Technology Mapping for Reliability Enhancement in Logic Synthesis

Zhaojun Wo and Israel Koren
Department of Electrical and Computer Engineering
University of Massachusetts,Amherst,MA 01003
E-mail: {zwo,koren}@ecs.umass.edu

Abstract— Reliability enhancements are traditionally implemented through redundancies at the system level or through the use of harden-cell-designs at the circuit level. Reliability is commonly ignored during the logic synthesis step. A major reason for this is the fact that constructing a cost function to measure sensitivity to faults at the logic synthesis level is complex. The work presented in this paper addresses one important aspect of synthesis for high reliability. It focuses on the problem of mapping a technology independent circuit to a technology specific one, using gates from a given library, with Fault Sensitivity [3] as an optimization metric. We believe that the difficulty in obtaining accurate metrics of fault sensitivity at the technology independent level makes it hard to optimize at this level, thus technology dependent mapping offers a direct method to improve reliability. In this paper, we present a concept named "effective fault area" for mapping onto library gates. Along with this concept, we adopt a Markov-model based analytical method to accurately estimate fault sensitivity during mapping with a low computational overhead. Several benchmark results show that the average reliability improvement is about 20.7% at the cost of 12.1% increase in delay.

I. Introduction

Transient faults are the most common type of failures in digital systems. The rate of transient faults due to high-energetic particles is called Soft Error Rate (SER). Alpha-particles and neutrons constitute more than 97% of the total high-energetic particle hits [1]. Reducing the effects caused by these two types of particles is desirable in order to reduce the rate of transient faults, and thus, increase the system reliability.

A typical digital system contains three parts: combinational logic, latches and memory units. In [2], the SER for these three parts with the technology scaling trend is predicted. It is concluded that the SER for memories stays roughly the same with technology scaling, the SER for latches increases linearly with scaling, while the SER for logic increases exponentially with scaling. As a result, decreasing the sensitivity to faults of combinational logic is highly desirable.

The need for high reliability circuit design is being actively addressed at several levels. At the system level, spatial or temporal redundancy techniques are used to improve system reliability. At the circuit level, robust cells and latch designs are studied. At the device level, work is being done to reduce the peak voltage generated by particle hits. To the best of our knowledge, no research on improving reliability at the logic synthesis level has been published mainly due to the lack of metrics for fault sensitivity at this level.

In this paper, we attempt to establish a metric for fault sensitivity based on a previously introduced "Fault Sensitivity (**FS**)" metric [3]. A concept of "**effective fault area**" is presented and a corresponding fault sensitivity cost function is described. Then, a mapping algorithm targeting the reduction of the effect of transient faults in combinational logic is presented. Based on our experimental results on the ISCAS85 benchmark suits, we conclude that about a 20.7% reduction in fault sensitivity can be achieved at the expense of a 12.1% increase in delay compared with minimal delay solutions.

II. Fault Model

A. Traditional Device to Transistor Level Particles-hit Model

Alpha-particles and neutrons are the main sources for high energy particle hits. Several transient models for faults have been proposed for high energetic particles. The hit mechanism for alpha-particles or neutrons can be illustrated in Fig.1. Only source and drain of active region are sensitive to particles hit (Fig.1(a)). The high energetic particles cross through PN junctions, ionizing movable charges. The charges are collected by the junctions (Fig.1(b)). As a result, leakage current is collected by the substrate. Transistor level fault simulation methods [4] are commonly used to analyze the efect of the particle hits. An input stimulus vector is related with primary input probability(p), charge injection volume(q), charge injection time(r), and injection node(N) [4]. For each stimulus vector, a transient current source, denoted by I_{inj}, is added at drain or source node in a transistor (Fig.1(c)). The current model for alpha-particles and neutrons is given in (1) [2, 5]

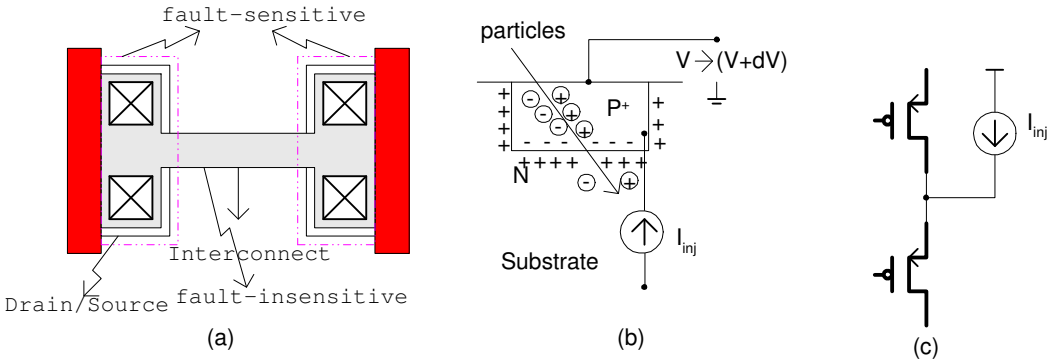

Fig. 1. (a) The sensitive and insensitive area of a PMOS node (b) particle hit on the drain of PMOS transistor (c) The circuit model for particle-hit

$$I_{inj} = \begin{cases} \frac{Q_{inj}}{\tau_2 - \tau_1}(e^{-\frac{t}{\tau_2}} - e^{-\frac{t}{\tau_1}}) & \textit{for } \alpha\textit{-particles} \\ K\frac{Q_{inj}}{\tau}\sqrt{\frac{t}{\tau}}e^{-\frac{t}{\tau}} & \textit{for neutrons} \end{cases} \quad (1)$$

where Q_{inj} is injected charge, τ_1, τ_2, τ are time constants and K is technology dependent parameter.

B. Fault Sensitivity

In [3], a Fault sensitivity metric was presented for measuring the potential effect of faults. It is defined as:

$$FS \equiv \sum_{Y_j \in PO} \sum_{i=1}^{N} A_i * \{\frac{1}{p \cdot q \cdot r} \sum_{p,q,r} E_{(i,Y_j)}(p,q,r)\} \quad (2)$$

where A_i is the sensitive area of node i. i.e., the area of the source and drain active regions since only these are sensitive to particle hits. $E_{(i,Y_j)}(p,q,r)$ is the outcome of a fault injection experiment and is given by

$$E_{(i,Y_j)}(p,q,r) = \begin{cases} 1 & \textit{if the injection into node } i \textit{ results} \\ & \quad \textit{in a fault at primary output } Y_j \\ 0 & \textit{otherwise} \end{cases} \quad (3)$$

where p *is the number of input combinations in the experiments*, q *is the number of particle injection levels considered*, r *is the number of time instances at which faults are injected*, and N *is the number of nodes in the circuit.* The unit of **FS** is area.

Simulation-based methods are generally used to calculate **FS** [4]. Alpha-particles and neutrons are the main sources for high energy particle hits. A total of $p*q*r*N$ simulation runs are needed at the transistor level. A transient current source, denoted by I_{inj}, is added at the drain or source node in a transistor. The current model for alpha-particles and neutrons is shown in (1).

A circuit simulator such as SPICE is called for each simulation run making this approach too time consuming even for relatively small circuits. In our technology mapping scheme, we adopt the above transient circuit models for basic gates such as NAND2. Using exhaustive SPICE simulations we extract the effective fault areas (we will introduce it later.) for these basic gates. As shown below, the **FS** metric is a simple linear combination of the input signal patterns and the effective fault areas.

C. Effective Fault Area

To calculate FS from (2), we have to perform charge injection simulations at the transistor level. In Fig.2(a), the Nand2 gate has to be simulated for 5 different nodes. We notice that for static CMOS design, the NMOS and PMOS components are fully separated by the output node. It is also noticed that a particle hit in NMOS will always lower the voltage of the injection node. Similarly, a hit in PMOS will always increase the voltage of the injection node. Equivalently speaking, considering the gate output, a hit in NMOS can *only* generate a negative glitch while a hit in PMOS can *only* generate a positive glitch. These effects motivate us to make the following equivalence: If all the NMOS part hits (I_{inj} from node to ground) are averaged to one digital pulse hit at the output (the negative pulse width equals to the sum of the injection effects 1,2 and 3 in Fig.2.), and all the PMOS part hits (I_{inj} from VDD to the node) are averaged to one digital pulse hit in the output (the positive pulse width equals to the sum of injection effects 4 and 5.). We further make a simplifying assumption that the output of a gate has two types of "glitches" (see Fig.2(b)) corresponding to positive and negative glitches due to particle hits in PMOS and NMOS, respectively. Each type of glitch is unit-width pulse. There is a "*weight*" assigned to each type of glitch (the unit of this weight is area) and we call this weight "**effective fault area**".

The **effective fault area** A_g^0, A_g^1 for gate g is defined

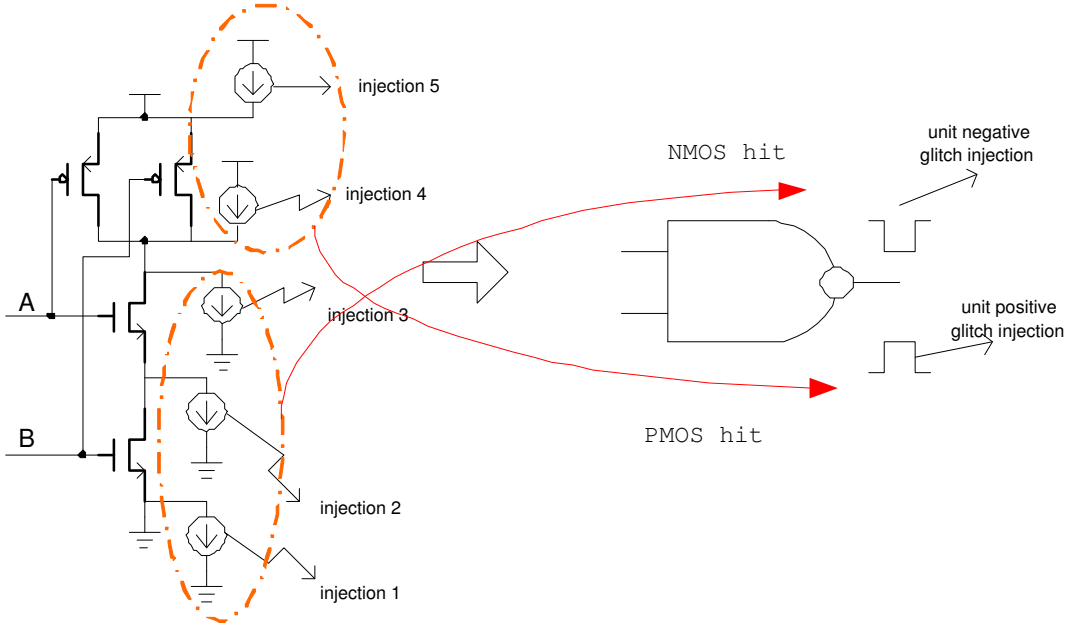

Fig. 2. (a)Five injection experiments needed in transistor-level simulation (b)Two unit width glitches from output, "fault sensitive area" equals the fault-sensitivity value

as:

$$\begin{cases} A_g^0 = \sum_{i \in NMOS} \{\frac{1}{p.q.r} \sum_{p,q,r} E_{(i)}(p,q,r)\} * A_i \\ A_g^1 = \sum_{i \in PMOS} \{\frac{1}{p.q.r} \sum_{p,q,r} E_{(i)}(p,q,r)\} * A_i \end{cases} \quad (4)$$

Eq(4) allows us to move from transistor level to gate level simulation. Given $A_g^{0,1}$, we inject digital pulses only at the gate's output and find out if a primary output error happens. The factors q, r, N in (2) are no longer needed. As a result, the number of total possible simulation runs is reduced. This greatly reduces the simulation time, not only because fewer nodes are simulated but also because fast logic simulators can be used instead of circuit simulators. $A_g^{0,1}$ can be pre-characterized by exhaustive simulations for given library gates.

Using the effective fault area definition, eq(2) can be rewritten as:

$$FS = \sum_{Y_j \in PO} \sum_{g_i} (P(g_i \rightarrow Y_j)\{P(g_i = 1) * A_{g_i}^1 + P(g_i = 0) * A_{g_i}^0)\} \quad (5)$$

where $P(g_i \rightarrow Y_j)$ is the probability that a glitch propagates from g_i's output to the primary output Y_j and $P(g_i = 0)$, $P(g_i = 1)$ is the probability that $g_i = 0$ and $g_i = 1$, respectively. The later are called signal probabilities.

D. Analytical fault sensitivity (FS) computation

We intend to use eq(5) as the cost function for our low Fault Sensitivity mapping algorithm. The question is how to effectively compute the two probabilities during each mapping decision. We use a zero gate delay assumption to simplify the computing of the probability. It means that the glitches generated by the difference of the gate delay is ignored. Under this assumption, for any Directed Acyclic Graph (DAG)-like structure, the signal probability for each node is determined by its logic expression only.

D.1 Signal probability computation

The difficulty in determining the signal probability for DAG-like structure is due to the existence of fanout re-convergent paths [6]. Re-convergent fanout paths introduce functional dependency and statistical correlations among the signals. The Markov-based chain-correlation approach is a suitable one for our task. We adopt the MCP [6] algorithm to compute each node's signal probability. The computing complexity is $O(NS)$ where N is the number of total gates and S is the number of wires.

D.2 Minimal fault sensitivity tree-mapping

From eq(5), each gate has its own contribution to **FS**. Our goal is to select an optimal combination of gates from a given library. Optimal DAG-cover-mapping is a NP-hard problem. However, if we limit the DAG to tree-cover-mapping, it always has an optimal solution [7].

Suppose a netlist can be split into trees (Fig.3). Our goal is to optimize FS in eq(5). For each gate g_i, we need to compute the path probability $P(g_i \rightarrow Y_j)$. This is difficult because of the presence of fanout re-convergent paths. We avoid computing $P(g_i \rightarrow Y_j)$ directly. Suppose a tree T_i is the target tree to be optimized. We denote by

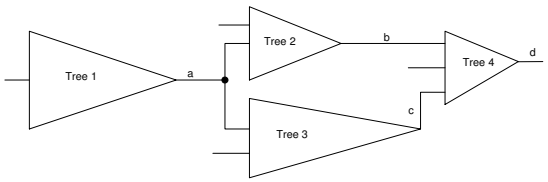

Fig. 3. Tree structure of a netlist

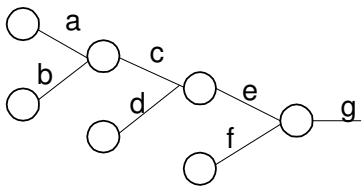

Fig. 4. FS computation for a tree structure

Minimal Fault Sensitivity mapping Algorithm{
INPUT: Technology independent synthesized logic function and library.
OUTPUT: Mapped netlist with minimal Fault Sensitivity (FS).
1. Decompose to nand2/inv based subject graph.
2. Split DAG at each multi-fanout node, decompose subject graph to trees.
3. DFS-sort trees of subject graph, assign levels to these trees.
4. For each sorted trees, find out each possible match within the tree.
5. Calculate each tree's primary input's probability and signal correlations.
6. Compute each node's signal probability and each pair's signal correlation using MCP algorithm.
7. Compute each node's path probability to the tree primary output.
8. Use dynamic-programming to obtain min_FS mapping in this tree.
9. Repeat steps 4-8 until all the trees are mapped.
10. compute final FS.
}

Fig. 5. Minimal FS mapping algorithm

FS_{T_i} the Fault-Sensitivity measured for the tree T_i. We have the following claim:

Lemma : For a tree-cover-mapping optimizing a global fault sensitivity FS, reducing the local fault sensitivity FS_{T_i} also reduces FS.

Proof: eq(5) can be re-written as :
$FS = \sum_{Y_j} \sum_{T_i} P(T_i \rightarrow Y_j) * FS_{T_i}$
The logic function from the tree T_i to any primary output Y_j is independent of the mapping of the tree, i.e., the logic function $F(T_i \rightarrow Y_j)$ is fixed. Under our zero delay gate model, $P(T_i \rightarrow Y_j)$ in eq(5) is fixed. Thus, the global fault-sensitivity FS is reduced by reducing FS_{T_i} •

Based on our claim, our cost function FS is further simplified to FS_{T_i} for each individual tree. For example, in Fig.3, the netlist contains 4 trees and therefore,

$$FS = FS_a * P(a \rightarrow d) + FS_b * P(b \rightarrow d) + FS_c * P(c \rightarrow d) + FS_d \quad (6)$$

It is equivalent to optimizing 4 trees ($FS_a \sim FS_d$) individually. Fig.4 is an example of a tree. It no longer has fanout-reconvergent paths for each gate. For example:

$$P(a \rightarrow g) = P(bdf) = P(b).P(d).P(f).C_{b,d}.C_{b,f}.C_{d,f} \quad (7)$$

where $C_{i,j}$[6] is the correlation factor for the two nodes i, j. Fig.5 shows our tree-based minimal fault-sensitivity mapping algorithm.

III. experimental results

To illustrate our mapping procedure, we present an example of a single tree. In Fig.6, all the nine primary inputs are assumed to have equal signal probabilities ($P(i) = 0.5$). Signals "1" and "5" are dependent ($C_{1,5} = 2$). The other signals are mutually independent ($C_{i,j} = 1$). The mapped result is shown using dashed frames.

TABLE I
Fault Sensitivity compared by two methods

Circuit	FS by [4]	FS by (5)	Error
Fig.6	8.0	8.3	3.8%
C17	2.85	2.82	-1.1%
C432	59.0	61.4	4.1%

Table I compares the calculated values of the fault sensitivity for three circuits. Column 2 is the exhaustive transistor-level simulation method [3, 4]. Column 3 is the result of the simulation based on eq(5). The accuracy of out mathod is within 5%.

Our minimal Fault Sensitivity program is written in C. It consists of about 6 thousand lines of codes. A dynamic programming technique similar to SIS [7] minimal area mapping algorithm is followed. We use TSMC 0.18um static CMOS library in our experiments. Our experimental library is smaller than the commercial library. It contains only 12 cells. We first use SIS to perform technology independent logic synthesis and technology decomposition to a Nand2/Inv based subject graph. Then the subject graph is converted to a DAG structure and the minimal Fault Sensitivity program is called. Finally, we use our simulation-based scripts [4] to verify the improvement of the metric. We ran this program on the circuits from the ISCAS85 benchmark suits. The results are shown in Table II.

In order to evaluate our mapped results, we generated three sets of results and compared with each other. Each set has three metrics: FS, area and delay. Set 1 is the result of our minimal FS synthesis. Sets 2 and 3 are min-

TABLE II
FS TREE-MAPPING RESULTS FOR ISCAS85 BENCHMARKS

Benchmarks	minimal FS			minimal Area			minimal delay		
	FS	area	delay	FS	area	delay	FS	area	delay
C17	2.82	28	0.2	2.82	28	0.2	2.82	28	0.2
C432	61.4	1147	8.0	69.0	1069	8.6	83.5	1213	6.8
C499	204.5	2128	17.8	209.3	1920	20.7	254.6	2132	16.8
C880	136.8	1786	16.3	148.0	1648	18.5	151.6	1901	13.6
C1908	190.0	2238	19.5	208.3	2058	22.3	252.2	2275	18.6
C2670	297.4	4233	26.0	327.4	3861	29.6	403.7	4478	23.9
C3540	549.8	11477	51.3	661.5	10273	63.7	812.1	13059	41.4
C5315	685.4	7876	43.3	727.9	7268	57.2	893.5	9023	38.0
C6288	1281.8	14173	101.5	1401.5	13521	143.0	1683.3	16174	89.4

Bench-marks	minFS VS minA (%)			minFS VS minDly (%)			minA VS minDly (%)		
	ΔFS	ΔA	ΔDly	ΔFS	ΔA	ΔDly	ΔFS	ΔA	ΔDly
C17	0	0	0	0	0	0	0	0	0
C432	−10.9	7.3	−7.0	−26.4	−5.4	17.6	−16.9	−11.9	26.5
C499	−2.3	10.8	−14.0	−19.7	−0.2	6.0	−17.8	−10.0	23.2
C880	−7.6	8.4	−11.9	−9.8	−6.0	19.9	−2.0	−13.3	36.0
C1908	−8.8	8.7	−12.6	−24.7	−1.6	4.8	−17.5	−9.5	19.9
C2670	−9.2	9.6	−12.2	−26.3	−5.5	8.8	−18.8	−13.8	23.8
C3540	−16.9	11.7	−19.5	−32.3	−12.1	23.9	−18.6	−21.4	53.8
C5315	−5.8	8.4	−24.3	−23.3	−12.9	14.1	−18.6	−19.6	50.5
C6288	−8.5	4.8	−29.0	−23.9	−12.4	13.5	−16.7	−16.4	60.0
average	−7.9	7.8	−14.5	−20.7	−6.2	12.1	−12.0	−12.9	32.6

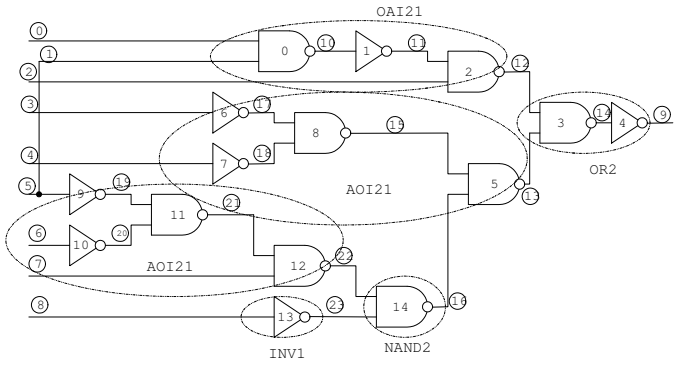

Fig. 6. A mapped result for a tree. Signal "1" and "5" are dependent

imal area and minimal delay results provided by SIS [7], respectively. We use a static-timing analysis method to compute the critical path delay where the unit delay is ns.

The three sets of results are then compared, two at a time, in the right half of Table I. We first compare the metrics for the minimal FS solutions with these of the minimal area ones. On the average, there is a 7.9% decrease in fault sensitivity, 14.5% reduction in delay, at the expense of 7.8% increase in area. We next compare the minimal FS solutions with the minimal delay solutions. On the average, there is a 20.7% reduction in FS, 6.2% reduction in area at the expense of 12.1% increase in delay. We finally compare the minimal area solutions with the corresponding minimal delay solutions. On the average, the minimal area solutions have a 12.0% reduction in FS, a 12.9% reduction in area, but a 32.6% increase in delay.

IV. CONCLUSION

In this paper, a tree-based technology mapping algorithm targeting Fault Sensitivity is presented. A new concept of "**effective fault area**" is presented, which allows to use an analytical model for calculating Fault Sensitivity. Dynamic programming is adopted to implement

the algorithm. Experimental results indicate that about 20.7% improvement in Fault Sensitivity can be achieved at an extra cost of 12.1% in delay compared to the minimal delay solutions.

References

[1] J.F. Ziegler, "Cosmic Ray Soft Error Rates of 16-Mb DRAM Memory Chips," *IEEE JSSC*, Vol.33, NO.2, pp. 246–252, Feb 1998.

[2] P. Shivakumar, "Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic," *Proc. of ICDSN*, pp. 389–398, June 2002.

[3] A. Maheshwari, I. Koren and W. Burleson, "Techniques for Transient Fault Sensitivity Analysis and Reduction in VLSI Circuits," *IEEE International Symposium on DFT*, pp. 597-602, 2003.

[4] M. Singh and I. Koren, "Fault Sensitivity Analysis and Reliability Enhancement of Analog-to-Digital Converters," *IEEE Trans. on VLSI Systems*, pp. 839–852, Nov. 2003.

[5] G.C. Messenger, "Collection of Charge on Junction Nodes from Ion Tracks," *IEEE Trans. on Nuclear Science*, pp.2024-2031, 1982.

[6] H. Li, J.K. Antonio and S.K. Dhall, "Fast and Precise Power Prediction for Combinational Circuits," *Proceedings of IEEE Symposium on VLSI*, pp.254–259, Feb 2003.

[7] E.M. Sentovich, "SIS: A System for Sequential Circuit Synthesis ," *ICCAD*, 1992.