An Experimental Comparative Study of Fault-Tolerant Architectures

Imran Wali, Arnaud Virazel, Alberto Bosio, Patrick Girard
LIRMM – University of Montpellier / CNRS
Montpellier, France
e-mail: {wali, virazel, bosio, girard}@lirmm.fr

Abstract—**This paper provides a comparative study based on experiments performed on four similar fault-tolerant architectures intended to reduce errors caused due to faults in combinational logic parts of microelectronic circuits and systems. The compared merits include area, power, performance and fault tolerance capability. The experimental results show that the improved Hybrid Fault-Tolerant Architecture can handle transient faults as effectively as Partial-TMR and exhibits permanent fault tolerance capability similar to that of Full-TMR. It offers 11.8% and 20.5% power saving compared to Partial and Full-TMR respectively. Furthermore, it can handle the fault accumulation effect better than TMR, hence an ideal candidate for low-power long duration mission-critical applications.**

Keywords-fault tolerant architecture; fault tolerance capability assessment.

I. Introduction

Complementary metal-oxide semiconductor (CMOS) device scaling is posing reliability challenges to future microelectronic circuits and systems [1]. Other alternative and evolutionary technologies are also facing reliability issues in their early development life cycles. Design architects must address the concern of preventing reliability from becoming a bottleneck for the development of high-performance, low-power systems, through the use of fault-tolerant techniques.

These techniques are commonly used to tolerate on-line faults, i.e., faults that appear during the normal functioning of the system, irrespective of their transient or permanent nature [2]. They use redundancy, i.e., the property of having spare resources that perform a given function and tolerate faults in the combinational [3]-[5] and/or sequential [6]-[9] part of the circuit. These techniques are generally classified by the type of redundancy used. Basically, three types of redundancy are considered: information, temporal and hardware [2].

Many studies in literature like [10]-[12] provide evaluation results within the scope of the architecture proposed therein. However, it is essential that these similar schemes be comprehensively compared using identical set of experiments and conditions in order to have a meaningful contrast. For any fault-tolerant architecture, the four merits that are essential to be analyzed are its area, power and performance overheads and most importantly its fault tolerance capability. Among these four merits area, power and performance can be evaluated using conventional circuit analysis tools. Unlike these attributes of a fault-tolerant architecture, fault tolerance capability cannot be evaluated using standard circuit analysis methodologies, but only by observing system behavior in the presence of faults [13].

In this paper, we present a comprehensive experimental comparative study of four fault-tolerant architectures with similar fault-tolerance capability in the context of spatial and temporal characteristics of faults and the architectural cost merits, which include area and power consumption. These architectures include Partial Triple Modular Redundancy (Partial-TMR) and Full Triple Modular Redundancy (Full-TMR) [2], Hybrid Fault-Tolerant (HyFT) [14][15] and improved Hybrid Fault-Tolerant (*i*HyFT) [16] architectures. For assessing the merits of these fault-tolerant architectures, we implement them on some ITC'99 benchmarks and use a Gate-level simulation based fault-injection framework to quantitatively assess and compare the fault tolerance capability of these schemes.

The remaining parts of this paper are organized as follows. Section 2 highlights the problematic of error occurrences in combinational logics and storage elements. Section 3 presents the fault-tolerant architectures under comparison. Section 4 details the experimental methodology while Section 5 gives results in terms of area, power, performance and fault-tolerance capability. Finally, Section 6 concludes the paper and provides some perspectives.

II. Problem Statement

Lidén et al. in 1994 experimentally estimated that only 2% of bit flips in memory elements also known as Single Event Upset (SEU) were caused by particle-induced transients or Single Event Transients (SET) generated in and propagated through Combinational Logic (CL). The rest were due to direct particle strike in latches. Their experiments involved using a 1μm CMOS process at 5MHz [17]. Since then physical gate-length has downscaled up to 50 times, supply voltages have dropped to 0.9 V and operating frequency has shown a thousand fold increase [1]. This massive change in technology has resulted in greater sensitivity of memory elements to high-energy particle, but the effects are more pronounced on CL networks [18]. A more recent work uses a probability model to estimate that the susceptibility to CL circuits to SET nearly doubles as the technology scales from 45 nm to 16 nm [19]. As a result research attention drawn towards developing techniques to limit Soft Error Rate (SER) in CL is becoming comparable to effort made in protecting state elements. Figure 1 symbolically illustrates the share and types of problems arising from sequential logic and combinational logic parts of digital circuit.

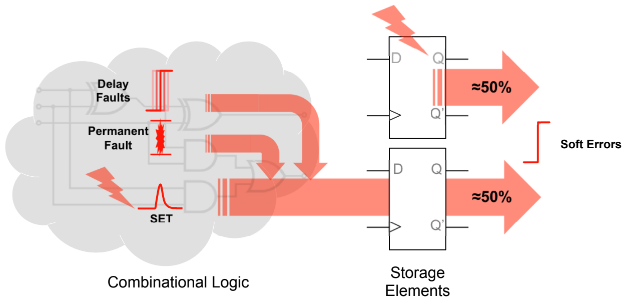


Figure 1. Error occurrences in combinational logics and storage elements

III. FAULT-TOLERANT ARCHITECTURES

Several hardware fault-tolerant architectures have been proposed in the literature [20]. The classical hardware redundancy architecture is the N Modular Redundancy (NMR). A NMR structure is a fault-tolerant architecture based on N modules performing the same function. The outputs of these modules are compared by using a majority voter. The case of $N = 3$ is called TMR and has been widely studied and used in practical system applications [2][3].

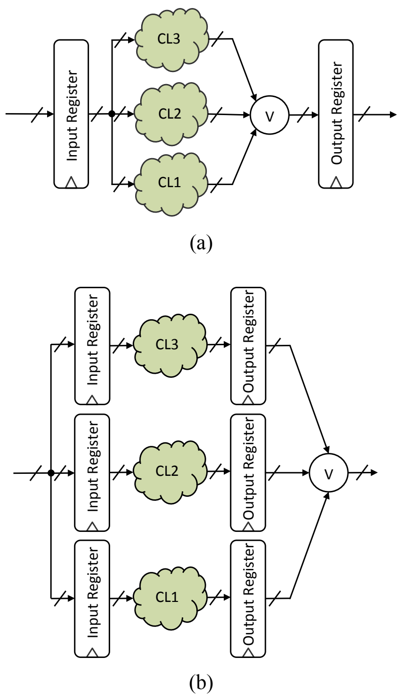


Figure 2. TMR Architectures (a) Partial-TMR and (b) Full-TMR

There are different methods to implement TMR architecture for logic circuits, depending on which part of this circuit is triplicated. In Figures 2.a and 2.b, we present two TMR structures that will be compared with the hybrid fault-tolerant architecture. The first implementation (Partial-TMR, Figure 2.a) consists of triplicating only Combinational Logic (CL) part of the logic circuit while the second one (Full-TMR, Figure 2.b) requires triplications of both combinational and sequential parts.

While having smaller area overhead, the partial-TMR solution cannot tolerate SEUs or permanent faults in pipeline registers. This problem can be solved using full-TMR solutions by triplicating the registers. Note that in full-TMR input registers are also triplicated so that errors caused by each register can be tolerated.

The second fault-tolerant architecture under comparison is the HyFT scheme presented in [14, 15]. This architecture employs information redundancy (duplication/comparison) for the error detection, timing redundancy (re-computation) for the transient error correction and hardware redundancy (re-configuration) for the permanent error correction. As presented in Figure 3.a, the hybrid architecture employs three copies of CL (CL1, CL2 and CL3) modules. The input demultiplexer and the output multiplexer are used to select two running CL copies and to put the third CL copy in standby mode.

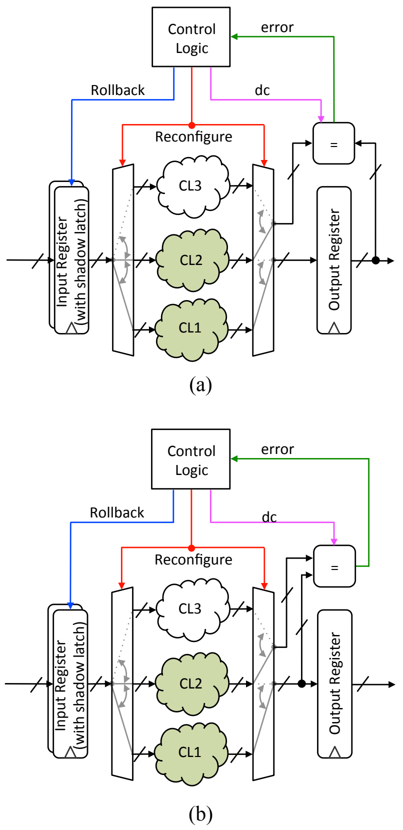
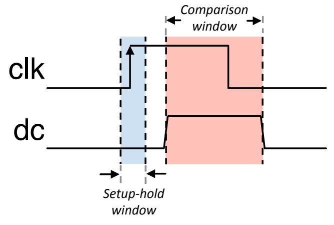
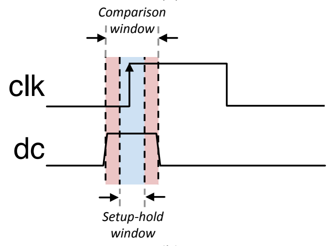


Figure 3. HyFT Architectures (a) HyFT and (b) *i*HyFT

The HyFT architecture is driven by a control logic module, which is divided in two parts. The first part consists of a state-machine that controls different configurations of the architecture, i.e., it decides which two CL copies to run in parallel. The second part controls the comparator, pipeline register, demultiplexer and multiplexer. For error detection it uses the pseudo-dynamic comparator presented in [21]. It combines a dynamic transition detector and a static comparator in order to detect hard, soft and timing errors during a *comparison-window* as shown in the timing diagram of Figure 4.a. The comparison takes place only during these brief intervals of time represented as red shaded regions with doted outline in Figure 4. The timing of *comparison-window* is defined by the high phase of a delayed clock signal '*dc*'.



(a)



(b)

Figure 4. Comparison-window timing for (a) HyFT and (b) *i*HyFT

Figure 3.b presents the improved version of the HyFT (*i*HyFT) architecture, which resulted as an attempt to improve the error detection capability and to reduce the performance overhead [16]. This improved scheme achieves aforementioned objectives by using a *comparison-window* across the *setup-hold window* as shown in Figure 4.b. With this comparison timing it can intrinsically detect erroneous signal transients that are more likely to be captured in the output register. The *comparison-window* timing was made possible by changing the placement of the comparator such that the comparator compares the output of two running CL copies directly from the multiplexer as shown in Figure 3.b. The ability to act against only the potentially fatal SETs not only reduces the number of fail-silent faults but also improves the performance.

IV. EXPERIMENT METHODOLOGY

Experiments are performed to compare the merits of the four fault-tolerant architectures presented in Section III. Each architecture is applied to a few of the ITC'99 benchmark circuits and are synthesized using NanGate 45nm Open Cell Library [22]. The area figures are obtained from the synthesized designs and power estimates are obtained by taking into account the switching activity generated by back-annotated gate-level simulations. The workload for simulation is a set of patterns optimized for stuck-at fault detection. The reason for using such a workload is to obtain switching activity distributed in all parts of the circuit.

The performance overhead is evaluated in two different aspects. Firstly, in terms of temporal performance degradation, which is basically the additional delay in the data-path due to the fault-tolerant architecture (e.g., voter delay in TMR), and secondly in terms of error recovery penalty under a certain fault rate.

The fault-tolerance capability of the four schemes is estimated by performing fault injection in the combinational logic parts of the circuits, by using a gate-level simulation based fault-injection framework. The framework uses the switching activity file to extract the list of all possible fault-locations. From this list it randomly selects a subset of locations for fault-injection. To each fault location in this subset, it randomly assigns a fault-injection time within the limits of simulation time duration and a SET duration also randomly selected from the range of typically anticipated SET pulses, i.e. from 0.25ns to 1.25ns [23]. Once the fault list is prepared, fault injection campaigns that comprise a number of simulations are run. Either a single SETs or a permanent stuck-at fault is injected per simulation by forcing the signals at the specified location, at the corresponding time indicated by the fault list.

During the fault injection campaign a fault-injection report is generated which contains the cycle-by-cycle outcome of each simulation. At the end of fault-injection campaign the fault-injection report is analyzed to classify the faults according to the fault effects into three categories:

1. **Silent faults**: the faults that have no impact on the workload computation nor are detected by the fault-tolerant architecture.
2. **Corrected faults**: the faults that are detected and corrected by the fault- architecture in place.
3. **Fail-Silent fault**: the faults, which result in a wrong computed result but are not detected by the fault-tolerant architecture.

The ratio of the number of fail-silent faults to the number of total injected faults gives us a figure to compare the fault tolerance capability of the four different schemes.

V. COMPARATIVE ANALYSIS

A. Area and Power Overhead

Table I gives the average area and power for the BaseLine (BL) circuits and the fault-tolerant schemes based on the results of their implementation on six ITC'99 benchmark circuits. It also gives their associated overheads of area and power with the BL circuits as reference.

TABLE I. AREA AND POWER ESTIMATION RESULTS

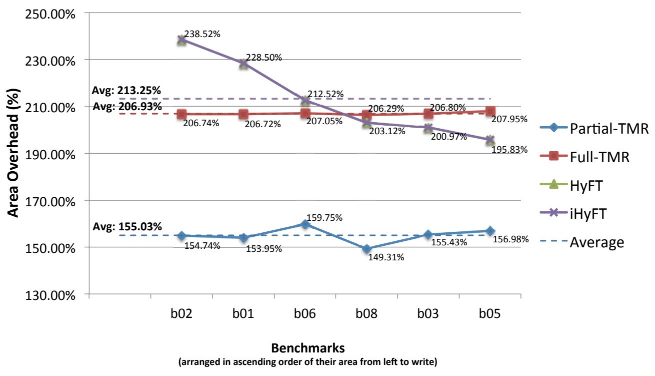
	Avg Area (μm^2)	**Avg Area overhead (%)**	**Avg Power (μW)**	**Avg Power overhead (%)**
BL	1231.00	0	351.50	0
Partial-TMR	3141.59	155.02	971.66	173.32
Full-TMR	3781.32	206.93	1077.74	206.09
HyFT	3739.43	213.24	859.67	157.36
*i*HyFT	3739.43	213.24	856.49	156.83

The most obvious area and power overhead figures are those of Full-TMR. As it is based on triplicating the CL blocks and also the registers, it occupies a little more than three times the area and consumes a few microwatts over BL. This extra area and power is due to the voter in the Full-TMR scheme.

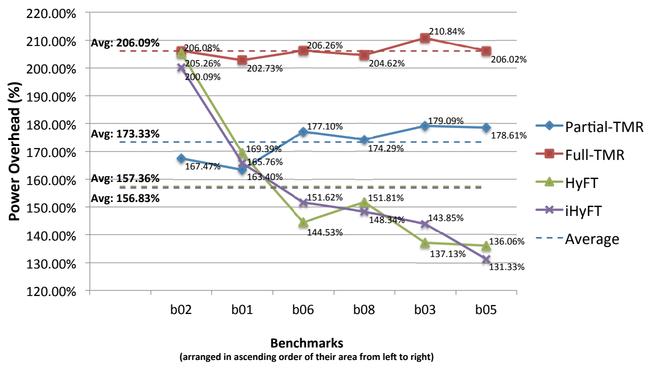
The average percentage of area overhead values in Table I show that the partial-TMR implementation consumes less in terms of area that is about 155%. The two most expensive architectures in terms of area are HyFT and *i*HyFT with an average overhead of around 213% on average for the considered set of benchmark circuits.

As far as the power consumption is concerned HyFT and *i*HyFT architectures are most efficient based on the average power overhead figure of about 157% in Table I. Partial-TMR stands at 173%, making Full-TMR the least power efficient scheme. This high power consumption is accounted to the triplication of sequential elements. On the other hand HyFT and *i*HyFT save power by having one CL copy in stand-by all the time.

The graphs in Figure 5 show the percentage increase in area (Figure 5.a) and power (Figure 5.b) of the BL circuits to implement the four fault-tolerant architectures discussed in Section III. Note that the benchmark circuits are arranged in ascending order of their size from left to right on X-axis to illustrate the impact of the size of CL block on the area and power overheads. The dotted lines in Figure 5 represent the average percentage figures of area and power overheads for the corresponding fault-tolerant architecture implementation.



(a)



(b)

Figure 5. Impact of CL block size on (a) Area and (b) Power Overhead

An important observation that can be made in the graphs of Figure 5 is that, the area and power overheads of both partial and full-TMR are relatively independent of the size of CL block to which they are applied. However, these overheads for HyFT and *i*HyFT change with different sizes of benchmarks such that the area and power overheads of HyFT and *i*HyFT decrease with the larger benchmarks. This observation also gives an idea of the anticipated impact on the area and power overheads for CL blocks larger than the benchmarks considered in this study. Although the average area overhead of HyFT and *i*HyFT is higher than other considered fault-tolerant architectures but with large CL blocks we can expect it to decrease. Where as the power overhead of HyFT and *i*HyFT, which is already the minimum, tends to further reduce with larger CL blocks.

B. *Performance*

The first evaluated measure of performance is the temporal performance degradation. In partial-TMR and full-TMR, it is defined by the delay of voter circuit in the data-path. In case of HyFT and *i*HyFT it is due to the delay of shadow latch multiplexers in input register responsible for rollback and the reconfiguration multiplexer and demultiplexer. The comparator being outside the critical path does not contribute to the temporal performance degradation. Using static timing analysis the temporal performance degradation for partial-TMR and full-TMR was estimated to be 0.73% for a 100MHz operation. The same for HyFT and *i*HyFT was found to be 9.7% without any design optimization.

The figures that can give us a measure of the second considered performance aspect, i.e. the error recovery penalty, can be interpreted from the transient fault injection results presented in Table II. These results are obtained by injecting transient faults at an average rate of 250K faults/second.

TABLE II. TRANSIENT FAULT INJECTION EXPRIMENT RESULTS SUMMARY

	Avg % of Silent faults	***Avg % of Corrected faults***	***Avg % of Fail-silent faults***
BL	92.51%	0.00%	7.49%
Partial-TMR	99.97%	0.00%	0.03%
Full-TMR	100%	0.00%	0.00%
HyFT	92.46%	7.26%	0.28%
*i*HyFT	94.10%	5.86%	0.04%

It can be observed in Table II that for partial-TMR and full-TMR the percentage of corrected faults is zero. This is because; TMR is an error masking technique rather than an error detection and correction one and does not indicate the presence of error. With no provision of identifying the corrected faults, they are kept within the category of silent faults in our analysis. It also indicates that the error recovery penalty for TMR is zero as it corrects errors by masking them instead of undergoing a reconfiguration and re-computation cycle. It can also be seen in Table II that HyFT corrected on average 7.26% of injected faults. For each detected and corrected SET the HyFT undergoes a recovery phase that takes 2 additional cycles [14]. According to these figures, HyFT spends around 14.52% of total computation time on recovering from potentially erroneous states. On the other side, *i*HyFT spends 11.72% of time in recovery phase under the same fault rate.

C. Fault-tolerance capability

1) Quantitative analysis

To compare the fault-tolerance capability of different architectures we analyze them in terms of the percentage of faults that resulted in a fail-silent outcome among the total number of injected faults. Table III gives the transient fault-injection experiment results.

TABLE III. TRANSIENT FAULT INJECTION RESULTS

	Percentage of fail-silent faults (%)				
	BL	***Partial-TMR***	***Full-TMR***	***HyFT***	***iHyFT***
b01	7.49	0.00	0.00	0.37	0.12
b02	8.11	0.11	0.00	0.28	0.11
b03	8.18	0.03	0.00	0.26	0.03
b05	7.11	0.02	0.00	0.21	0.00
b06	7.07	0.04	0.00	0.33	0.09
b08	7.45	0.05	0.00	0.33	0.08
Average	**7.56%**	**0.03%**	**0.00%**	**0.28%**	**0.04%**

Table III shows that the incorporation of each fault-tolerant architecture into the BL circuit reduces the percentage of fail-silent faults to a different extent. The percentage of fail-silent faults that was originally 7.56% in BL is brought down to 0.03% by partial-TMR. A through analysis of the fault-injection report revealed that these 0.03% faults were among those which were injected at the inputs of CL blocks and effected all the three TMR copies in the same way, thus resulted in a common-mode failure. Full-TMR on the other hand did not encounter this problem because of it construction and turned out to be the most effective by tolerating the effects of all the injected transient faults.

In case of HyFT 0.28% of injected faults escaped detection and effected the results. With further investigation we found out that these fail-silent outcomes were not linked to a specific location as in case of partial-TMR, but escaped detection due to their specific timing characteristics. Static timing analysis showed that these 0.28% fail-silent faults were among those that were injected at a time such that their effects appeared at the inputs of register during the clock *setup-hold window*. Since in HyFT the *comparison-window* does not overlap the *setup-hold window* as discussed in Section III and shown in Figure 4.a, these transient faults managed to affect the data during captured but escaped detection by missing the *comparison-window*. This problem of non-overlapping *setup-hold window* and *comparison-window* was solved by *i*HyFT and therefore significant reduction in the percentage of fail-silent faults is observed in *i*HyFT of about 0.04%.

Similar observations can be made form the permanent fault injection results shown in Table IV. An average 1.36% of faults injected in partial-TMR result in fail-silent outcome, mainly due to the common-mode effect. Full-TMR and *i*HyFT show nearly complete tolerance against permanent faults and in HyFT 0.08% faults escaped detection mainly due to the *setup-hold window* and *comparison-window* separation.

TABLE IV. PERMANENT FAULT INJECTION RESULTS

	Percentage of fail-silent faults (%)				
	BL	***Partial-TMR***	***Full-TMR***	***HyFT***	***iHyFT***
b01	98.37	2.37	0.00	0.15	0.02
b02	96.28	2.03	0.00	0.06	0.00
b03	98.15	1.38	0.00	0.06	0.00
b05	97.84	0.50	0.00	0.08	0.00
b06	97.23	0.66	0.00	0.13	0.00
b08	98.03	2.34	0.00	0.07	0.00
Average	**98.03%**	**1.36%**	**0.00%**	**0.08%**	**0.00%**

2) Qualitative analysis

Some aspects of fault-tolerance capability that have an implication on the lifetime reliability of the circuit cannot be inferred from the fault injection experiment result discussed in the previous section. Therefore, we analyze them qualitatively in this section.

When a circuit enters into the wear-out phase of it's lifetime, most of the wear-out mechanisms show early symptoms as increasing signal propagation latency prior to inducing permanent device failures [24]. The ability of the HyFT and *i*HyFT architectures to detect these early symptoms and act upon by causing reconfigurations reduces the aging effects on the system by distributing the stress on two of the three CL copies. The capability of selective sparing helps reduce the rate of failures and increase the life span of circuit parts that embed such fault-tolerant architecture.

Another qualitative aspect of fault-tolerance capability is fault accumulation effect that distinguishes both considered versions of TMR from HyFT and *i*HyFT. TMR is an error masking architecture that does not indicate the presence of error, instead just corrects them until only one computational copy exhibits an error. When faults accumulate due to wear-out and multiple copies start getting affected, TMR fails to correct them and the lack of any provision of indicating error ends up in fail-silent outcomes. Whereas HyFT and *i*HyFT are able to correct errors until two faulty copies manifest the effect of fault at the output in a same way at the same time, which is very less likely. In all other possible scenarios HyFT and *i*HyFT, if cannot correct can at least indicate the presence of error and continue fail-safe operation.

VI. CONCLUSION

In order to produce a meaningful comparison of the state-of-art fault-tolerant architectures, we present herein an experimental analysis based on standard circuit analysis tools and a simulation based fault-injection framework to obtain results in terms area, power and performance overheads and fault tolerance capability. The results show that the improved Hybrid fault-tolerant architecture saves notable amount of power, while offering similar robustness improvements as

TMR. In addition it's lifetime reliability improvement and ability to deal with fault accumulation effect makes it feasible for low-power mission critical applications.

We intend to continue the analysis with larger benchmark circuits to validate the projected effectiveness of HyFT and *i*HyFT when used with larger combinational logic blocks. We also aim to perform multiple fault injection campaigns to quantitatively access the lifetime reliability improvement and the fault accumulation effects in different fault-tolerant architectures.

ACKNOWLEDGMENT

This work has been partially funded by "National Centre for Scientific Research" (CNRS) under the framework of "International Associated Laboratory - French-Italian research Laboratory on hardware-software Integrated Systems" (LIA - LAFISI).

REFERENCES

[1] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors (ITRS) 2013", Retrieved Aug, 2015 from http://www.itrs.net/reports.html2013.

[2] I. Koren, and C. Krishna, "Fault Tolerant Systems", Morgan Kauffman Publisher, 2007.

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

[4] 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, Oct 2008, pp. 7-15.

[5] 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, Nov 2009, pp. 581-592.

[6] M. Zhang et al., "Sequential element design with built-in soft error resilience," IEEE Transactions on Very Large Scale Integration Systems, Vol. 14, No. 12, Dec 2006, pp. 1368–1378.

[7] D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," Proc. of the 36th Annual IEEE/ACM Int. Sym. on Microarchitecture, Dec 2003, pp. 7-18.

[8] S. Das et al., "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance," IEEE J. of Solid-State Circuits, Vol. 44, Issue 1, Jan 2009, pp. 32-48.

[9] M. E. Imhof, and H.-J. Wunderlich, "Soft Error Correction in Embedded Storage Elements," Int. On-Line Testing Symp., July 2011, pp. 169-174.

[10] J. Yao et al., "DARA: A Low-Cost Reliable Architecture Based on Unhardened Devices and Its Case Study of Radiation Stress Test," IEEE Transactions on Nuclear Science, Dec 2012, vol. 59, no. 6, pp. 2852-2858.

[11] V. Subramanian, and A.K. Somani, "Conjoined Pipeline: Enhancing Hardware Reliability and Performance through Organized Pipeline Redundancy," 14th IEEE Pacific Rim International Symposium on Dependable Computing, Dec 2008, pp. 9-16.

[12] 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, April 2007, pp. 1-6.

[13] A. Benso, Alfredo, and P. Prinetto, "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation," Springer US, 2003.

[14] D. A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and H.-J. Wunderlich, "A Hybrid Fault Tolerant Architecture for Robustness Improvement of Digital Circuits," Asian Test Symposium, Nov 2011, pp. 136-141.

[15] I. Wali, A. Virazel, A. Bosio, L. Dilillo, and P. Girard, "An Effective Hybrid Fault-Tolerant Architecture for Pipeline Cores," IEEE European Test Symposium, May 2015, pp. 1-6.

[16] I. Wali, A. Virazel, A. Bosio, P. Girard, and M. Sonza Reorda, "Design Space Exploration and Optimization of a Hybrid Fault-Tolerant Architecture," to appear in Proc. of IEEE Int. On-Line Test Symp., 2015.

[17] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson "On Latching Probability of Particle Induced Transients in Combinational Networks," Symp. on Fault-Tolerant Computing, June 1994, pp. 340–349.

[18] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," Int. Conf. on Dependable Systems and Networks, June 2002, pp. 389-398.

[19] J. Velamala, R. LiVolsi, M. Torres, and C. Yu, "Design sensitivity of Single Event Transients in scaled logic circuits," 48th Design Automation Conference, June 2011, pp. 694-699.

[20] P. K. Lala, "Self-Checking and Fault-Tolerant Digital Design", Morgan Kauffman Publisher, 2000.

[21] D. A. Tran, A. Virazel, A. Bosio, L. Dilillo, P. Girard, A. Todri, M.E. Imhof, and H.-J. Wunderlich, "A Pseudo-Dynamic Comparator for Error Detection in Fault Tolerant Architectures," VLSI Test Symposium, April 2012, pp. 50-55.

[22] NanGate FreePDK45 Open Cell Library, Retrieved Aug, 2015, from: http://www.nangate.com/?page_id=2325.

[23] G. Wirth, Kastensmidt, L. Fernanda, and I. Ribeiro, "Single Event Transients in Logic Circuits_Load and Propagation Induced Pulse Broadening," IEEE Transactions on Nuclear Science, Dec 2008, vol. 55, no. 6, pp. 2928-2935.

[24] J. A. Blome, S. Feng, S. Gupta, and S. Mahlke, "Online timing analysis for wearout detection," In Proc. of the 2nd Workshop on Ar- chitectural Reliability, Dec 2006, pp. 51-60.