Logic Masking for SET Mitigation Using Approximate Logic Circuits

A. Sánchez-Clemente, L. Entrena, M. García-Valderas C. López-Ongil,
Electronic Technology Department
Universidad Carlos III de Madrid
Madrid, Spain
{ajscleme, entrena, celia, mgvalder}@ing.uc3m.es

Abstract—**Logic masking approaches for Single-Event Transient (SET) mitigation use hardware redundancy to mask the propagation of SET effects. Conventional techniques, such as Triple-Modular Redundancy (TMR), can guarantee full fault coverage, but they also introduce very large overheads. Alternatively, approximate logic circuits can provide the necessary flexibility to find an optimal balance between error coverage and overheads. In this work, we propose a new approach to build approximate logic circuits driven by testability estimations. Using the concept of unate functions, approximations are performed in lines with low testability in order to minimize the impact on error coverage. The proposed approach is scalable and can provide a variety of solutions for different trade-offs between error coverage and overheads.**

Keywords-Single-Event Transient; Soft error; Error detection and correction; Approximate circuit; testability

I. INTRODUCTION

Soft errors have become a concern for high reliability applications in the nanometric era. The most important cause of soft errors are energetic particles, such as alpha particles, neutrons or heavy ions. When a particle hits a circuit node, the collected charge produces a transient voltage perturbation at the node. If the affected node is a memory cell or latch, the voltage perturbation can produce a change of state, known as an SEU (Single-Event Upset). Otherwise, a transient voltage pulse known as an SET (Single-Event Transient) may propagate through the logic and eventually be latched in one or several memory elements.

SEUs have traditionally received more attention as they can directly affect the state of a memory cell or latch. However, with reduced transistor sizes, increased device densities and higher clock frequencies, SETs are also becoming very relevant. Actually, SETs are more likely to occur because they can affect any combinational node. However, the probability that an SET provokes an error is reduced due to three masking factors: logic masking, latch window masking and electrical masking. Logic masking occurs when the SET is not located in a sensitized path and thus cannot be propagated to a sequential element. Latch-window or timing masking occurs when the transient reaches the memory elements outside of their latching time window. Electrical masking occurs when the transient is attenuated by subsequent logic gates until it is eventually filtered out.

SET mitigation approaches make use of these masking effects as a means to eliminate the propagation of transient voltage pulses. Electrical masking is exploited by using larger transistors or nodal capacitances [1],[2]. Latch-window masking is exploited by using time redundancy based on latching data at different times [3],[4]. Finally, techniques based on logic masking are proposed in [5],[6],[7],[8] and [13].

In this work we focus on logic masking techniques. The well-known Duplication With Comparison (DWC) or Triple-Modular Redundancy (TMR) techniques can be considered as extreme examples of logic masking. These techniques have full error detection or error masking capabilities, but they introduce more than 100% and 200% area overhead, respectively. In practice, designers need flexibility to find an optimal balance between the level of protection and the area, power and performance requirements. For this purpose, partial logic masking can be used. Partial error detection or correction can be achieved by using DWC or TMR with a partial replication of the logic [5][6]. While this may be effective, it is still based on a conventional technique, has limited scalability and may introduce performance penalties. More flexible alternatives are based on the use of implications [13] or approximate logic circuits [7],[8].

An approximate logic circuit is a circuit that performs a possibly different but closely related logic function, so that it can be used for error detection or error masking where it overlaps with the original circuit. In [7] the authors propose a method for concurrent error detection of permanent stuck-at faults using approximate logic circuits. The approximate logic circuits are built by selectively converting minterms from the off-set or on-set of logic functions into don't-cares. The selection is based on the local observability of the inputs of the node and it applies only to a subset of node types. They also propose the use of local observability don't-cares to explore a richer number of approximations. However, this does not guarantee correctness of the solution and they have to use a SAT algorithm to detect incorrect approximations and discard them. Although the authors claim the approach is scalable, they do not provide results for different approximations. The use of approximate logic circuits for SET mitigation is proposed in [8]. This approach is based on trial-and-error and is only tested for some very small subcircuits.

In this work we propose a new approach to build approximate logic circuits that is based on the concept of unate functions. This approach allows approximating a logic circuit at any line or variable. Since the error coverage properties of the resulting approximated circuits are directly related with the testability of the lines selected for approximation, testability

This work has been supported in part by the Spanish Government under contract TEC2010-22095-C03-03.

978-1-4673-2085-6/12/$31.00 ©2012 IEEE

measures can be used to properly guide the circuit approximation process, taking into account both global controllability and observability. The proposed method can be used for both error detection and correction, it is scalable and can provide a variety of solutions for different trade-offs between error coverage and overheads. On the other hand, it does not enforce the use of a particular synthesis tool and can be used in combination with any commercial synthesis tool.

The remaining of the paper is as follows. Section II introduces the use of approximate logic circuits in error detection and correction approaches. Section III describes the circuit approximation approach based on the concept of unate functions. Section IV describes the line selection approach. Finally, Section V shows the experimental results and Section VI presents the conclusions of this work.

II. Error Detection and Correction using Approximate Circuits

Error detection and correction approaches using approximate logic circuits are illustrated in Fig. 1. These are similar to DWC or TMR, respectively, with the main difference that approximate logic circuits are used instead of exact copies of the original logic circuit. Errors in the Checker can be covered by using a Totally Self-Checking Checker with two-rail error coding. Similarly, errors in the Voter can be covered by using a triple voting schema.

When approximate circuits are used, the outputs of the original circuit and the approximate circuits may differ even in the absence of faults. This imposes some restrictions on the approximate circuits. For error detection, the checker must only be active when the outputs of the approximate circuit and the original circuit are expected to coincide. For error correction, it must be ensured that at least one of the approximate circuits produces the same output as the original circuit for any input vector. These restrictions can be satisfied by using implication-based approximate logic circuits.

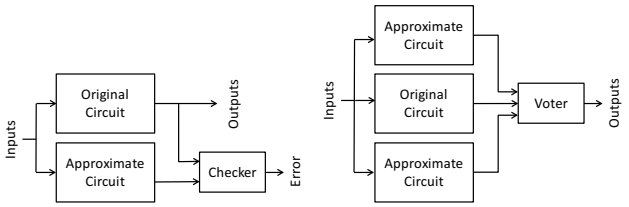

Figure 1. Error detection (a) and error correction (b) schemas using approximate logic circuits

A logic function F_0 is a 0-approximation of a function F if $\overline{F_0} \Rightarrow \overline{F}$. Conversely, $F \Rightarrow F_0$, i.e., $F \subseteq F_0$. Similarly, a logic function F_1 is a 1-approximation of a function F if $F_1 \Rightarrow F$, i.e., $F_1 \subseteq F$. F_0 and F_1 will be generally referred to as implication-based approximate logic functions of F and they satisfy the relationship $F_1 \subseteq F \subseteq F_0$.

The error mitigation capabilities of implication-based logic functions can be explained with the diagram shown in Fig. 2 [8], where the on-sets of the original and the approximate logic functions are represented. Error detection actually consists in checking the implication relationship. When F_0 (F_1) is used in an error detection schema, the protected area is $\overline{F_0}$ (F_1). Therefore, the error detection coverage that can be achieved with this approach is limited to the probability of the off-set of F_0 or the on-set of F_1, which in turn are always smaller than the off-set and the on-set of F, respectively. Thus, this approach can only provide a high error coverage for functions with a large, highly probable off-set or on-set.

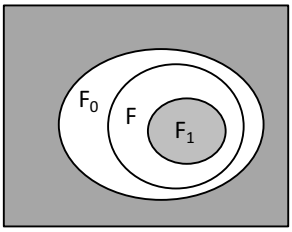

Figure 2. Graphical representation of the relationship among the original and the approximate functions

For error correction, the relationship $F_1 \subseteq F \subseteq F_0$ ensures that at least F_0 or F_1 produce the same output as F for any input vector. For an input vector $v \in F_1$, $F_1 = F = F_0 = 1$ and for an input vector $v \in \overline{F_0}$, $F_1 = F = F_0 = 0$. Thus, if there is an error in anyone of the three functions, the other two will still be correct and the error will be masked. This is indicated as a shadow area in Fig. 4. The area between F_0 and F_1 is the unprotected area. For any input vector in this area, F_0 and F_1 differ, so the result of voting will be the value of F. This is correct if no error exist in F. However, if there is an error in F, it will not be masked. Note that errors may also be produced in F_0 or F_1. Errors in F_1 are always masked in the on-set of F and errors in F_0 are always masked in the off-set of F.

A simplified error correction approach is proposed in [8] that uses the error masking function

$$O = F_1 + F F_0$$

instead of a majority voting function. The error masking capabilities of this function are similar to the voting function except in the case of an error in F_1 for an input vector in the off-set of F_0. Thus the off-set of F_0 is also unprotected. As in the error detection case, this approach is only useful for functions with a large, highly probable on-set.

With the error masking approach using a majority voting function, there is no theoretical limitation for the error coverage. Then, the goal is to find implication-based approximate circuits with an optimal balance between area/power reduction and soft error coverage, and without compromising performance. To this purpose, the circuit approximation approach must be strongly linked to estimations of the probability of the resulting unprotected input combinations. We propose the use of testability measures to estimate this probability. Note that these consider both global controllability and observability, as opposed to the approach in [7] were circuit approximation is driven by local observability.

Testability analysis techniques have been developed mainly for stuck-at faults. If a stuck-at fault in a line has low testability, it means that there are few input vectors that can test the fault. Then, a good approximate circuit can be built by

assigning the line to a constant value. The probability of the unprotected area will be the probability of the input vectors which test the stuck-at fault. In the next section we will show how implication-based approximate circuits can be obtained by making a line constant using the concept of unate functions.

III. CIRCUIT APPROXIMATION

Our algorithm to compute approximate circuits is based on the concept of unate functions. This concept can be defined for logic functions as well as for logic networks, so that the resulting circuit approximation approach can be applied on both.

A function $F(x_1, x_2, \cdots, x_n)$ is positive (negative) in x_i, if there exists a sum-of-products expression in which x_i, does not appear complemented (uncomplemented). F is unate in x_i if it is positive or negative in x_i; otherwise it is binate in x_i. This concept can be extended to the multiple-output function performed by a network in the following way. The function F_j at output O_j of a network is positive (negative) in a line x if there exists a sum-of-products expression of the function in which x does not appear complemented (uncomplemented). Note that the line x is considered as a primary input of function F_j in the previous definition. The multiple-output function F performed by a network is positive (negative) in x if all output functions are positive (negative) in x.

By definition, a positive function F in x_i can be expressed as

$$F = x_i g(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) + h(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$$

and a negative function can be expressed as

$$F = \overline{x_i} g(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) + h(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$$

In network terms, a function F is positive (negative) in x if all paths from x to the primary outputs have an even (odd) number of inversions. In such a case, we say that line x has an even (odd) parity. If a function F is binate in x, then there are at least two paths from x to the primary outputs with different parities, and we say line x has no parity.

The following theorem determines the theoretical foundations of our approach to obtain implication-based approximate functions.

Theorem. Let F be a positive (negative) function in x. Let F_0 and F_1 be the functions obtained by making $x = 1(0)$ and $x = 0(1)$, respectively. Then, F_0 is a 0-approximation of F and F_1 is a 1-approximation of F.

Demonstration. If F is a positive function in x, then there exist functions g and h so that F can be expressed as

$$F = xg + h$$

Then, $F_0 = g + h$ and $F_1 = h$. $\overline{F_0} \Rightarrow \overline{g}, \overline{h} \Rightarrow \overline{F}$. Thus, F_0 is a 0-approximation of F. Analogously, $F_1 \Rightarrow h \Rightarrow F$, and F_1 is a 1-approximation of F. The demonstration is similar for the case of a negative function.

Note that x can be any line in a network, and F_0 and F_1 are the cofactors of F with respect to x. This theorem also holds for the case of multiple-output functions.

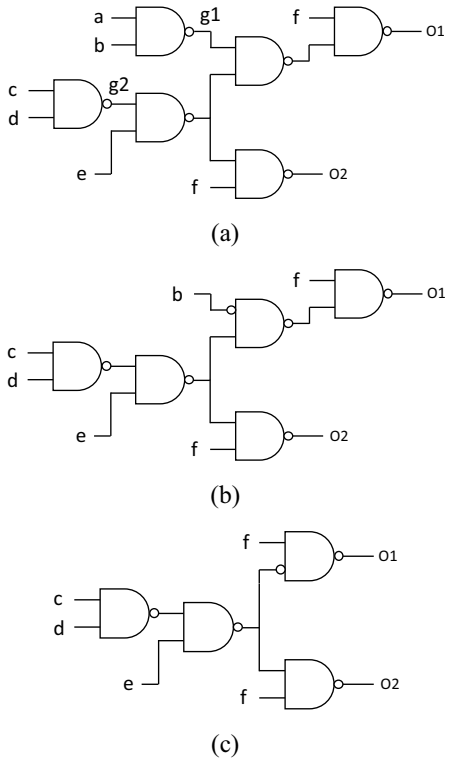

Figure 3. Example circuit (a), 1-approximation obtained with a = 1 (b) and 0-approximation obtained with g1 = 1 (c)

Example 1. In the circuit shown in Fig. 3(a), $O1$ is negative in a and positive in $g1$, or alternatively, a has odd parity and $g1$ has even parity. Thus, the theorem can be directly applied on these lines. By making $a = 1$, the 1-approximation shown in Fig. 3(b) is obtained. Similarly, a 0-approximation is obtained by making $g1 = 1$ (Fig. 3(c)). Note that the latter can also be obtained by making $a = 0$, because $a = 0$ implies $g1 = 1$.

The above theorem provides the basic means to build approximate logic circuits in an easy manner. Any line with parity can be selected and tied to a logic constant to build a 0- or 1-approximation. Although it cannot be directly applied to lines with no parity, it is possible to approximate a function in this kind of lines by applying a simple transformation to the circuit. Any network can be made unate except for possibly the primary inputs, by duplicating the gates and splitting the lines with no parity into two subsets with even and odd parities respectively [14]. This temporarily creates a larger circuit that allows the application of the previous theorem. Duplications that are not removed after approximation can be removed later on by resynthesizing the approximate logic circuits.

The parity of a gate can be computed by traversing the network from primary outputs to primary inputs. At the same time, duplications are created for lines with no parity using the following algorithm:

a) For a primary output O, $parity(O) = 0$
b) For an input wi of a gate G with output wo
$parity(wi) = parity(wo) \oplus Inv(G)$
c) On a multiple fanout point, the parity of the stem wire

is the parity of the branches if all branches have the same parity. Otherwise, the input gate is duplicated creating two stems with 0 and 1 parities, respectively. Each fanout is then linked to the stem with the same parity.

Inv(X) denotes the inversion value of *X*. Gates with no inversion value (e.g. XOR gates) can be transformed temporarily into their AND-OR equivalent for the purpose of approximation.

Note that the approximation of a line does not change the parity of the lines in the circuit, because parities are computed from outputs to inputs. Thus, the parity calculation algorithm needs to be executed only once.

Example 2. Fig. 4(a) shows a simple circuit with an XNOR gate where all lines except the output have no parity. The circuit after gate transformation and duplication is shown in Fig. 4(b). The assignment $a = 0$ can be applied on $g1$ ($g1'$) to obtain a 0(1)-approximation of the circuit. After simplification of the logic, the approximate circuits are shown in Fig. 4(c). For the 90 nm technology library used in the experiments, the 0- and 1-approximations require only 52% and 67% of the area of the original circuit, respectively.

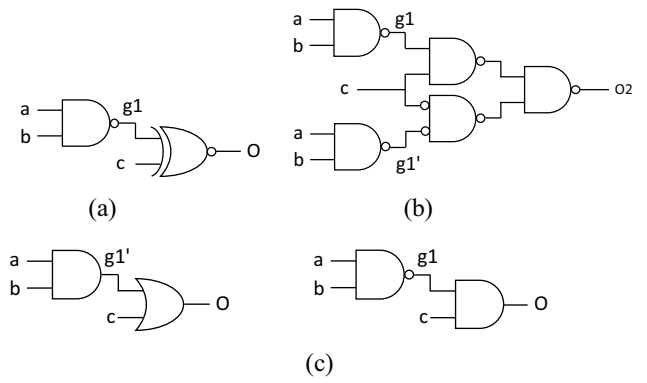

Figure 4. Example circuit including an XNOR gate (a), transformed circuit (b) and 0- and 1-approximations obtained with a = 0.

The approximate circuit obtained using gate duplication may be larger than the original one in some cases. This may happen if, after duplication, only one of the copies is approximated and it cannot be further merged with the other copy in the final synthesis step. This problem is solved by making a synthesis-aware approximation. Rather than substituting the selected line x by a constant 0 or 1, an appropriate gate is inserted at line x with a don't-care side input. Depending on the value assigned to the don't-care, the approximation is taken or not. This decision is passed to the synthesis tool, where it can be properly handled to obtain the best solution.

Example 3. In the circuit shown in Fig. 3(a), lines c and $g2$ have no parity. In order to consider possible approximations in these lines, the lines with no parity are duplicated as shown in Fig. 5(a). In the so transformed circuit, all lines except inputs c, d, and e, have parity. Note that only the stem input lines have no parity; the inputs to gates $g2$ and $g2'$ do have parity and can be approximated. However, since they have different parities, they cannot be set to the same value for the same approximate circuit. For instance, a 0-approximation can be obtained by making $c = 1$ at the input of $g2$ and a 1-approximation can be obtained by making $c = 0$ at the input of $g2'$. Following the above considerations, the approximations are made by inserting an OR or an AND gate with a don't-care side input (Fig. 5(b)). This approach enables the synthesis tool to evaluate the convenience of making the approximation or not, and choose the best implementation. For this example, the approximate circuits when the don't-cares are taken are shown in Fig. 5(c). Since these are more complex than the original circuit, the synthesis tool should not consider them and produce the original circuit again. On the other hand, consider the case where c is approximated to either 0 or 1 as needed. In this case, the 0-approximation is obtained with $c = 1$ at $g2$ and $c = 0$ at $g2'$, and the 1-approximation with the opposite assignments. The resulting approximate circuits are simpler than the original one, as shown in Fig. 5(d). Alternatively, a similar result could have been obtained by relaxing the condition that all outputs must have the same parity. This would produce approximate circuits that have a combination of positive and negative outputs, but they can still be used in error detection and correction schemas.

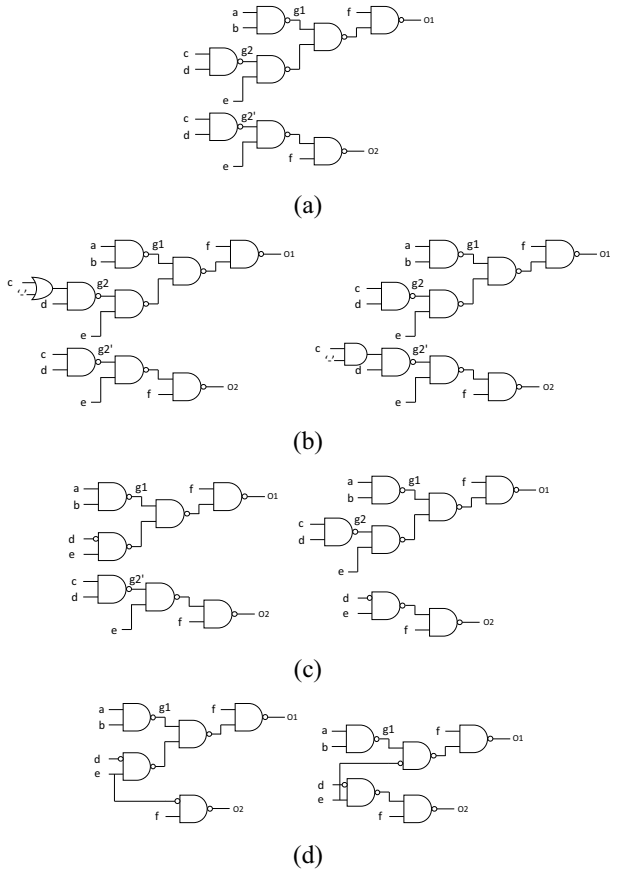

Figure 5. Original circuit after duplication of lines with no parity (a), 0- and 1-approximations obtained with c = 1 (b), 0- and 1-approximations when don't-cares are taken (c), 0- and 1-approximations with c = 0 and c = 1

Since circuit approximation is achieved by setting lines to constant values, the approximate circuits are guaranteed to be smaller or equal than the original circuit and to consume less power, except for possibly the case of duplicated gates. However, this case is eliminated by the final logic synthesis step. Similarly, the approximate circuits are also guaranteed to have smaller or equal critical path delay than the original circuit.

IV. Selection of Lines for Approximation

The proposed circuit approximation approach can be easily guided by testability measures, as proposed in Section II. The idea is to select the lines with the lowest testability and build approximate circuits by making these lines constant.

The lines with the lowest testability can be determined by testability analysis. However, while testability analysis algorithms, such as SCOAP [9], are useful to evaluate testability needs or guide ATPG algorithms, they may be inaccurate or take a high computational effort to estimate the probability of testing a single fault when there is reconvergent fanout. Alternatively, sufficient probability estimations can be obtained by fault simulation with a random set of input vectors, or better with a typical workload for the target circuit, if available. Then the probability of testing a stuck-at fault is estimated as the fraction of input vectors that test the fault.

Even in the case that accurate testability estimations are known, it is still difficult to estimate the final fault coverage. Unprotected areas are different for each approximate circuit. Moreover, the unprotected areas produced by each line approximation often overlap. In our approach we select a testability threshold and approximate the lines whose testability is below the selected threshold. Different approximations are obtained by varying the testability threshold.

V. Experimental Results

A. Experimental setup

The following paragraphs describe the methodology used for the experimental results. Experiments were conducted with a group of benchmarks from the LGSynth93 set. For each benchmark, the steps required to perform the experiment are described below.

First, the circuit was made unate using the algorithm described in Section III. Then, the testability of each line was estimated by stuck-at fault simulation using the parallel fault simulator HOPE [10] with a set of 10,000 random input vectors. Next, approximate circuits were generated according to these testability estimations. The testability threshold can be specified as desired for different trade-offs between area and fault coverage. Finally, the original circuit and the 0- and 1-approximate circuits were synthesized separately in order to avoid sharing the logic among them, and embedded in a higher-level entity that included output registers and a voter. Synthesis was performed with Synopsys for the SAED 90 nm technology library [11].

The final error coverage was estimated by SET fault injection using the AMUSE system [12]. AMUSE is an emulation-based fault injection system that provides very high performance with accuracy close to electrical simulation by using a quantized representation of time, voltage and delays. The experiments were performed for pulses of 300 ps and for pulses with the size of the critical path. The latter can be considered as an estimation of stuck-at fault coverage for stem lines and was obtained for the sake of comparison with other approaches. In all cases, the high performance of AMUSE allowed a very extensive fault injection campaign which included all possible SETs for every gate, clock cycle and quantized time instant.

B. Results

Table 1 shows the results of the tests performed. The first column contains the name of benchmarks, while the second one shows the size of the benchmarks after synthesis, measured as the number of logic gates. The testability threshold used in each case is shown in the third column. Next, the results of the proposed approach are shown, including the area overhead due to 0- and 1-approximations respectively, the coverage estimated for stuck-at faults and the coverage estimated for SET pulses of 300 ps. The last two columns show the area overhead and coverage of the solution proposed in [7], in those cases where data are available.

In order to show the scalability of the proposed technique, two benchmarks (frg2 and i10) have been tested with a larger set of thresholds. Figures 6 and 7 show the variation of error coverage w.r.t. the area overhead for frg2 and i10, respectively. The area overhead is the sum of the areas of both approximate circuits. The labels show the testability threshold applied in each case.

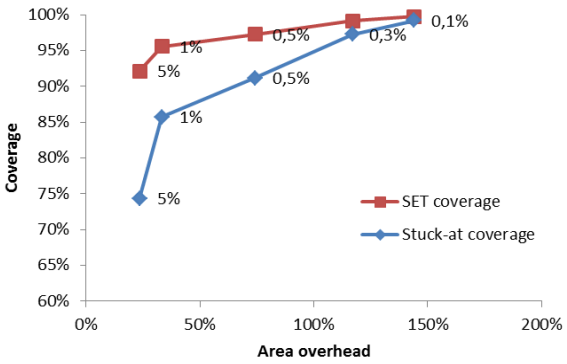

Figure 6. Scalability of technique for frg2 benchmark

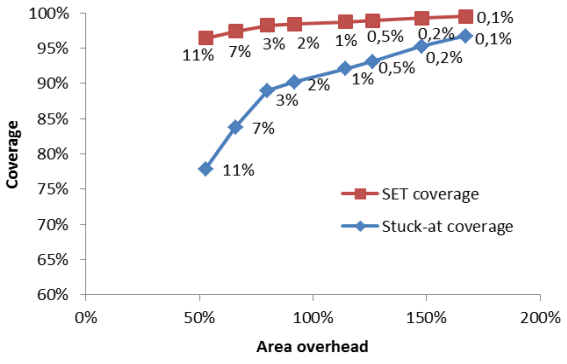

Figure 7. Scalability of technique for i10 benchmark

VI. CONCLUSIONS

As SET effects are becoming more relevant, efficient SET mitigation techniques are needed. Partial logic masking techniques based on approximate logic circuits can provide the flexibility designers need in order to find a solution that satisfies the reliability requirements with reduced overheads.

The proposed approach allows to approximate logic circuits with a direct relationship to the testability of the lines selected for approximation, so that testability measures can be used to properly guide the construction of approximate logic circuits. Further improvements are possible by considering logic transformations that expose additional lines with low testability.

REFERENCES

[1] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," IEEE Trans. Computer-aided Design, vol. 25, pp. 155–166, Jan. 2006

[2] J. M. Cazeaux et al., "On transistor level gate sizing for increased robustness to transient faults," in Proc. Intl. On-line Testing Symposium, pp. 23–28, 2005.

[3] M. Nicolaidis. "Time redundancy based soft-error tolerance to rescue nanometer technologies. Proc. VLSI Test Symposium, pp. 86-94, 1999

[4] H. Yu, M. Nicolaidis, L. Anghel. "An effective approach to detect logic soft errors in digital circuits based on GRAAL". Proc. Int. Symposium on Quality Electronic Design (ISQED), pp. 236 – 240, 2009

[5] K. Mohanram, N. A. Touba. "Cost-effective approach for reducing soft error failure rate in logic circuits". Proc. Int. Test Conference (ITC), pp. 893-901, 2003

[6] K. Mohanram, N. A. Touba. "Partial error masking to reduce soft error failure rate in logic circuits". Proc. 18th IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 433-440, 2003.

[7] M. R. Choudhury, K. Mohanram. "Approximate logic circuits for low overhead, non-intrusive concurrent error detection". Proc. Design Automation and Test in Europe (DATE), pp. 903-908, 2008

[8] B. D. Sierawski, B. L. Bhuva, L. W. Massengill. "Reducing Soft Error Rate in Logic Circuits Through Approximate Logic Functions". IEEE Trans. on Nuclear Science, vol. 53, no. 6, Dec. 2006.

[9] L. H. Goldstein. "Controllability/Observability Analysis of Digital Circuits". IEEE Trans. on Circuits and Systems, vol. CAS-26, pp. 685-693, 1979

[10] H. K. Lee, D. S. Ha. "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits". IEEE Trans. on Computer- Aided Design of Integrated Circuits and Systems, Vol. 15, No. 9, pp. 1048-1058, September 1996.

[11] "SAED 90nm Generic Library". Synopsys Armenia Educational Department, [Online]. Available: http://www.synopsys.com/ Community/UniversityProgram

[12] L. Entrena, M. García-Valderas, R. Fernández-Cardenal, A. Lindoso, M. Portela, C. López-Ongil. "Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection". IEEE Trans. on Computers, vol. 61, no. 3, pp. 313-322, Jan. 2012

[13] K. Nepal, N. Alves, J. Dworak, R. I. Bahar. "Using implications for Online Error Detection". Proc. Int. Test Conference (ITC), 2008.

[14] H. Kim and J. P. Hayes. "Realization-Independent ATPG for Designs with Unimplemented Blocks". IEEE Trans. Computer-aided Design, Vol. 20, No 2, Feb. 2001

TABLE 1. AREA AND COVERAGE RESULTS FOR LGSYNTH93 BENCHMARKS

			Error masking				CED [7]	
Benchmark	**Gates**	**Th. (%)**	***Area A0 (%)***	***Area A1 (%)***	***Stuck-at Coverage (%)***	***SET Coverage (%)***	***Area (%)***	***Coverage (%)***
alu1	31	7	61.29	61.29	91.35	91.35	-	-
alu2	267	0.5	47.19	79.03	83.68	96.49	-	-
c432	133	8	36.84	40.60	82.98	96.42	-	-
		3	56.39	75.94	87.81	97.43		
c880	213	2	13.62	53.05	81.17	95.87	-	-
		1	84.51	65.73	93.62	98.60		
c5315	1097	3	74.38	73.93	94.98	99.19	-	-
c7552	1007	10	52.83	89.77	94.56	99.30	-	-
cmb	31	0.1	0	41.94	93.85	95.95	32	98
cordic	141	0.05	37.59	10.64	95.37	97.57	28	82
dalu	577	3	0.17	3.81	80.72	95.77	21	80
		0.5	62.39	62.05	94.82	98.91		
des	2826	1	70.42	68.22	82.94	98.09	-	-
frg2	666	0.5	55.41	18.77	91.18	97.29	30	80
		0.1	73.87	70.12	99.23	99.77		
i2	89	1	0	4.49	96.26	98.03	5	84
		0.01	0	25.84	96.80	98.32		
i8	656	0.5	67.07	67.23	81.74	95.80	-	-
i10	1346	2	47.10	44.73	90.21	98.42	36	81
		0.2	75.41	72.36	95.29	99.27		
s444	96	3	66.67	67.71	92.31	96.15	-	-
term1	208	0.3	19.71	21.15	97.09	98.78	15	71
unreg	83	15	40.96	19.28	85.37	87.16	-	-
		6.3	75.90	42.17	91.33	93.91		
x1	381	1	15.49	20.21	93.90	97.17	36	68
		0.1	41.73	41.73	95.01	97.69		