Integrating scan design and soft error correction in low-power applications

Michael E. Imhof, Hans-Joachim Wunderlich, Christian G. Zoellin
Institut fuer Technische Informatik
Universitaet Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
email: {imhof, wu, zoellin}@iti.uni-stuttgart.de

***Abstract*— Error correcting coding is the dominant technique to achieve acceptable soft-error rates in memory arrays. In many modern circuits, the number of memory elements in the random logic is in the order of the number of SRAM cells on chips only a few years ago. Often latches are clock gated and have to retain their states during longer periods. Moreover, miniaturization has led to elevated susceptibility of the memory elements and further increases the need for protection.**

This paper presents a fault-tolerant register latch organization that is able to detect single-bit errors while it is clock gated. With active clock, single and multiple errors are detected. The registers can be efficiently integrated similar to the scan design flow, and error detecting or locating information can be collected at module level. The resulting structure can be efficiently reused for offline and general online testing.

***Keywords*— Robust design, fault tolerance, low power, latch, register, single event effects**

I. Introduction

Today's technology scaling comes with effects concerning both power consumption and reliability. The paper presented here deals with the combination of both aspects. One of the reliability issues are single event effects (SEE) due to radiation. Already in the 70's, memory cells were subject of investigations of radiation induced error rates [1], and nowadays, SEEs are of concern for static memories [2], latches and flip-flops [3] and even for random combinational logic [4, 5].

Up to now, a large variety of hardening techniques has been proposed for the different types of structures. As the soft error rate (SER) of memory elements in random logic is continuously increasing [6] and as the amount of flip-flops and latches is rapidly growing, this paper is contributing to the protection of these storage elements.

Besides the high vulnerability to SEEs, scaling leads to an increased power density on chip which prohibits a frequency increase as seen in the past. The classic low power design techniques have still increasing relevance [7, 8], but must be complemented by massive parallelism [9] and power management in networks and systems on a chip. While power gating is employed, if modules will be unused for a rather long period of time, and their state is not needed any more in the course of further computations, clock gating is favored for shorter breaks where the computation will be resumed and the state must be retained.

There is a variety of protection schemes against single-event upsets (SEU) for flip-flops available [10, 11, 12, 13, 3, 14, 15, 16, 17, 18, 19], all of them introduce redundancy, additional activity and additional power consumption. Significant progress has also been made to limit this power increase by special designs like BISER [20], RAZOR [12], DF-DICE [21] and others. However, these schemes do not target the clock gated phase, which is actually in most cases a larger time period a state must be retained than the clocked phase.

This paper presents a technique to protect especially this phase, but it can also be used for fault detection in the clocked phase. For an efficient and automated implementation, basic cells are developed, which can be used in the same way as scan elements for an automated integration. A lightweight observation tree can be synthesized for fault location, indicating which register may be corrupted and if a restart after clock gating is required. Small extensions allow to reuse the scheme for clocked online fault detection and off-line test.

The next section introduces the basic register design with power and fault-tolerance aspects taken into account. Section 3 describes the error detection and correction trees with some quantitative analysis. Section 4 extends the basic registers for the clocked phase. Section 5 investigates the LEON design as part of a multi-core processor system, analyzes the effort to introduce the proposed error detection scheme, and compares it to error detecting registers from the literature.

II. Low power SEU-detection at gate level

Modern low power system design has to address both leakage and switching current. Techniques for leakage control are beyond the scope of this paper, techniques for switching current control can be summarized by avoiding anything which introduces unnecessary circuit activity. To some extent, this paradigm contradicts the commonly used design for test and design for fault tolerance and robustness techniques.

a) Scan elements: The level sensitive scan design technique (LSSD) [22] is considered as especially robust against timing variations. Figure 1 shows the level sensitive $L1/L2$ shift register latch which is controlled by one system clock and two test clocks A and B.

In the most straightforward way, the $L1$ latch is used for data storing at input D controlled by clock CLK, and the input

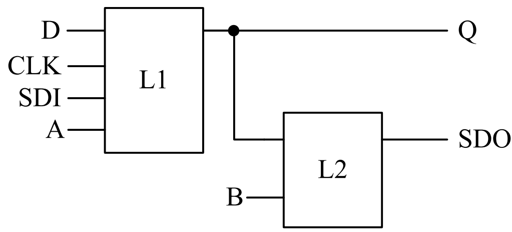

Fig. 1. Shift register latch

”scan data in” is controlled by the two non-overlapping clock signals A,B. Hence, the latch $L2$ is only used for implementing the shift mode and introduces drawbacks with respect to power consumption. It increases the active area and this way the leakage current, and it adds to the load of the $L1$ latch, thereby increasing weighted switching activity. For this reason, power conscious design prefers partial scan. In pipelined circuits without feedback lines, partial scan does not reduce testability [23], and in the Cell processor for instance, only around $1/4$ of the latches are scannable [24].

Also in system mode, latches have to be controlled by a non-overlapping clock scheme. The simplest case is seen in figure 2. The times τ_1^l and τ_1^h are the low resp. high phases of CLK_1, τ_2^l and τ_2^h are these phases of CLK_2, and CLK_1 and CLK_2 are never high at the same time.

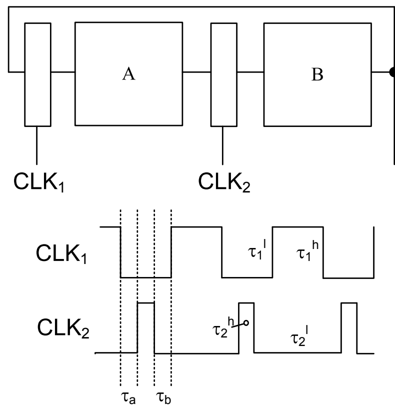

Fig. 2. Non-overlapping clock scheme

As these latches are available anyway they can be combined to so called $L1/L2^*$ latches to implement a single SRL (Figure 3). Now, the $L2$ latch is not any more redundant and does not increase the active area and the leakage current.

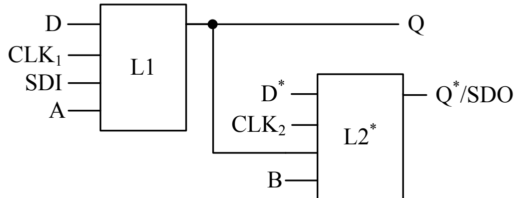

Fig. 3. $L1/L2^*$ shift register latch

b) Robust and fault tolerant registers: Hardening flip-flops and registers can be employed at all design levels, however, in most cases they have significant impact on both leakage and switching current. Layout techniques related to cell and transistor sizing, diode insertion or adding capacitances are effectively employed at physical level [25, 26]. All of them increase both switching and leakage current. At transistor and gate level, redundant memory elements are introduced, and a C-element or a voter guarantee a stable output. Most prominent representative of this class of designs is the BISER (Built-In Soft Error Correction) [3], and many predecessors and variations have been published [17, 14, 10, 13, 15, 18, 19, 12].

Basic principle of all of them is to add two memory elements. One of these memory elements may be a C-element or an additional voter is required (Figure 4).

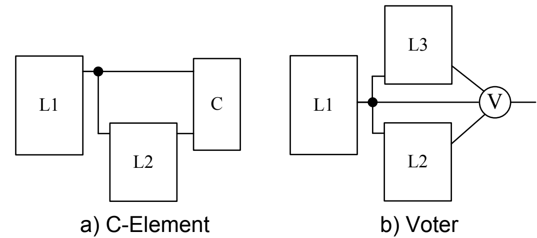

Fig. 4. Principle of robust latch design

If the $L2$ latch is used for scan design anyway, the overhead of these structures in area, power and even delay is limited. However, if for power reasons an $L1/L2^*$ technique has to be used, a direct implementation is not possible. Finally, if for power reasons partial scan design is employed, the overhead in area and power of the technique in figure 4 may reach up to 700%. This aspect is not solved by protecting registers by Hamming code [27].

c) SEU detection: Now we show that implementations with much less impact on power, delay and area can be found if fault correction and fault tolerance at register level is not re-

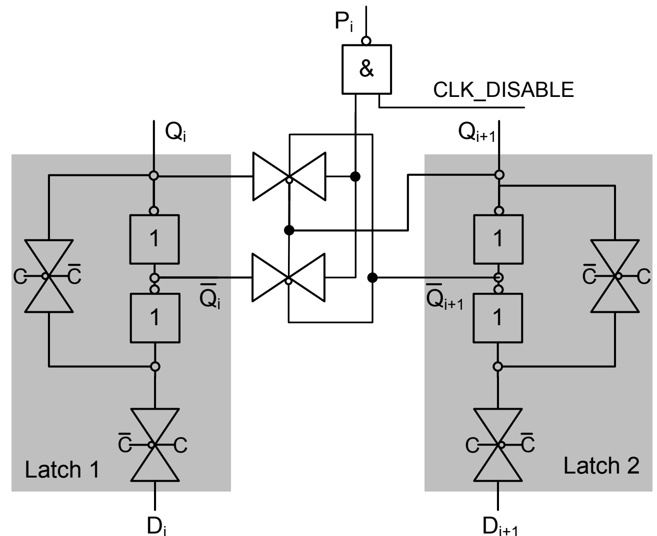

Fig. 5. Parity computation between 2 latches

quired and only fault detection is needed. Instead of using three memory elements we compute just a simple parity signal for a 8 or 16 bits register. The parity tree is gated by the inverted clock gating signal to reduce switching activity during operation. The parity logic itself is integrated into the latch design, and allows synthesis just by abutment. The final routing of the parity lines does not differ from any scan chain routing. First we describe how to handle the leaves of this structure. Figure 5 shows the schematic of the parity computation of 2 latches.

With these parity pair latches (PPL) a register is formed like in figure 6. Since the parity signals have to be amplified anyway the additional impact of gating the tree is just the difference between an inverter and a NAND gate. For the complete tree only the PPL and an XOR cell are required as seen in figure 6.

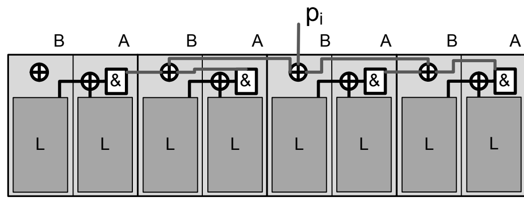

Fig. 6. Parity tree, consisting of two cell types

With this design, each cell consists of just two half XOR gates, and at most two wires cross each cell. The critical path is just 4 inverters, 1 NAND and 3 pass transistors which is less than three times the delay through a latch, and in the same range as any of the double latch solutions mentioned above. Basically, 2 different types of cells are required, and the register synthesis is not more complex than any scan synthesis. By placing two of the registers face to face we design error detecting 16 bit registers by just one additional XOR-gate (Figure 7).

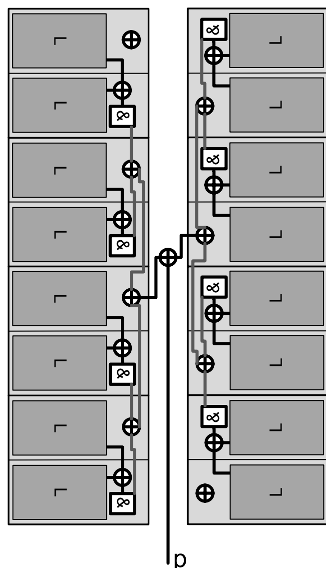

Fig. 7. Error detecting 16 bit register

III. SEU IDENTIFICATION AT MODULE LEVEL

The error information of the basic registers has to be passed to module's top level, stored in a register and will be used for error handling. If only fault detection is required in order to restart computations, a simple XOR tree may connect all the registers.

More effort is needed, if we want to detect also multiple errors, or if fault location is required.

In embedded memories, transparent online test can be used for error location [28, 29]. The transparent test technique for static and dynamic memory arrays from [28] will be adapted to random logic in this paper. The modulo-2 address characteristic of a bit-oriented memory is computed by a bit-wise XOR of the addresses of those memory cells which contain a 1 (Figure 8).

Memory correct

addr	data
000	0
001	0
010	1
011	1
100	0
101	1
110	0
111	0

010
011
⊕ 101
C_c = 100

Memory faulty

addr	data
000	0
001	0
010	1
011	1
100	0
101	0
110	0
111	0

010
⊕ 011
C_f = 001

Fig. 8. Modulo-2 address characteristic

The characteristic has some substantial benefits compared with signature analysis and error correcting codes [30, 31]. Single error location is especially easy, assume c_{ref} is the characteristic of the correct memory content, and c_f is the incorrect one of a single-bit error. The bit-wise computation of $add_f = c_{ref} \oplus c_f$ provides the address of the erroneous bits Double errors are always detected, and the overall aliasing probability is 2^{-N}, where N is the number of address bits. If all the registers of the module are numbered starting from r_1 to r_{N-1}, each parity bit p_i gets a unique address i, and an overall characteristic could be computed sequentially bit by bit like in figure 8. The bit $p_0 = 0$ is not used, as address 0 does not contribute to error detection.

However, the sequential computation is only appropriate for memory arrays, but in random logic with gated clock a combinational logic is required. In a straight-forward way this is implemented by a tree whose leaves are the N bit binary words $p_i \wedge i$, and each node represents bit-wise XOR. Figure 9 illustrates this for 7 registers and parity bits $p_1, \ldots, p_7$.

The number of vertices in the tree of Figure 9 is $2^{N+1} - N$, due to address 0 not being used. As the root node has not to be connected two times there are

$$N \cdot (2^{N+1} - N - 1) \tag{1}$$

point-to-point connections. A significant amount of hardware can be saved if only significant bits are passed between the levels (Figure 10).

In level $k \in \{0, \ldots, N-1\}$ there are 2^k vertices and each vertex $v_{k,l}$, $l \in \{0, \ldots, 2^k - 1\}$ corresponds to 2^{N-k} ad-

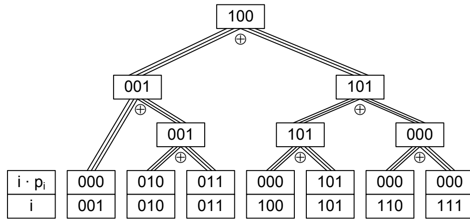

Fig. 9. Non-optimal computation of modulo-2 characteristic

dresses. $c_{k,l} = (c_{k,l,N-1}, \ldots, c_{k,l,0})$ is the output of vertex $v_{k,l}$. Since the k most significant bits $(a_{N-1}, \ldots, a_{N-k+1})$ of the addresses of vertex $v_{k,l}$ are identical, the result bits $(c_{k,i,N-1}, \ldots, c_{k,i,N-k+1})$ only depend on the parity of the 2^{N-k} leaf vertex register-parities:

- $(c_{k,l,N-1}, \ldots, c_{k,l,N-k+1}) = (0, 0, \ldots, 0)$ if the parity is 0
- $(c_{k,l,N-1}, \ldots, c_{k,l,N-k+1}) = (a_{N-1}, \ldots, a_{N-k+1})$ if the parity is 1

We do not need to compute this vector of k bits. Instead, the generation of the information can be deferred to the successor in level $k-1$ and we just compute and forward the parity:

$$p_{k,l} = p_{k+1,2\cdot l} \oplus p_{k+1,2\cdot l+1}$$

The $N-k-1$ bits $(c_{k,l,N-k-2}, \ldots, c_{k,l,0})$ are computed from the characteristics of the predecessors of $v_{k,l}$:

$$c_{k,l,j} = c_{k+1,2\cdot l,j} \oplus c_{k+1,2\cdot l+1,j}$$
$$j \in \{N-k-2, \ldots, 0\}$$

Finally, $c_{k,l,N-k-1}$ corresponds to the highest address bit that distinguishes the two predecessors of $v_{k,l}$. From our previous observations, we know that we can derive any most significant bit of the characteristic from the parity bits $p_{k,2\cdot l}$ and $p_{k,2\cdot l+1}$. Here, we know that address bit $N-k-1$ of predecessor $v_{k,2\cdot l}$ is 0, and 1 for $v_{k,2\cdot l+1}$ respectively. It follows:

$$\begin{aligned} c_{k,l,N-k-1} &= (0 \wedge p_{k+1,2\cdot l}) \oplus (1 \wedge p_{k+1,2\cdot l+1}) \\ &= p_{k,2\cdot l+1} \end{aligned}$$

Therefore, each vertex on level k has 2 connections to read each parity of the 2 predecessors and $2\cdot(N-k-1)$ connections to read the characteristic of the predecessors. Hence, the number of input connections on level k is $2^k \cdot 2 \cdot (N-k)$.

Since the least significant parity bit in any level does not contribute to the overall characteristic, we do not need to compute

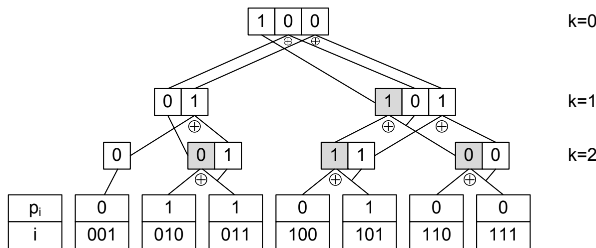

Fig. 10. Optimal tree organization

this bit (see Fig. 10). The overall number of point to point connections is:

$$\sum_{k=0}^{N-1} 2^{k+1} \cdot (N-k) - N$$

which is easily transformed into

$$2^{N+2} - 3 \cdot N - 4 \tag{2}$$

and is significantly less than formula (1).

IV. Online detection and offline test

The structure presented so far may also be used with the clock signals enabled, if we extend the register presented in section 2 to allow for local error detection. For an LSSD structure, as treated here, the so called GRAAL [32] structure was proposed, which extends every system latch with a shadow latch and compares the states of both latches (Figure 11).

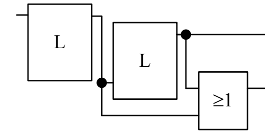

Fig. 11. GRAAL

The basic idea of the approach presented here is not to compare the register content but the parities (Figure 12).

One additional latch per register stores the parity bit already computed for the characteristic computation. If the parity latch is operated with the same clock as the register, the critical path is elongated and the frequency is minimally reduced. This disadvantage can be eliminated if CLK' is, as in conventional "time borrowing", derived from CLK with a delay, that matches the critical path through the parity tree. After the parity has been computed, the stored value is compared to the one continuously computed by the XOR-tree. Therefore one additional XOR-gate generates the error information at register level. This error signal can be used for the treatment of errors in the same way as for the RAZOR approach.

The error detecting design scheme presented so far is especially appropriate for partial scan design, and can be reused to support offline testing. The characteristic at module level increases the observability of all of the latches and compresses the test response.

In [33] it has been shown, that this extreme response compaction to a modulo-2 characteristic or even a parity bit does not affect fault coverage, if appropriate test patterns are applied.

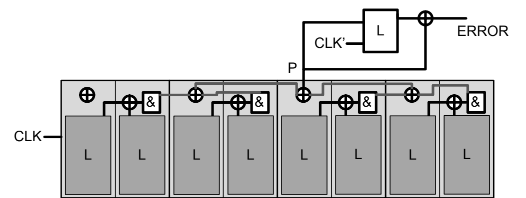

Fig. 12. Local error detection with low delay overhead

V. Experimental Results

In this section, we study the impact of the presented error location technique on embedded processor cores for Multi-processor System on Chip (MPSoC) such as LEON3. We compare the technique presented here with the error masking BISER flip-flop presented in [3] and the error detecting RAZOR shadow latch presented in [12]. For each technique, the number of transistors to implement all the required logic is estimated and compared.

LEON3 is a very modular core that allows to selectively include components such as FPU, MMU, Cache, memory controller, etc.. The configuration of LEON3 evaluated here has about 16k memory elements in the random logic. The memory elements are clustered into 2047 registers of up to 8 bits. The latches used here are of the transmission gate type and require 8 transistors to implement. We do not take into account the overhead for clock generation and distribution.

Table I shows the overhead calculation for the proposed scheme. Column *# Trans* gives the number of transistors per gate of type *Gate Type*. Column *# Gates* lists the number of gates of each type and column *Total* the associated total number of transistors. Each register of 8 bits consists of 4 PPL and 3 separate XOR gates as shown in Figure 6. The number of XORs required to implement the characteristic computation is computed using the considerations in section 3 and matches the synthesis results exactly. A simple comparator for the reference characteristic and a faulty characteristic is implemented in a straight forward way. The last section of Table I concerns the overhead incurred for the online detection as presented in section 4. In order to get a core-level signal that a failure has occurred, all the $ERROR$ lines are aggregated at core level by an OR-tree. As online error detection is optional, we give the total transistor count for both variants of the presented scheme.

Tables II and III give the same information for RAZOR and BISER. For RAZOR we have counted the number of transistors in the implementation presented in [12]. Each latch in the design is implemented in this way. Again, we need a simple OR-tree to provide a core-level signal but this time over all latches instead of just the registers. The BISER scheme has been designed for error masking instead of error detection. This way, the overhead for the OR-tree is avoided. The implementation of BISER used here is taken from [3]. More recent variations [19] provide additional features such as aging sensors, but also increase the overhead.

	Gate Type	# Trans.	# Gates	Total
Per register	PPL	24	4	96
	XOR	6	3	18
	Total			114
All registers		114	2047	233358
Characteristic	XOR	6	4072	24432
Comparator 11bit	Latch	8	11	88
	XOR	6	11	66
	OR	4	10	40
	Total			194
Online detection	Latch	8	2047	16376
	XOR	6	2047	12282
	OR tree	4	2046	8184
	Total			36842
Overall (w/o online)				257984
Overall (w online)				294826

TABLE I

Transistor count for complete online detection

	Gate Type	# Trans.	# Gates	Total
Per latch	INV	2	11	22
	XOR	6	1	6
	OR	4	1	4
	MUX	4	1	4
	Transm. Gate	2	3	6
	AND	4	1	4
	Total			46
Per register		46	8	368
All registers		368	2047	753296
OR tree	OR	4	16376	65504
Overall				818800

TABLE II

Transistor count for a RAZOR implementation

	Gate Type	# Trans.	# Gates	Total
Per latch	Non-scan latch	8	2	16
	Scan latch	20	2	40
	C-Element	6	1	6
	INV	2	5	10
	AND	4	1	4
	OR	4	1	4
	Total			80
Per register		80	8	640
Overall		640	2047	1310080

TABLE III

Transistor count for a BISER implementation

From the analysis it is obvious that the overhead of the presented technique is lower by an order of magnitude compared to the established methods, which are based on duplication and triplication. Compared to the presented technique with online error detection, a design that uses RAZOR incurs 177% more transistors and the same design with BISER even requires 344% additional transistors. If online error detection is not required, RAZOR has 217% and BISER has 407% more transistors.

The lower number of transistors has direct impact on both the silicon area and most important on the associated leakage current. Furthermore, there is only insignificant impact on the switching activity since the NAND-gate built into the PPL avoids any superfluous switching activity in the characteristic computation tree.

VI. Conclusion

In this paper, we have presented an error detection and location method that adapts the benefits of error correcting coding used in memory arrays to memory elements in random logic. The

method is specifically targeted towards clock gated low-power designs, where the latches have to retain their states during long periods of time.

The method is able to easily locate the register affected by a soft error. It consist of simple cells which may easily be integrated similar to the scan design flow. The resulting structure can be efficiently reused for offline and general online testing. Compared to other soft error resilient memory elements, the presented structure requires up to 77% less overhead.

VII. Acknowledgment

This work has been supported by the DFG Project Realtest under grant Wu245/5-1.

References

[1] J. Ziegler, H. Curtis, H. Muhlfeld, C. Montrose, B. Chin, M. Nicewicz, C. Russell, W. Wang, L. Freeman, P. Hosier, *et al.*, "IBM experiments in soft fails in computer electronics (1978-1994)," *Journal of Research*, vol. 40, no. 1, 1998.

[2] B. Gill, M. Nicolaidis, and C. Papachristou, "Radiation Induced Single-Word Multiple-Bit Upsets Correction in SRAM," in *11th IEEE International On-Line Test Symposium (IOLTS)*, 2005.

[3] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *IEEE Computer*, vol. 38, no. 2, pp. 43–52, 2005.

[4] A. Nieuwland, S. Jasarevic, and G. Jerin, "Combinational Logic Soft Error Analysis and Protection," *12th IEEE International On-Line Test Symposium (IOLTS)*, pp. 99–104, 2006.

[5] M. Nicolaidis, "Design for soft error mitigation," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 405–418, 2005.

[6] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.

[7] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," *32nd Conference on Design Automation (DAC), San Francisco, CA, USA, June 12-16, 1995*, pp. 242–247, 1995.

[8] M. Pedram and J. Rabaey, *Power Aware Design Methodologies*. Kluwer Academic Publishers Norwell, MA, USA, 2002.

[9] S. Borkar, "Thousand core chipsa technology perspective," in *Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, 2007*, 2007, pp. 746–749.

[10] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology," *Nuclear Science, IEEE Transactions on*, vol. 43, no. 6, pp. 2874–2878, 1996.

[11] T. Monnier, F. Roche, and G. Cathebras, "Flipflop hardening for space applications," *Intl. Workshop on Memory Technology*, 1998.

[12] D. Ernst, N. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, *et al.*, "Razor: a low-power pipeline based on circuit-level timing speculation," *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 7–18, 2003.

[13] S. Krishnamohan and N. Mahapatra, "A highly-efficient technique for reducing soft errors in static CMOS circuits," *Computer Design: VLSI in Computers and Processors (ICCD04). Proceedings. IEEE International Conference on*, 2004.

[14] M. Nicolaidis, "Design for mitigation of single event effects," in *IOLTS*. IEEE Computer Society, 2005, pp. 95–96.

[15] A. Drake, A. KleinOsowski, and A. Martin, "A Self-Correcting Soft Error Tolerant Flop-Flop," *12th NASA Symposium on VLSI Design, Coeur dAlene, Idaho, USA, Oct*, pp. 4–5, 2005.

[16] A. Goel, S. Bhunia, H. Mahmoodi, and K. Roy, "Low-overhead design of soft-error-tolerant scan flip-flops with enhanced-scan capability," *Proceedings of the 2006 conference on Asia South Pacific design automation*, pp. 665–670, 2006.

[17] M. Omaña, D. Rossi, and C. Metra, "Latch susceptibility to transient faults and new hardening approach," *IEEE Trans. Computers*, vol. 56, no. 9, pp. 1255–1268, 2007.

[18] M. Fazeli, A. Patooghy, S. Miremadi, and A. Ejlali, "Feedback Redundancy: A Power Efficient SEU-Tolerant Latch Design for Deep Sub-Micron Technologies," *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 276–285, 2007.

[19] M. Zhang, T. Mak, J. Tschanz, K. Kim, N. Seifert, and D. Lu, "Design for Resilience to Soft Errors and Variations," *Proceedings of the 13th IEEE International On-Line Testing Symposium*, pp. 23–28, 2007.

[20] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and 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, pp. 1368–1378, 2006.

[21] R. Naseer and J. Draper, "The DF-dice storage element for immunity to soft errors," *Proceedings of the 48th IEEE International Midwest Symposium on Circuits and Systems*, 2005.

[22] E. Eichelberger and T. Williams, "A logic design structure for LSI testability," *Proceedings of the 14th design automation conference*, pp. 462–468, 1977.

[23] H.-J. Wunderlich and A. Kunzmann, "An analytical approach to the partial scan problem," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 1, no. 2, pp. 163–174, 1990.

[24] O. Takahashi, S. R. Cottier, S. H. Dhong, B. K. Flachs, and J. Silberman, "Power-conscious design of the cell processor's synergistic processor element," *IEEE Micro*, vol. 25, no. 5, pp. 10–18, 2005.

[25] M. R. Choudhury, Q. Zhou, and K. Mohanram, "Design optimization for single-event upset robustness using simultaneous dual-vdd and sizing techniques," in *2006 International Conference on Computer-Aided Design (ICCAD'06), November 5-9, 2006, San Jose, CA, USA*, 2006, pp. 204–209.

[26] R. Garg, N. Jayakumar, S. Khatri, and G. Choi, "A design approach for radiation-hard digital electronics," in *Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*, 2006, pp. 773–778.

[27] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy," *Proceedings of the 15th Symposium on Integrated Circuits and Systems Design*, p. 95, 2002.

[28] S. Hellebrand, H.-J. Wunderlich, A. A. Ivaniuk, Y. V. Klimets, and V. N. Yarmolik, "Efficient online and offline testing of embedded drams," *IEEE Trans. Computers*, vol. 51, no. 7, pp. 801–809, 2002.

[29] S. Boutobza, M. Nicolaidis, K. M. Lamara, and A. Costa, "A transparent based programmable memory bist," in *11th European Test Symposium (ETS 2006), 21-24 May 2006, Southhampton, UK*, 2006, pp. 89–96.

[30] T. Damarla, C. Stroud, and A. Sathaye, "Multiple error detection and identification via signature analysis," *Journal of Electronic Testing (JETTA)*, vol. 7, no. 3, pp. 193–207, 1995.

[31] R. W. Hamming, "Error Correcting and Error Detecting Codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, 1950.

[32] M. Nicolaidis, "Graal: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies," *IEEE International Test Conference (ITC07).*, pp. 1–10, 21-26 Oct. 2007.

[33] M. Elm, S. Holst, and H.-J. Wunderlich, "Scan chain organization for extreme response compaction," *Submitted for publication*, 2008.