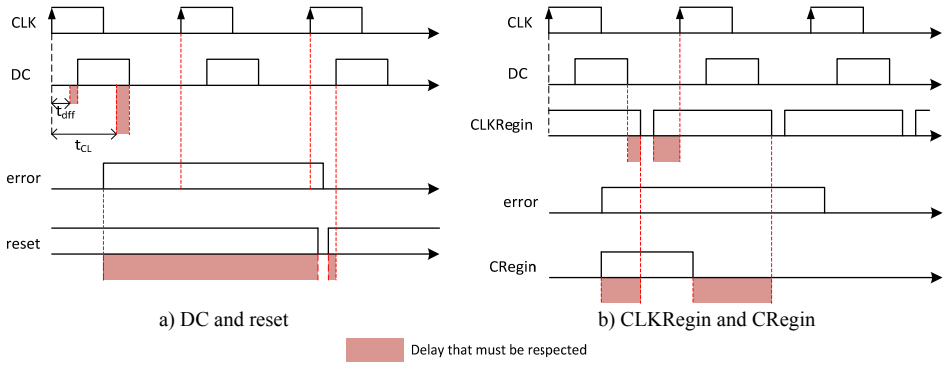


Fig.7. Timing constraints for control signals

Secondly, during error correction, *error* signal must be held at logic-1. As re-computation requires two *CLK* cycles, this means that *reset* is only applied (logic-0) after two *CLK* rising edges following an *error* rising edge. Besides, the comparator must be ready for the comparison windows. Therefore, *reset* low phase must finish before *DC* rising edge. The presented constraints are also shown in Fig.7-a.

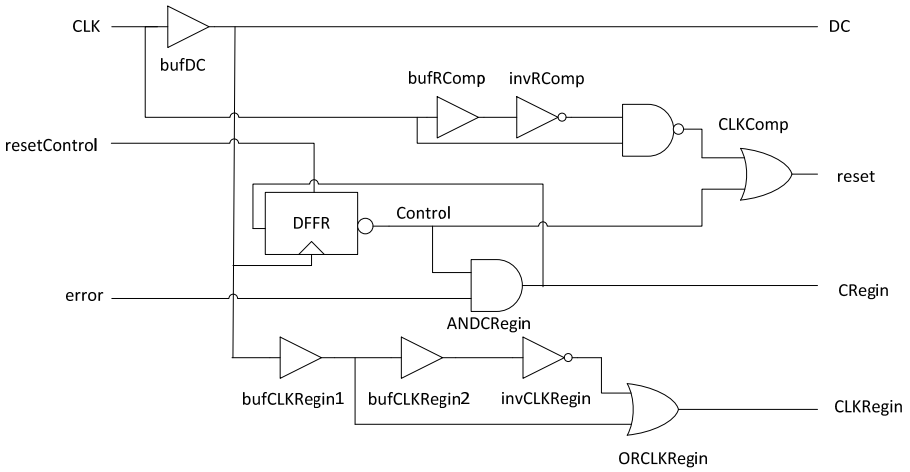


Fig.8. The second sub-module of the control logic module

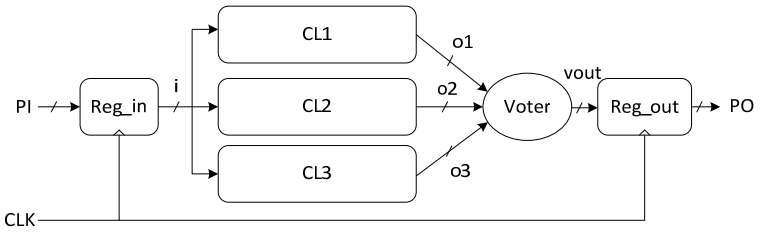
Fig.7-b presents the third timing constraint set for *CLKRegin* and *CRegin* signals of input register Reg_in (Fig.6). In order to enable re-computation possibility, each input vector must be stored in shadow latch DLLs of Reg_in until the corresponding output has been proven correct by the pseudo-dynamic comparator. Consequently, *CLKRegin* low phase must be set after the comparison windows. Moreover, this phase must finish before next data arrival in main flip-flop DFFs of Reg_in at *CLK* rising edge. After error detection, new input vector must not be captured by shadow latch DLLs. Consequently, *CRegin* must rise to logic-1 before *CLKRegin* low phase. Then, *CRegin* must return to logic-0 so that DLLs is ready to capture new input after re-configuration.

Fig.8 illustrates an implementation of the second sub-module that satisfies different constraints stated above.

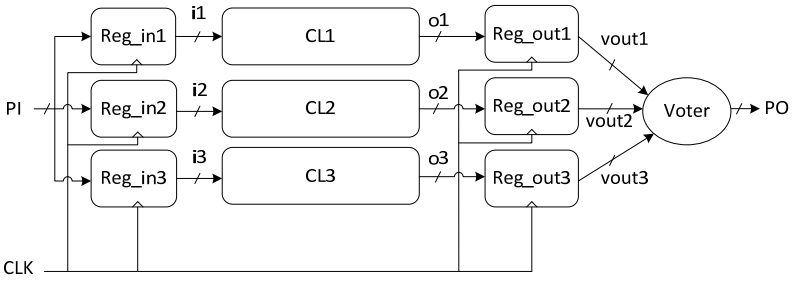
IV. Evaluation

The objective of this section is to evaluate the new hybrid fault-tolerant architecture presented in Fig.3 using simulations with commercial Electronic Design Automation (EDA) tools. To demonstrate its advantages, the new solution is compared to TMR architectures. Two TMR versions are studies: Partial TMR (TMR1, Fig.9-a) and Full TMR (TMR2, Fig.9-b). In TMR1 scheme, only combinational part of original logic circuits is triplicated (CL1, CL2, CL3) while in TMR2 structure, input (Reg_in1,

Reg_in2, Reg_in3) and output (Reg_out1, Reg_out2, Reg_out3) registers are also triplicated. Whilst TMR2 has the disadvantage of higher silicon area and power consumption, it allows single SET and timing errors tolerance, which is not possible with TMR1. Note that in both TMR schemes, word-wise voters are employed to guarantee better data integrity [17].



a) TMR1 (Partial TMR)



b) TMR2 (Full TMR)

Fig.9. TMR architectures

For evaluation purpose, the presented fault-tolerant architectures are applied for combinational part of largest ISCAS'85 and ITC'99 benchmark circuits. These architectures are implemented using static CMOS gates from the Nangate Open Cell Library (OCL, [18]), which contains 45nm technology standard cells specified by the Predictive Technology Model [19]. For dynamic CMOS DOR gates of the new hybrid fault-tolerant architecture, we use the DOR4 cell designed in [13] according to design and electrical rules of the FreePDK process design kit [20].

In following sub-sections, we present simulation results concerning area overhead, fault-tolerant capability and power consumption.

A. Area overhead

This sub-section compares the silicon area of the new hybrid fault-tolerant architecture with TMR1 and TMR2 solutions using ISCAS'85 and ITC'99 benchmark circuits. Comparisons are performed with TMR since TMR is tolerating permanent faults. For this purpose, architectures are mapped to libraries. As resulted netlists are also used for timing behavior and power consumption simulations in further sub-sections, different constraints are applied during synthesis. First of all, short paths of combinational logic CLs (Fig.3) must be defined to be longer than the comparison window generated by the control logic module of the new hybrid fault-tolerant architecture. Besides, in order to simulate timing variations, different short path constraints are applied for three CL copies of fault-tolerant architectures.

Silicon areas of the synthesized architectures are presented in Table I. The first three columns of this table detail characteristics of original benchmark circuits: name (Name), CL input (n) and output (m) numbers. The fourth column shows the silicon area of original logic circuits' combinational part (CL). The fifth column presents the area of hybrid fault-tolerant architectures (HFT) while the sixth column details area of TMR1 solutions. The areas are presented in square micrometers. **Area overhead** of hybrid fault-tolerant technique compared to TMR1 is shown in the next column. This overhead is expressed in percentage of TMR1 architectures' area. The two last columns correspond to area of TMR2 architectures in square micrometers (TMR2) and **area reduction** of hybrid fault-tolerant structures compared to them. This reduction is calculated in percentage of TMR2 architectures' area.

We can observe that area overheads of hybrid fault-tolerant architectures compared to TMR1 solutions are negligible for largest ITC'99 benchmark circuits (about 3.5%). This is not the case for c5315 and c7552 circuits of ISCAS'85 benchmarks. This can be explained by additional area of input register, input demultiplexer and output multiplexer in hybrid fault-tolerant architectures, which are important compared to CLs' size.

Compared TMR2 architectures, the area reduction realized using hybrid fault-tolerant technique is at about 6% except for

c6288 which have small CL input and output numbers.

TABLE I
SYNTHESIZED AREA OF FAULT-TOLERANT ARCHITECTURES

Circuit	n	m	CL (μm^2)	HFT (μm^2)	TMR1 (μm^2)		TMR2 (μm^2)	
					Area	Overhead	Area	Reduction
c5315	178	123	5543	20322	18973	7,1%	21695	6,3%
c6288	32	32	3007	9931	9567	3,8%	10147	2,1%
c7552	206	107	5016	18886	17322	9,0%	20152	6,3%
b14s	278	300	15953	54776	52919	3,5%	58147	5,8%
b15s	486	520	28353	96960	93808	3,4%	102906	5,8%
b20s	523	513	29039	99377	95941	3,6%	105311	5,6%
b21s	523	513	29100	99561	96125	3,6%	105495	5,6%
b22s	768	758	43075	147234	142279	3,5%	156081	5,7%

B. Fault-tolerant capability

This sub-section aims to simulate timing behaviors of the new hybrid fault-tolerant architecture in case of error occurrences at combinational logic modules. This is performed using SPICE-like simulations. For this purpose, the new hybrid fault-tolerant architecture is applied for combinational circuit c6288 from ISCAS'85 benchmark. Besides, in order to simulate transient and permanent error occurrences, an additional 2-input XOR gate is inserted between combinational logic CL2 and output multiplexer Mux of the architecture (Fig.3). This XOR gate uses an additional input signal *fault* to flip the 0^{th} output bit *o2[0]* of CL2.

In following simulations, the new hybrid fault-tolerant architecture is run with *CLK* signal of 10ns period and 10% duty cycle. This period is larger than maximum delay of combinational logic c6288 (8.2ns) and thus, guarantees that no timing error can occur without fault injection. Before running different input vectors from t=70ns, the architecture is initialized so that its pseudo-dynamic comparator is reset and that its initial configuration is CL1 and CL2 running in parallel.

Transient errors

To simulate a transient fault at CLs, a 0-1-0 glitch is injected to *fault* signal at the beginning of the 10^{th} period (from t=90ns to t=100ns). Consequently, the captured value in the output register and one of the running CLs' outputs are different at the beginning of the 10^{th} period. This situation may correspond to either a SET or a timing error that occurs during the 9^{th} period. Besides, in this simulation, primary input vector *PI* is kept unchanged during two *CLK* periods of error detection and correction. As discussed above, this normally is managed at system level.

Simulation results are presented in Fig.10. Signals shown in this figure correspond to that presented in Fig.3: *CLK* signal (clk); primary input (*PI*) and output (*PO*) vectors (pi[31:0] and po[31:0]); *error* signal (error); input vectors *i1*, *i2* and *i3* of CL1, CL2 and CL3 (i1[31:0], i2[31:0] and i3[31:0]); output vectors *vout1* and *vout2* of Mux (vout1[31:0] and vout2[31:0]). The additional fault signal is also presented in Fig.10. Logic value of vectors in this figure is expressed in Hexadecimal.

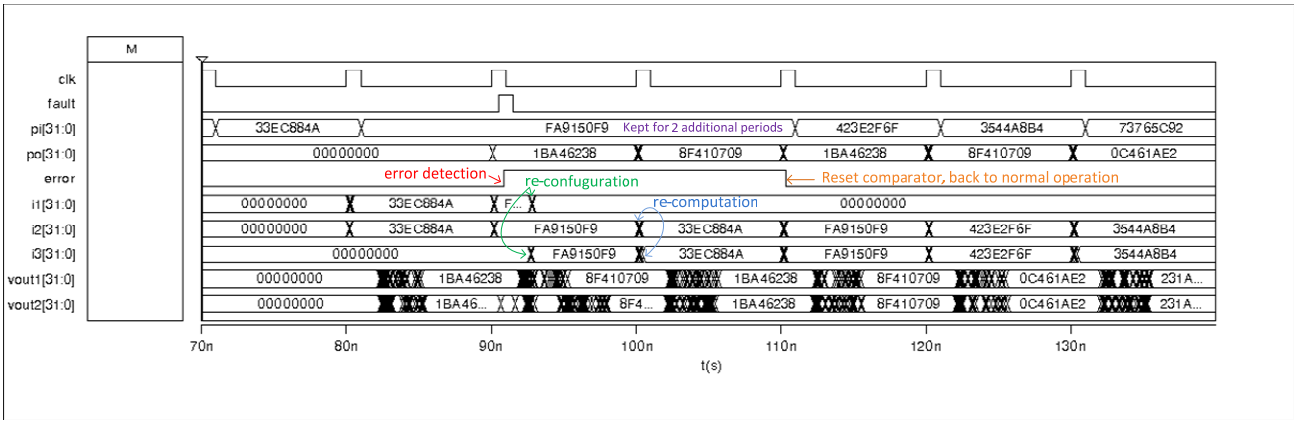


Fig.10.SETs and timing errors tolerance of the new hybrid fault-tolerant architecture

In Fig.10, we can observe that *error* signal turns to logic-1 signaling that *vout2* and captured *PO* have different values during the comparison window. During the same period between t=90ns and t=100ns, *CL1* is put in stand-by (stable logic 00000000_{16} at *i1*) while *CL3* is turn on (*i3* receives captured primary input). The re-configuration successfully finishes before the beginning of new *CLK* period. At next *CLK* positive edge (t=100ns), the previous value of primary input ($33EC884A_{16}$) is applied to *CL2* and

CL3. This shows that the input register and the control logic module have correctly triggered a re-computation. Note that *error* remains at logic-1 during this period. The re-computation finishes before next *CLK* edge at t=110ns. As *vout2* and *PO* are identical, *error* returns to logic-0 signaling that the captured output is correct. The transient error is tolerated by the architecture with two additional *CLK* periods.

Permanent errors

To simulate a permanent fault at CLs, *fault* signal is kept at logic-1 from t=90ns. Consequently, CL2 output is permanently affected. Simulation results are presented in Fig.11 with the same signals of Fig.10.

In the first two periods after error occurrence (from t=90ns to t=110ns), the architecture works exactly like in case of transient errors. It is re-configured so that CL2 and CL3 run in parallel to re-compute the input vector 33EC8B4A. However, after CLK edge at t=110ns, *error* signal is briefly reset to logic-0, then return to logic-1, signaling that *PO* and *vout2* are not identical. This is caused by the permanent error that remains in CL2. Consequently, a new re-configuration is done during the period between t=110ns and t=120ns. CL2 is put on stand by while CL1 is turned on. After re-computation during next period, the permanent error is tolerated and *error* returns to logic-0 at t=130ns.

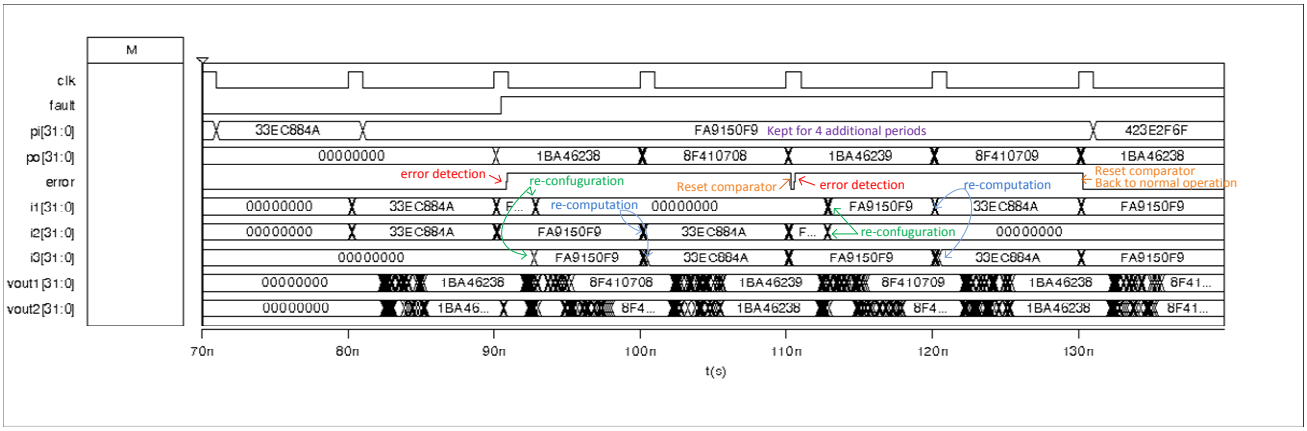


Fig.11.Hard errors tolerance of the new hybrid fault-tolerant architecture

C. *Power consumption*

Besides the area overhead and the fault-tolerant capability, power saving is another advantage of the proposed hybrid fault-tolerant architecture compared to other solutions. In this sub-section, we compare power consumption of the new hybrid fault-tolerant architecture with TMR1 and TMR2methods. In order to perform such comparison, we use transistor level evaluations that consist of running SPICE-like simulations of synthesized architectures obtained previously. For each original logic circuit, the three corresponding fault-tolerant architectures are fed by the same set of 100 random input vectors. Average currents at power supply node VDD are monitored to deduce average power consumption of the architectures. Note that the number of random input vectors is chosen so that simulation results may represent typical power consumption of fault-tolerant architecture during normal operations, while simulations can be done with available resources of our simulators (time and memories).

Due to limit of time and memories available for simulation, we only evaluate power consumption of largest ISCAS'85 benchmark circuits. Table II presents results of these simulations. In this table, the first three columns correspond to name (Name), input (n) and output (m) numbers of the benchmark circuits. The three next columns detail average power consumption of the hybrid fault-tolerant (HFT), the TMR1 and the TMR2architectures. All power consumptions are calculated in milliwatt. The two final columns show average power saving of hybrid fault-tolerant architectures compared to TMR1 and TMR2architectures. These values are expressed in percentage of corresponding TMR architecture's average power consumption.

TABLE II
POWER SAVING OF THE NEW HYBRID FAULT-TOLERANT ARCHITECTURE
COMPARED TO TMR METHODS

Name	n	m	Average (mW)			Reduction	
			HFT	TMR1	TMR2	TMR1	TMR2
c5315	178	123	3.5	5.0	5.4	30.3%	35.4%
c6288	32	32	5.5	8.4	8.4	34.7%	35.3%
c7552	206	107	4.9	7.1	7.9	30.3%	37.6%

In the hybrid fault-tolerant architecture, only two CLs are running in parallel. This gives us about 33.3% power saving

compared to three operating CLs in TMR architectures. In Table II, we observe that average power saving is comparable to this ratio.

For c5315 and c7552 benchmark circuits, hybrid fault-tolerant technique reduces about 30% of power consumption compared to TMR solution. This value, which is smaller than expected, may be explained by power consumption overhead introduced by redundant modules such as input demultiplexer, output multiplexer and shadow latches in input registers of the new hybrid fault-tolerant architecture. This is not the case for c6288, because its input and output numbers are small, which lead to much less additional hardware in hybrid fault-tolerant architecture. Besides, we can see that for this circuit, power reduction value is slightly higher than expected. This can be explained by the fact that pseudo-dynamic comparators consume less than word voters.

For TMR2 solution, power reduction values that are higher than 33.3% are due to the fact that additional registers in TMR architectures consume more than input demultiplexer and output multiplexer in hybrid fault-tolerant architectures.

Note that Table II only contains power consumption estimation for ISCAS'85 benchmarks circuits. However, for larger circuits such as ITC'99, the result of power saving using the hybrid fault-tolerant architecture compared to TMR techniques must be of the same order, *i.e.* about 30% of TMRs' power consumption.

V. CONCLUSION

In this paper, we have presented a new hybrid fault-tolerant architecture to improve robustness of digital CMOS circuits and systems. This architecture also employs information redundancy (duplication/comparison) for error detection, timing redundancy (re-computation) for transient error correction and hardware redundancy (re-configuration) for permanent error correction. Furthermore, it introduces the use of a pseudo-dynamic comparator and modified D flip-flop in a configuration, which allows detection/correction of hard, SET and timing errors in combinational part of logic circuits. Comparison with TMR methods using ISCAS'85 and ITC'99 benchmark circuits shows that the new architecture reduces power consumption by about 33% while keeping area overhead as low as 3%-6%. Another advantage of the new hybrid fault-tolerant architecture is that it can be used together with information redundancy methods, which enable SEU protection of combinational elements with optimized area overhead. Such architectures, which tolerate all kind of faults in logic circuits, will be the subject of further researches.

REFERENCES

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", 2011.

[2] I. Koren, C. M. Krishna, "Fault-Tolerant Systems", Ed. Organ Kaufmann, 2007.

[3] M. Nicolaidis, L. Anghel, N. Achouri, "Memory Defect Tolerance Architectures for Nanotechnologies", J. of Electronic Testing, Vol. 21, Issue 4, pg. 445-455, August 2005.

[4] Chin-Lung Su, Yi-Ting Yeh, Cheng-Wen Wu, "An Integrated ECC and Redundancy Repair Scheme for Memory Reliability Enhancement", Proc. of the 20th Int. Sym. on Defect and Fault-Tolerance in VLSI Systems (DFT'05), pg. 81-92, 2005.

[5] R. E. Lyons, W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability", IBM Journal of Research and Development, Vol. 6, Issue 2, pg. 200-209, April 1962.

[6] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, S. J. Patel, "Sequential element design with built-in soft error resilience," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 12, pg. 1368–1378, December 2006.

[7] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", Proc. of the 36th Annual IEEE/ACM Int. Sym. on Microarchitecture (MICRO-36), pg. 7-18, December 2003.

[8] S. Das, C. Tokunaga, S. Pant, W. -H. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, D. T. Blaauw, "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance", IEEE J. of Solid-State Circuits, Vol. 44, Issue 1, pg. 32-48, January 2009

[9] M. E. Imhof, H.-J. Wunderlich, "Soft Error Correction in Embedded Storage Elements", Proc. of IEEE International On-Line Testing Symposium (IOLTS11), pp. 169-174, 2011.

[10] J. Vial, A. Bosio, P. Girard, C. Landrault, S. Pravossoudovitch and A.Virazel, "Using TMR Architectures for Yield Improvement", Int. Symp.on Defect and Fault-tolerance in VLSI Systems, pp. 7-15, 2008.

[11] J. Vial, A. Virazel, A. Bosio, P. Girard, C. Landrault and S.Pravossoudovitch, "Is TMR Suitable for Yield Improvement?", IET Computers and Digital Techniques, vol. 3, No 6, pp. 581-592,November 2009.

[12] D. A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, H.-J. Wunderlich, "A Hybrid Fault Tolerant Architecture for Robustness Improvement of Digital Circuits", Proc. of the 20th IEEE Asian Test Symposium (ATS11), pp. 136-141, 2011.

[13] D.A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, A. Todri, M.E. Imhof, H.-J. Wunderlich, "A Pseudo-Dynamic Comparator for Error Detection in Fault Tolerant Architectures", Proc. of the 30th IEEE VLSI Test Symposium (VTS12), pp. 50-55, 2012.

[14] M. Nicolaidis, "Graal: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies", Proc. of IEEE International Test Conference (ITC07), pp.1–10, 2007.

[15] S. Valadimas, Y. Tsiatouhas, A. Arapoyanni, "Timing error tolerance in nanometer ICs", in Proc. of IEEE IOLTS, pp.283-288, 2010.

[16] D. J. Palframan, N. S. Kim, M. H. Lipasti, "Time Redundant Parity for Low-Cost Transient Error Detection", in Proc. of IEEE Design, Automation & Test in Europe, pp. 1-6, March 2011.

[17] S. Mitra, E. J. McCluskey, "Word-voter: a new voter design for triple modular redundant systems", Proc. of the IEEE 18th VLSI Test Symposium, pg. 465-470, 2000.

[18] Nangate, 45nm Open Cell Library v1.3, http://www.nangate.com, 2009.

[19] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration", ACM Journal on Emerging Technologies in Computing Systems , Vol. 3, No. 1, 2007.

[20] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F.Love, "FreePDK: An Open-Source Variation-Aware Design Kit", IEEE Int. Conf. on Microelectronic Systems Education, pg. 173-174, 2007.

[ref1] P. Shivakumar et al., “Modeling the effect of technology trends on the soft error rate of combinational logic”, Int. Conf. on Dependable Systems and Networks, pp. 389-398, 2002.
[ref2] J. Velamala et al., “Design sensitivity of Single Event Transients in scaled logic circuits”, Design Automation Conf., pp. 694-699, 2011.
[ref3] N. Avirneni, V. Subramanian and A.K. Somani, “Low overhead Soft Error Mitigation techniques for high-performance and aggressive systems”, Dependable Systems & Networks, pp.185-194, 2009.
[ref4] S. Gupta, F. Shuguang Feng, A. Ansari, J. Blome and S. Mahlke, “The StageNet fabric for constructing resilient multicore systems”, Int. Symp. on Microarchitecture, pp.141-151, 2008.
[ref5] J. Yao, H. Shimada, K. Kobayashi, “A Stage-Level Recovery Scheme in Scalable Pipeline Modules for High Dependability”, Int. Workshop on Innovative Architecture for Future Generation High Performance, pp. 21-29, 2010.
[ref6] M. Mehrara, M. Attariyan, S. Shyam, K. Constantinides, V. Bertacco and T. Austin, “Low-Cost Protection for SER Upsets and Silicon Defects”, Design, Automation & Test in Europe Conference, pp. 1-6, 2007.