Dynamic Stress Tests for "Narrow Metal Imperfections" in Full Scan Circuits

Tech. Rep. 96-06 Dept. of Computer Science, SUNY, Buffalo

Vinay P. Dabholkar Sreejit Chakravarty
Department of Computer Science
State University of New York at Buffalo
{*vpd,sreejit*}@*cs.buffalo.edu*

Abstract

Dynamic stress testing is an important process that is used to improve the reliability of circuits. In this paper, we investigate the problem of computing cyclic sequences which can be used during burn-in of full scan circuits. Using a switching activity model we show that the stress due to electromigration and hot-electron degradation can be accurately modeled. Notwithstanding the difficulty of computing optimal dynamic stress tests, we demonstrate experimentally that efficient heuristics can be developed to compute good dynamic stress tests for full scan circuits.

1 Introduction

Growing size and complexity of VLSI circuits, reduction in feature sizes, narrower interconnection lines and smaller contacts is making production of reliable chips a challenging task. **Stress testing** is an effective method to improve product reliability [1]. To understand this consider Figure 1 where failure probability is plotted against time. This curve is known as the **bathtub curve** [1, 2]. Failure probability is high in the early period of product life. This is known as **infant mortality**. It is explained by the presence of "weak ICs" in the production lots. Such ICs contain "imperfections" which are a consequence of manufacturing defects. "weak ICs" fail soon after passing production testing. During *stress testing* ICs are subjected to stress conditions. The "weak ICs" fail and are not shipped out. This reduces early field failure rate and improves product reliability.

Products under stress fail in different ways. The most appropriate way to stress a product depends upon the failure modes targeted. Stress induced failures are primarily of two kind: (a) **threshold stress failures**; and (b) **cumulative stress failures** [1]. *Threshold stress failures* occur in a specific product sample when the stress level exceeds the threshold strength of the sample. For example, mechanical stress of a brittle material or the temperature at which electronic component begins to malfunction. In *cumulative stress failure*, cumulative effect of repeated low-level stressing causes failures. Although such stresses may be well below the level that would cause immediate product failure, each time they are applied they produce some irreversible changes in the product. After repeated exposure to these stresses, the cumulative change causes the product to fail. Examples of *cumulative stress failures* are failures that occur due to electromigration in a device under high current density, hot-electron degradation, oxide breakdown, fatigue failures of solder joints due to extended thermal cycling etc. [1, 2].

ICs are stressed in a variety of ways such as burn-in, power cycling, temperature cycling, voltage variations, clock variations etc [1]. **Burn-in**, the standard method used in industry for stressing ICs, is to subject ICs to high temperature and high voltage for an extended period [2]. Typical values used are 50 to 250 hours at 70 to 125^{o}C and 6 to 11 Volts [3]. One of a number of scenarios where *burn-in* is used is depicted in Figure 2. This figure, taken from [4], depicts MCM production flow. Highlighted blocks show the places where *burn-in* and testing are performed. *Burn-in* is performed at two places, once on the bare die and then on the assembled module. In addition to MCMs, burn-in is used to eliminate *infant mortality* in other semiconductor ICs like ASICs etc. [5, 1, 2]. *Burn-in* induces cumulative stress failures. There are three kinds of *burn-in*: **Static, Dynamic** and **Monitored**. Each burn-in stresses a class of defects and manufacturing imperfections [3]. In this paper, we will restrict ourselves to dynamic burn-in.

Dynamic burn-in is a process in which devices are continuously activated by changing the inputs at high temperature and subjected to high voltage. This results in high power dissipation, current densities and chip temperature which causes "weak ICs" to fail. **Dynamic stress tests** are the set of vectors that are continuously applied during *dynamic burn-in*. Such vectors should be cyclic since they will be applied repetitively over a long period. Throughout this paper we refer to such tests as **cyclic dynamic stress tests**. Such tests can be applied at high speed since the output is not being monitored.

The objective of this research is to compute good *cyclic dynamic stress tests* . Any measure of goodness of such stress tests should be a function of the effectiveness of the tests to force the latent defects into observable failures. In this paper we restrict ourselves to two failure modes: (a) Failures due to electromigration [6, 2, 7] and (b) device degradation due to hot-carriers [8, 2]. Both the failure modes are important causes of device failures. Traditionally wear out has been the main cause of these failures. But scaling down of the devices is drastically reducing the mean time to failure due to these failure modes. For example, due to narrow multilevel interconnection lines and smaller contacts mean time to failure due to electromigration can be reduced by up to a factor of the seventh power of the scaling factor [2]. Similarly, hot-electron degradation is known to depend exponentially on the scaling factor for channel length [2]. This motivates the computation of *cyclic dynamic stress tests* which will accelerate the "aging" process and force the "weak ICs" to fail due to these failure modes.

In this paper we propose a switching activity based model that correlates switching activity with stressing circuit nodes. It is shown how this model takes into account failure modes such as electromigration and hot-electron degradation. Failure due to electromigration happens if there are very narrow metal interconnects and switching accelerates the procedure forcing the narrow metal impurities into observable failures. Based on switching activity model, problems for computing good cyclic stress tests for dynamic burn-in in full scan circuits are proposed. It is shown that these problems are computationally difficult. However, we demonstrate experimentally that good heuristics can be developed to compute stress tests for dynamic burn-in of full scan circuits.

The only related work known to us is [9]. It presents a study of computing *cyclic stress tests* for *Monitored Burn-in*. However, *monitored burn-in* has several drawbacks. (i) Since, in dynamic burn-in, selection of vectors is not restricted to a test set, higher switching activity can be generated. (ii) It requires a tester throughout the burn-in process. The cost of tying down a tester for a long period could be prohibitively large. (iii) In monitored burn-in the output of test application has to be latched back to scan registers. This could limit the speed and effectiveness of burn-in. In addition to presenting a taxonomy of burn-in problems, it is shown in [9] that most cyclic stress test problems pertinent to monitored burn-in can be solved efficiently. Optimal algorithm and faster heuristics along with experimental results were presented. Constrasting this with the difficulty of computing good dynamic stress tests, one would be tempted to use stress tests computed for *monitored burn-in* during *dynamic burn-in*. An important contribution of this paper is to show that good heuristics for dynamic burn-in give better stress tests than that computed for monitored burn-in.

The paper is organized as follows. Section 2 describes suitability of using the switching activity model to target certain classes of defects. In section 3, notation and a model for computing switching activity in full scan circuits is presented. Measure of goodness used to compare the cyclic sequences is discussed along with the problem definition in section 4. Heuristics for computing stress tests are presented in section 5 and experimental results are discussed in section 6.

2 Switching Activity Model

2.1 Electromigration and Switching Activity

Electromigration is the motion of the ions of a conductor such as aluminium, due to the passage of an electric current through it. A divergence of the ion flux ultimately gives rise to circuit failure. A positive divergence leads to a void in the metal while a negative divergence leads to a buildup of metal called a hillock, which can lead to short circuit to adjacent or overlying metal. Passing high density current continuously will accelerate this process. Hence average switching of a node which involves charging and discharging of output load capacitance is considered a good measure of stressing the circuit node as far as electromigration is concerned [7].

Consider a switch level description of a NAND gate in Figure 3 (a). A $0 \rightarrow 1$ transition charges the parasitic capacitance at the gate output through the pull-up network and a $1 \rightarrow 0$ transition discharges it through the pull-down network. Continuous switching is likely to induce stress in metal interconnections lines as well as in the contacts. Thus average switching activity accurately models stress induced by electromigration.

2.2 Hot Carrier Degradation and Switching Activity

Electrons and holes in the channel and pinchoff regions of a transistor are called hot-carriers because they gain enough energy from the lateral electrical field produced by source-drain voltage. In case of nMOS transistor, electrons gain so much energy that they tunnel into the dielectric (SiO_2) between the gate and the channel. This traps electrons in the SiO_2 which leads to threshold voltage shifts and transconductance degradation. This ultimately leads to device failure. The problem is not as acute in case of pMOS because holes have less mobility than electrons and hence they don't gain much energy.

Figure 3 (a) and (b) show switch level description of NAND and NOR gates respectively. In case of a NAND gate every $1 \rightarrow 0$ transition at the output results in a current through both nMOS transistors. Therefore, higher the switching activity better the stress induced due to hot-carrier degradation in the NAND gate. However, a $1 \rightarrow 0$ transition at the output of a NOR gate may result in current through only one of nMOS transistor. For example, if the inputs change from $(A = 0, B = 0)$ to $(A = 1, B = 0)$ output of the NOR gate changes from 1 to 0 but only transistor with input A conducts. Thus switching activity approximately models the stress induced by hot-electron degradation. In this paper, our primary aim is to model failures due to electromigration. This happens if there are very narrow metal interconnects which is a result of many manufacturing defects. This model can be modified so that it correlates more closely to the hot-electron induced failures.

2.3 Delay Model

Switching activity in combinational circuits depend on gate delays and the sequence of input vector applied. The former is approximated by delay models. There are various delay models such as zero delay model, unit delay model, general delay model etc.

Under the zero-delay model all gates are assumed to have zero delay. The four possible transitions,on application of t_i followed by t_{i+1}, and the corresponding logic levels are : static-zero (s0) $\Rightarrow$ Logic level remains zero; static-one (s1) $\Rightarrow$ Logic level remains one; rising (r) $\Rightarrow$ Logic level changes from 0 to 1; falling (f) $\Rightarrow$ Logic level changes from 1 to 0. For example for a two-input AND gate output behavior under zero-delay model is described by Table 1. Tables for the other gates can be easily derived. The zero delay model neglects glitches at gate outputs and therefore underestimates switching activity. Nevertheless, we chose zero-delay for various reasons. First, it is the most prevalent model used for switching activity computation [10, 11, 12, 13]. Second, the other delay models overestimate switching activity. Thus zero delay model is a pessimistic model. In our case, it is better to use the pessimistic model that underestimates stress due to switching.

3 Switching Activity in Full Scan Circuits

Block diagram of Full Integrated Scan is shown in Figure 4 [14]. C is the combinational part of the circuit and R is the test register where the tests are scanned serially. In dynamic burn-in the output from the combinational circuit is not latched back to the scan chain. Note that with every shift of a vector in the scan chain the inputs to the combinational circuit C change.

3.1 An Example

Consider the scan circuit shown in Figure 5. Inputs x_1 and x_2 are primary inputs while inputs x_3 and x_4 come from scan latches F_1 and F_2 respectively. Switching activity at node g_1 is determined by logic values at primary inputs only. We call such nodes **combinational nodes** or **c-nodes**. A *cyclic stress test* for a *c-node* would consist of two input vectors that will be applied to primary inputs repetitively. For example, node g_1 can be switched by the pair $(00, 11)$ applied to inputs x_1, x_2 respectively. When the number of c-nodes targeted is more than one then the stress test will be, a cycle of length 2 or larger. To illustrate this consider the following example. Let $\{n_1, n_2, n_3\}$ be the three c-nodes targeted. Assume that at most two of the three nodes can be switched by any pair. Let the pair (v_1, v_2) switch $\{n_1, n_2\}$, the pair (v_2, v_3) switch nodes $\{n_2, n_3\}$ and the pair (v_3, v_1) switch nodes $\{n_3, n_1\}$. In this case cyclic sequence (v_1, v_2, v_3) is a better stress test than any of the pairwise sequences.

Switching activity at node g_2 is determined by input from scan latches only. We call such nodes **sequential nodes** or **s-nodes**. A *cyclic stress test* that targets an *s-node* is a vector which can be cycled through the scan chain. For example, the sequence 011 is a stress test for node g_2. As 011 is cycled through the scan chain, inputs x_3 and x_4 take the values: $01 \rightarrow 11$ followed by $11 \rightarrow 10$ followed by $10 \rightarrow 01$. Note that two out of three input transitions switch g_2.

Switching activity at nodes like g_3 and g_4 depend upon both the primary inputs as well as scan latch inputs. We call such nodes **hybrid nodes** or **h-nodes**. A cyclic stress test for h-nodes consists of a vector that is cycled through scan latches and a set of primary input vectors that is applied during scan-in/out process.

3.2 Measure of Goodness of a cycle

Determining measure of goodness of a cycle is an important issue. Ratio of the total number of nodes switched during the application of a cyclic sequence to the length of the cyclic sequence gives the average number of node switched per clock of the cyclic sequence. In [9], average number of nodes switched per clock of the cyclic sequence was used as the measure. This measure has the following drawback. A cyclic sequence may be generating high switching activity in only a few of the targeted nodes but a number of nodes either might not be switching or the switching might be low for a number of nodes. This can lead to high average switching for the cyclic sequence but we contend that this is not a good sequence to stress the circuit.

To illustrate this, consider a hypothetical example consisting of four nodes, viz. $\{n_1, n_2, n_3, n_4\}$. Let C_1 and C_2 be two cyclic sequences each of length 10. Let C_1 generate following switching: $\{n_1 = 10, n_2 = 5, n_3 = 1, n_4 = 1\}$ and let C_2 generate $\{n_1 = 4, n_2 = 4, n_3 = 4, n_4 = 4\}$. Average switching of C_1 per clock is 1.7 while that of C_2 is 1.6. According to the measure used in [9], C_1 is better than C_2. However, it can be seen that in C_1 50% of the nodes have very low activity (10%) and in C_2 100% of the nodes have moderate activity (40%).

This motivates the following question: When can we say that a cyclic sequence generates reasonable activity at a node ? In other words, a **threshold** for good average activity needs to be determined such that nodes with average activity below this *threshold* would be said to have poor switching. Note that in our approach to computing stress tests groups of nodes are considered at a time and the measure of goodness reflects it. Let t be a *threshold*. A cyclic sequence C_1 is better than a cyclic sequence C_2 for a set of nodes S if the number of nodes with average switching activity greater than t for C_1 is more than that for C_2.

In practise, this threshold would depend upon the reliability requirement. Note that there might be inherent limitations as to how much average activity can be generated at a node. For example, for some nodes 15% activity could be the best that can be achieved with any cycle. However, we show that it is intractable to compute optimal average switching activity at a node.

Stress-Test: Given a full scan circuit Q, a set of nodes $s \in Q$ and a *threshold* switching value t, compute stress tests such that maximum number of nodes have average switching activity greater than t.

4 Computing Dynamic Stress Tests

We show that for all the three types of nodes i.e. c-nodes, s-nodes and h-nodes, the problem of computing an optimal cyclic sequence is NP-hard [15]. Note that for monitored burn-in these problems were shown to be solvable in polynomial time [15]. The measure of stress used was average switching activity in all targeted nodes. Notwithstanding this difficulty in computing cyclic tests for dynamic burn-in it is interesting to investigate if fast heuristics that stress the circuit more than in monitored burn-in can be developed. Next we discuss the heuristics and experimental results for generating dynamic stress tests.

4.1 c-nodes

Heuristic-1 describes the algorithm used for generating stress tests for a given set of c-nodes. The heuristic consists of two phases. In phase 1 (steps (1), (2) and (3)), we compute a pair, from a given test set, for every targeted node such that the pair switches the node and at the same time switches maximum number of other nodes. This is done as follows.

Phase 1: In step (1), each test vector v is simulated and output of the simulation is stored in a BitVector[v] array. BitVector[v][n] ($n \in$ SetOfCNodes) stores the value at the output of node n when v is applied. Set of nodes switched by pair p is obtained by computing the hamming distance between BitVector[v_i] and BitVector[v_j]. Note that this preprocessing step helps speed up the switching activity computation since computing hamming distance is faster than performing logic simulation of a vector. Let $w(p)$ denote the number of node pairs p switches. Let pair[n] denote a pair which switches node n and for which w(pair[n]) is maximum for given n. Step (2) computes pair[n] for each node $n \in$ SetOfCNodes. In step (3) a subset of pairs from pair[i]'s is selected which switch all the nodes under consideration. Let $\mathcal{P}$ denote the selected set of pairs and let $\mathcal{V}$ be the set of vectors in $\mathcal{P}$.

Phase 2: Phase 2 constructs a cycle from the selected set of pairs. We start with an initial pair and add one vector at a time. As the cycle is updated average switching activity of each node is updated which is stored in the form of an array. Step (4) describes the *while* loop (steps 4.5-4.17) that constructs the cycle of vectors. During every iteration of the loop a vector is selected using the following set of rules. Let c be the last vector of the partial cycle constructed so far. For each vector $v \in \mathcal{V}$ compute the set of nodes which the vector pair (c, v) switches (step 4.8). These nodes are classified into 10 Bins (Bin[1]-[10]). Bin[1] stores the number of nodes having average switching activity between 0 and 0.1 that vector pair (c, v) switches. Bin[2] stores the number of nodes with activity between 0.1 and 0.2 that vector pair (c, v) switches. (step 4.8) Each Bin has a weight associated with it (BinWeight[i]). Weights to bins are assigned such that BinWeight[i] is less than BinWeight[i+1] ($1 \leq i \leq 9$). The idea is to select a vector which switches more nodes with poorer switching activity. For every vector, weight is computed as shown in step 4.9 and a vector with maximum weight is selected as the next vector of the cycle. The process is continued until either no improvement is found or the cycle is longer than a threshold length.

Heuristic-1 (SetOfCNodes)

(1) **for** (each vector $v \in$ Test set T)
  BitVector[v] $\leftarrow$ Simulate (v)
endfor
(2) **for** (all pairs $p = (v_i, v_j)$ such that $v_i, v_j \in T$)
  SetOfNodesSwitched $\leftarrow$ HammingDistance (BitVector, v_1,v_2)
  for (each node $n \in$ SetOfNodesSwitched)
    update pair [n]
  endfor
endfor
(3) $\mathcal{P} \leftarrow$ greedy selection of pairs that switch all the nodes
$\mathcal{V} \leftarrow \{v_i \in \mathcal{P}\}$
(4.1) $(v_i, v_j) \leftarrow$ pair with maximum weight in $\mathcal{P}$
(4.2) Cycle $\leftarrow \{v_i, v_j\}$
(4.3) CurrentVector $\leftarrow v_j$
(4.4) UpdateNodeTransitions (v_i, v_j)
(4.5) **while** (Cycle is not too long)
(4.6)   MaxWeight $\leftarrow 0$
(4.7)   **for** (each vector $v \in \mathcal{V}$)
(4.8)   Bins $\leftarrow$ Simulate (CurrentVector, v)
    /* Evaluate the vector v */
(4.9)   weight $\leftarrow \sum_{i=1}^{10}$ BinWeight$[i] *$ Bin$[i]$
(4.10)   **if** (weight $>$ MaxWeight)
(4.11)   MaxWeight $\leftarrow$ weight
(4.12)   BestVector $\leftarrow v$
(4.13)   **endif**
(4.14)   **endfor**
(4.15)   UpdateNodeTransitions (CurrentVector, BestVector)
(4.16)   CurrentVector $\leftarrow$ BestVector
(4.17) **endwhile**

4.2 s-nodes

Heuristic-2 describes the algorithm that generates stress tests for s-nodes. Note that in case of s-nodes primary inputs can be ignored. Therefore assignments to only the sequential inputs are discussed. In the initialization step (step (1)), scan chain is assigned the sequence 010101..etc. The idea is to have switching at every input per shift in the beginning. At every step, one shift bit is selected using a *k-lookahead* in which all possible combinations of k-shifts (a total of 2^k) are simulated and the first bit of the vector with maximum average activity is added to the cycle (steps (2.1)-2.7)). For example, in case of 2-lookahead, $\{00, 01, 10, 11\}$ are simulated. Weighted sum for each shiftvector is computed similar to the one is Heuristic-1 (step (2.4)). In the example, if 01 generates maximum weighted average activity then 0 will be shifted in (step (3)). While the partial cycle is being constructed, a few good folding points are maintained. For example, if 01010111001101 is the partial cycle constructed so far and if average activity spectrum was good at positions 1, 3 and 7 then $\{1, 3, 7\}$ is added to the FoldingPoint list. Once the cycle is sufficiently long, cycle is folded at all the folding points and the one with best average activity is chosen. For example, assume the sequence 010101 was in the scan chain in the beginning and the remaining part 11001101 was shifted in. Then corresponding to the three good folding points the three cycles 0101011, 010101110 and 0101011100110 will be simulated and best one will be chosen.

Heuristic-2 (SetOfSNodes, k) /* k is lookahead */
(1) Initialize scan chain to the sequence 010101...etc.
(2) /* k-lookahead */
(2.1) **while** (chain is not too long)
(2.2) **for** (v = 1 to 2^k)
(2.3) Bins $\leftarrow$ ShiftAndSimulate (v)
(2.4) weight $\leftarrow \sum_{i=1}^{10}$ BinWeight$[i] *$ Bin$[i]$
(2.5) **if** (weight $>$ MaxWeight)
(2.6) MaxWeight $\leftarrow$ weight
(2.7) BestShiftVector $\leftarrow v$
(2.8) **endif**
(2.9) **endfor**
(3) /* Let b be the first bit of BestShiftVector */
ShiftAndSimulate (b)
(4) UpdateGoodFoldingPoints ()
(5) **endwhile**
(5) **for** (each good folding point p)
(5.1) CompleteCycle (p)
(5.2) retain the best cycle
(5.3) **endfor**

4.3 h-nodes

As mentioned earlier h-nodes depend on both the primary inputs as well as the sequential inputs. In this heuristic we combine the approaches in Heuristic-1 and Heuristic-2 for the selection of primary input vectors and scan chain cycle respectively. In every phase of this heuristic (1) a shift bit and, (2) a primary input vector are selected. This is done by modifying Step (2.3) of Heuristic-2 as follows.

In step (2.3) of Heuristic-2 *ShiftAndSimulate* function simulates vector v by shifting in each bit of v and at the same time updating the average activity generated at the targeted nodes. In case of h-nodes, at every shift of vector v a primary input vector is to be selected. We select this vector by simulating all the test vectors and choosing the best vector.

5 Experimental Results

Heuristics described in the last section were implemented and evaluated on ISCAS89 benchmark circuits [16]. A machine having 4 Sparc-II processors was used as the platform. Code is implemented in C++. Table 1 shows c-node, s-node and h-node distribution of each circuit.

5.1 c-nodes

In general c-nodes constitute the smallest fraction among the three categories. But it can be seen from Table 1 that there could be circuits which have substantial fraction of c-nodes. For example, circuits like s510, s820, s1488 and s1494 have 82%, 63%, 44%, 44% c-nodes respectively. This justifies the separate treatment for c-nodes.

For each circuit (except s5378) one cyclic test was computed per group of 50 nodes. For every group nodes are classified into three categories. Nodes with average activity (1) less than 30% (column 30); (2) between 30% and 60% (column 30-60) and; (3) more than 60% (column 60). Average cycle length (averaged over all groups) is given in fifth column. CPU time is given in seconds.

Threshold for checking whether a cycle is too long was fixed to be 40. It can be seen that there were 8 circuits for which a pair of vectors was found that switches all the c-nodes. This is the best cycle that can be found in c-nodes. If minimum of 30% switching activity is considered in the acceptable range then almost always 90% of the c-nodes can be activated satisfactorily. Also note that the heuristic is in general fast.

5.2 s-nodes and h-nodes

Heuristic-2 was run with 2-lookahead and 4-lookahead. Results are are shown in Table 4 and Table 5. For a number of circuits like s382, s444, s641, s713, s1488, s1494 4-lookahead heuristic gives better cycles. In fact, in case of s1488 and s1494 4-lookahead heuristic does substantially better than that of 2-lookahead heuristic. However, in case of s298, s349, s526n, s526 2-lookahead results are better. Also note that the CPU time. for 4-lookahead heuristic is about 4 to 5 times more than that for 2-lookahead.

For s-nodes, cycles generated for monitored burn-in can also be used for dynamic burn-in. We compare the solutions computed by the optimal algorithm for generation cyclic stress tests for monitored burn-in described in [9] with k-lookahead heuristics. Results obtained by the Monitored burn-in heuristic are shown in Table 6. It can be seen that in general lookahead heuristics for dynamic burn-in computes much better cycles than that computed for monitored burn-in. However, note that for circuits like s641 cycle obtained for monitored burn-in is better than that obtained by lookahead heuristics. The running time for monitored burn-in heuristic is order of magnitude larger than that of lookahead heuristics. In fact for circuit s5378, monitored burn-in heuristic was able to compute cycles for only 3 out of 24 groups of s-nodes and that took 5640 minutes. On the other hand, comparable solution is obtained for all the sets of nodes using both 2-lookahead as well as 4-lookahead heuristic.

Any of the approaches can be easily combined by taking for each set of nodes the best cycle among the three heuristics. One such table obtained by combining monitored burn-in and 2-lookahead heuristics is shown in Table 6.

Results obtained for h-node heuristic are presented in Table 8. By observing the second column of Table 8 it can be noted that h-nodes are more difficult to activate than that of c-nodes and s-nodes. Moreover, simulating h-nodes takes more time resulting in longer run times.

6 Conclusions

In this paper, we studied the problem of computing cyclic sequences for dynamic burn-in of full-scan circuits. Nodes are classified into three types viz. c-nodes, s-nodes and h-nodes; and it was showed that the problem of computing a best stress test is intractable for each class of nodes. Furthermore fast heuristics, to compute good cyclic sequences, are presented. Heuristic-1 presented for c-nodes switches almost 90% of the nodes in every circuit. The k-lookahead heuristics compute good cyclic sequences for s-nodes. But in case of h-nodes, about 20-25% nodes in each circuit are not activated reasonably by the sequences computed by the heuristics. Future work should be directed towards computing better cycles for h-nodes. These heuristics are also faster than the optimal algorithm for monitored burn-in and compute better results.

References

[1] H. A. Chan, P. J. Englert, M. A. Oien, and S. Rajaram, "Environmental stress testing," *AT&T Tech-*

nical Journal, pp. 77–85, April 1994.

[2] M. H. Woods, “MOS VLSI reliability and yield trends,” *Proceedings of IEEE*, vol. 74, pp. 1715–1729, December 1986.

[3] E. Hnatek, “Thoughts on VLSI burn-in,” in *IEEE International Test Conference*, pp. 531–535, 1984.

[4] Y. Zorian, “Tutorial: MCM testing strategies,” in *International Test Conference*, 1995.

[5] C. Harry and C. Mathiowetz, “ASIC reliability and qualification: A user’s perspective,” *Proceedings of IEEE*, vol. 81, pp. 759–766, May 1993.

[6] T. Kwok, “Electromigration in VLSI metallization,” in *Metallization: Performance and Reliability Issues for VLSI and ULSI* (G. S. Gildenblat and G. P. Schwartz, eds.), vol. SPIE vol. 1596, pp. 60–71, don’t know yet, 1991.

[7] P. Yang and J.-H. Chern, “Design for reliability: The major challenge for VLSI,” *Proceedings of IEEE*, vol. 81, pp. 730–743, May 1993.

[8] W. Weber, “Dynamic stress experiments for understanding hot carrier degradation phenomena,” *IEEE Transactions on Electron Devices*, vol. 35, pp. 1476–1485, September 1988.

[9] V. Dabholkar, S. Chakravarty, F. Najm, and J. Patel, “Cyclic stress tests for full scan circuits,” *IEEE VLSI Test Syposium*, 1995.

[10] H. Kriplani, F. Najm, P. Yang, and I. Hajj, “Maximum current estimation in CMOS combinational circuits,” in *ACM/IEEE 29th Design Automation Conference*, pp. 2–7, 1992.

[11] A. Ghosh, S. Devadas, K. Keutzer, and J. White, “Estimation of average switching activity in combinational and sequential circuits,” in *ACM/IEEE 29th Design Automation Conference*, pp. 253–259, 1992.

[12] B. Kapoor, “Improving the accuracy of circuit activity measurement,” in *ACM/IEEE 31th Design Automation Conference*, pp. 734–739, 1994.

[13] F. Najm, “A survey of power estimation techniques in VLSI circuits,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 2, pp. 446–455, December 1994.

[14] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[15] V. Dabholkar and S. Chakravarty, "Stress tests for dynamic burn-in of full scan circuits," *Tech. Report No. 95-52, Dept. of Computer Science, SUNY at Buffalo*, 1995.

[16] F. Brglez, D. Byan, and K. Kozminsky, "Combinational profiles of sequential benchmark circuits," in *ACM/IEEE Int'l Symposium on Circuits and Systems*, pp. 1929–1934, 1989.

Appendix A: Average switching activity in c-nodes, s-nodes, and h-nodes

(1) Average switching activity in c-nodes: Let x be a c-node with output load capacitance c_x in scan circuit Q and assume primary inputs change from v_i to v_j. Let $P_x(v_i, v_j)$ denote switching activity at x

$$P_x(v_i, v_j) = \begin{cases} c_x & \text{if } v_i \rightarrow v_j \text{ causes output of gate } x \text{ to switch} \\ 0 & \text{otherwise} \end{cases}$$

Note that only to vectors are needed two switch one c-node. But if a set of c-nodes is targeted for stressing then, in general, a cyclic sequence may be needed. For example, consider a set of three c-nodes $\{n_1, n_2, n_3\}$. Assume a cyclic pair $p_1 = (v_1, v_2)$ which switches only n_1, another cyclic pair $p_2 = (v_2, v_3)$ which switches only n_2 and a third cyclic pair $p_3 = (v_3, v_1)$ which switches only n_3. Now each cyclic sequence p_1, p_2 or p_3 has 100% switching for 33% of the nodes and has 0% switching for the remaining 67% of the nodes. On the other hand, the cyclic sequence (v_1, v_2, v_3) has 33% switching for 100% of the nodes which is more desirable because each node is being switched at some level.

(2) Average switching activity at h-nodes: Consider a cycle v of length m. Let $v(k)$ denote the input to combinational part C of Q at k^{th} cycle during scan-in of v. Then, average activity during the shifting of complete cycle is:

$$P_x(v) = \sum_{k=1}^{m} P_x(v(k), v(k+1))$$

(3) Average switching activity at s-nodes: For s-nodes primary inputs can be ignored. As mentioned above, let $v(k)$ denote the input to C (without primary input part), then

$$P_x(v) = \sum_{k=1}^{m} P_x(v(k), v(k+1))$$

AND	s0	s1	r	f
s0	s0	s0	s0	s0
s1	s0	s1	r	f
r	s0	r	r	s0
f	s0	f	s0	f

Table 1: Zero Delay Model

circuit	c-nodes	s-nodes	h-nodes
s208.1	20	8	76
s298	7	78	34
s349	18	60	83
s382	15	94	49
s386	41	5	113
s420.1	46	14	158
s444	4	113	64
s510	173	1	37
s526	7	119	67
s526n	7	120	67
s641	34	47	298
s713	34	47	312
s820	183	4	102
s832	181	4	102
s838.1	98	26	322
s1423	23	170	464
s1488	291	18	344
s1494	288	14	345
s5378	79	2465	235
s9234	108	4805	684
s9234.1	414	2741	2442
s13207	215	7367	369
s15850	95	9395	282

Table 2: c-nodes, s-nodes, h-nodes in ISCAS89 circuits

circuit	30	30-60	60	avg cycle length	time sec
s208.1	0.00	0.00	100.00	2	0.1
s298	0.00	0.00	100.00	2	0.1
s349	0.00	11.11	88.89	40	0.4
s382	0.00	20.00	80.00	40	0.6
s386	0.00	68.29	31.71	40	2.5
s420.1	0.00	0.00	100.00	2	0.3
s444	0.00	0.00	100.00	2	0.1
s510	16.00	61.30	22.70	40	7.7
s526n	0.00	0.00	100.00	2	0.2
s526	0.00	0.00	100.00	2	0.2
s641	0.00	0.00	100.00	40	1.8
s713	0.00	0.00	100.00	40	1.6
s820	10.00	69.21	20.79	40	21.0
s832	10.50	60.90	28.60	40	22.5
s838.1	0.00	0.00	100.00	2	0.8
s1423	0.00	0.00	100.00	2	0.5
s1488	5.33	65.13	29.54	40	39.5
s1494	6.00	63.54	30.46	40	43.3
s5378	0.00	3.73	96.28	40	47.9
s9234	4.67	0.00	95.33	15	54.9
s9234.1	0.00	0.89	99.11	15	180.5
s13207	2.00	32.00	66.00	40	345.4
s15850	0.00	2.00	98.00	21	79.2

Table 3: c-nodes: Heuristic-1

circuit	30	30-60	60	avg cycle length	time sec
s208.1	0.00	0.00	100.00	12	0.6
s298	25.71	14.71	59.57	22	2.2
s349	29.00	11.00	60.00	24	3.0
s382	26.77	19.18	54.05	34	4.5
s386	0.00	0.00	100.00	10	1.0
s420.1	0.00	0.00	100.00	20	2.2
s444	27.59	26.72	45.69	52	10.2
s510	0.00	0.00	100.00	7	1.0
s526n	27.33	16.33	56.33	29	7.2
s526	28.18	16.77	55.05	29	7.2
s641	17.02	10.64	72.34	20	3.7
s713	17.02	10.64	72.34	20	5.5
s820	0.00	0.00	100.00	9	1.4
s832	0.00	0.00	100.00	9	1.1
s838.1	0.00	0.00	100.00	36	8.1
s1423	20.25	5.50	74.25	80	77.6
s1488	33.33	16.67	50.00	10	2.8
s1494	14.29	14.29	71.43	20	2.8
s5378	39.86	0.24	59.90	188	3886.8
s9234	37.92	7.71	54.37	253	17875.0
s13207	32.87	5.01	62.12	892	26770.4
s15850	38.43	14.88	46.69	796	36555.9

Table 4: s-nodes: Heuristic-2 with 2-lookahead

circuit	30	30-60	60	avg cycle length	time sec
s208.1	0.00	0.00	100.00	12	2.7
s298	31.07	14.71	54.22	28	11.3
s349	29.00	16.00	55.00	26	17.2
s382	12.96	39.41	47.63	89	26.6
s386	0.00	0.00	100.00	10	3.9
s420.1	0.00	0.00	100.00	20	12.2
s444	25.69	28.72	45.59	33	45.4
s510	0.00	0.00	100.00	7	3.9
s526	33.02	12.77	54.21	31	43.4
s526n	32.67	11.67	55.67	31	42.9
s641	14.89	4.26	80.85	56	20.0
s713	14.89	4.26	80.85	56	22.0
s820	0.00	0.00	100.00	9	6.7
s832	0.00	0.00	100.00	9	5.0
s838.1	0.00	0.00	100.00	36	45.8
s1423	20.25	5.50	74.25	80	407.6
s1488	0.00	50.00	50.00	20	15.0
s1494	0.00	50.00	50.00	20	14.4
s5378	39.86	0.24	59.90	188	22567.7

Table 5: s-nodes: Heuristic-2 with 4-lookahead

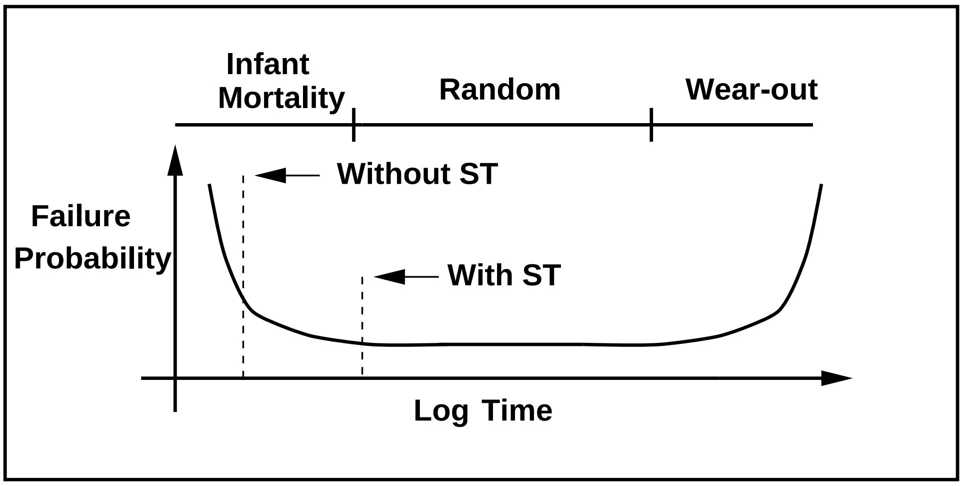

Figure 1: Bath tub curve

circuit	30	30-60	60	avg cycle length	time sec
s27	0.00	100.00	0.00	6	0.1
s208.1	0.00	100.00	0.00	34	4.7
s298	32.64	56.22	11.14	5	44.0
s344	27.11	33.11	39.78	7	25.7
s349	48.00	52.00	0.00	13	41.1
s382	26.05	71.68	2.27	18	103.9
s386	40.00	60.00	0.00	80	40.0
s420.1	0.00	100.00	0.00	76	90.6
s444	28.15	54.05	17.79	11	169.4
s510	0.00	100.00	0.00	65	66.2
s526	30.77	68.56	0.67	7	673.2
s526n	29.67	67.67	2.67	5	612.0
s641	6.38	82.98	10.64	40	84.2
s713	17.02	82.98	0.00	54	60.6
s820	0.00	100.00	0.00	14	254.3
s832	0.00	25.00	75.00	24	297.2
s838.1	0.00	100.00	0.00	152	1321.4
s1423	19.50	78.50	2.00	19	5170.9
s1488	22.22	66.67	11.11	130	644.0
s1494	0.00	92.86	7.14	142	738.7
s5378*	30.00	68.75	1.25	205	338798.5

*partial results, average of 4 sets of nodes

Table 6: s-nodes: Monitored burn-in results

circuit	30	30-60	60	avg cycle length	time sec
s208.1	0.00	0.00	100.00	12	5.3
s298	23.71	43.72	32.57	20	46.2
s349	29.00	11.00	60.00	24	44.2
s382	23.77	59.18	17.05	24	108.4
s386	0.00	0.00	100.00	10	41.0
s420.1	0.00	0.00	100.00	20	92.8
s444	22.92	48.05	29.03	55	179.6
s510	0.00	0.00	100.00	7	67.3
s526n	24.00	43.67	32.33	40	619.1
s526	25.51	30.11	44.39	45	680.4
s641	6.38	82.98	10.64	40	87.8
s713	17.02	10.64	72.34	20	66.0
s820	0.00	0.00	100.00	9	255.7
s832	0.00	0.00	100.00	9	298.2
s838.1	0.00	0.00	100.00	36	1329.5
s1423	15.75	66.00	18.25	49	5248.5
s1488	22.22	66.67	11.11	130	646.8
s1494	0.00	92.86	7.14	142	741.4
s5378	10.33	58.67	31.00	162	4625.5

Table 7: s-nodes: Combining monitored burn-in and Heuristic-2

circuit	30	30-60	60	avg cycle length	time sec
s208.1	26.62	38.54	34.84	15	37.3
s298	20.59	29.41	50.00	21	41.1
s349	12.00	8.03	79.97	26	74.1
s382	34.69	24.49	40.82	27	47.6
s386	18.67	17.33	64.00	9	154.9
s420.1	32.00	0.50	67.50	20	596.4
s444	29.14	28.57	42.28	31	109.5
s510	21.62	27.03	51.35	11	50.4
s526n	24.82	37.47	37.70	28	349.5
s526	20.82	40.53	38.64	54	350.3
s641	14.48	18.06	79.96	42	871.7
s713	12.33	13.14	74.52	25	1021.2
s820	16.67	18.67	64.67	12	390.1
s832	16.67	14.00	69.33	14	372.6
s838.1	37.87	1.71	60.42	38	6205.2
s1423	26.17	25.29	48.54	107	16377.7
s1488	25.34	20.96	53.70	15	1888.6
s1494	24.63	22.92	52.44	13	2070.5
s5378	24.11	10.00	65.89	269	285123.6

Table 8: h-nodes: 2-lookahead heuristic

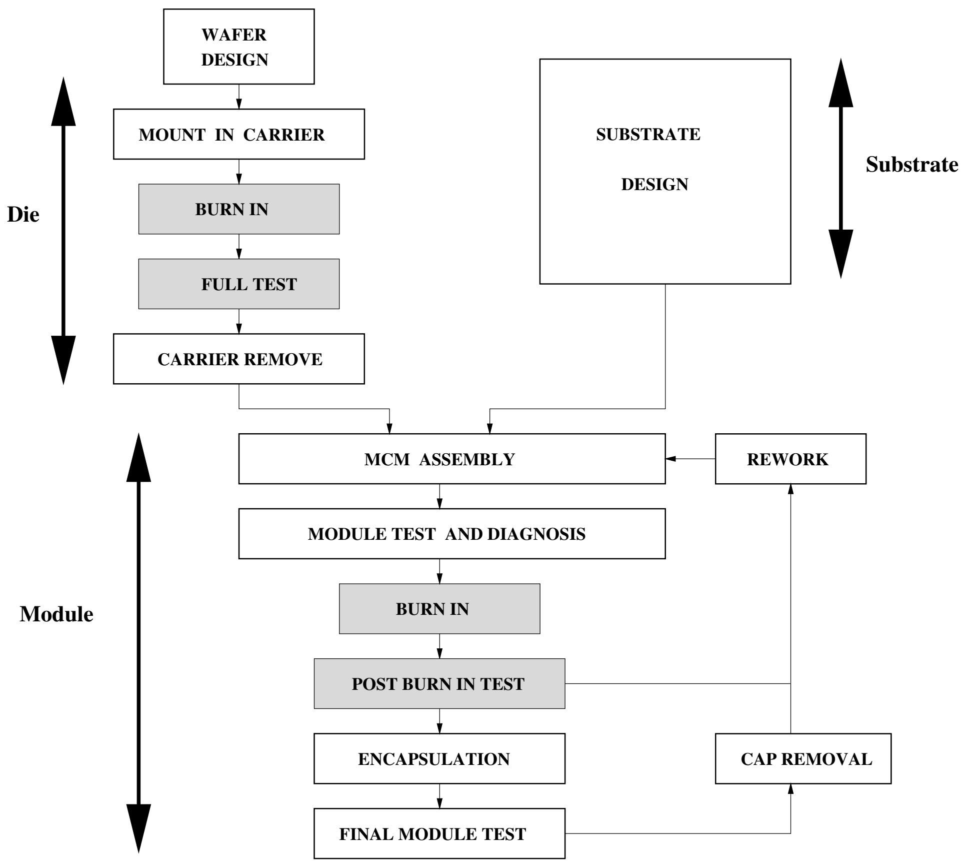

Figure 2: MCM Production Flow

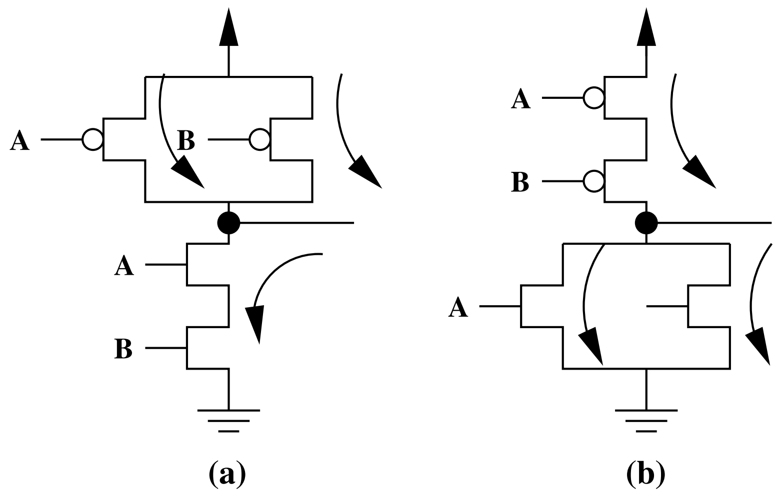

Figure 3: Switching in NAND and NOR gates

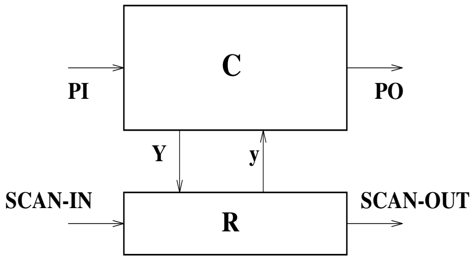

Figure 4: Scan Structures

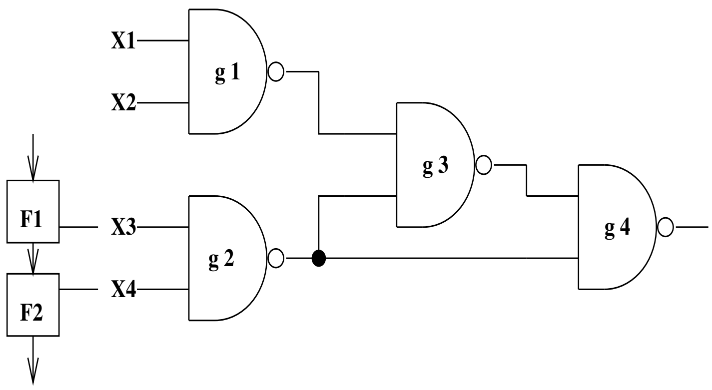

Figure 5: Scan Example