Cyclic Stress Tests for Full Scan Circuits

Vinay P. Dabholkar Sreejit Chakravarty
Department of Computer Science
State University of New York
Buffalo, NY 14260
Farid Najm
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
Janak Patel
Center for Reliability and High-Performance Computing
University of Illiois at Urbana-Champaign
Urbana, IL 61801

Abstract

To ensure the production of reliable circuits and fully testable unpackaged dies for MCMs burn-in, both dynamic and monitored, remains a feasible option. During this burn-in process the circuit needs to be stressed for an extended period of time. This requires computation of cyclic input sequences to stress the circuit. A taxanomy of stress related problems for full scan circuits is presented. It is shown that there are efficient ways to compute the sequences for most variations of monitored burn-in problems. The difficulty of computing stress tests for dynamic burn-in problems is discussed. Preliminary experimental results on ISCAS89 benchmark circuits are presented.

1 Introduction

Growing size of VLSI circuits, high transistor density and advent of MCM technology is making production of reliable chips a challenging task. Functional testing and various kind of burn-ins have been used to guarantee greater reliability. It has been observed that potential advantages in circuit density and performance of MCM technology cannot be realized without access to fully tested, unpackaged integrated circuits. Until die integrity is guaranteed by qualification of the process that fabricates the die, functional testing and burn-in procedures remain the best viable alternatives [1]. Hence, *monitored burn-in*, a kind of burn-in in which functional testing and burn-in are done at the same time, are becoming popular.

There are three different types of Burn-Ins : Static, Dynamic and Monitored. Selection of a particular technique depends upon the types of defects targeted and the extent to which these defects are activated by various techniques [2].

- **Static Burn-In:** applies a DC bias to the device at an elevated temperature (normally 125^{o}C) in a manner predetermined to either forward bias or reverse bias as many junctions as possible within the device. Static burn-in is most effective in weeding out devices with thermally activated surface related defects showing up as excessive leakage current, speed degradation or threshold voltage shifts after electrical tests.

- **Dynamic Burn-In:** where all clocks and address lines are continually sequenced, results in high power dissipation, current densities and chip temperature than static burn-in. Increased current densities stress defects such as epitaxial and crystal imperfections, metallization, oxide and junction anomalies, that include pipes and emitter shorts.

- **Monitored Burn-In:** Because of long electrical test times associated with large circuits, testing during dynamic burn-in is becoming widespread. This is called monitored burn-in. There are couple of advantages in using monitored burn-in against dynamic burn-in [3]. First, it utilizes the "dead time" during burn-in in testing. Second, carefully ordered test vectors can stress circuit nodes to their maximum. On the other hand, during dynamic burn-in, if the input vectors are not carefully chosen, switching activity in some parts may not be sustained at a higher level. Third, if a junction breaks down during burn-in, monitored burn-in will make it easier to detect the location of the fault.

Since power dissipation due to switching transient current and charging and discharging of output load capacitance is generally more than that due to leakage current [4], dynamic burn-in and monitored burn-in are more widely used [3]. In this paper, we consider dynamic and monitored burn-in. During burn-in, cyclic sequences need to be applied over an extended period of time such that the average switching activity for a targeted set of nodes is maximized. To our knowledge no systematic study of these problems exists. The main purpose of this paper is to study solutions to these problems in the context of Full Integrated Scan circuits [5]. It is shown that computation of cyclic stress tests can be done efficiently for most problems pertinent to monitored burn-in. For these problems we present optimal, polynomial time algorithms. In addition, faster heuristics for large circuits that compute near optimal solutions is presented.

The paper is organized as follows. Power dissipation model is briefly discussed in Section 2. In section 3, a taxonomy of cyclic stress test computation problems is presented. Next, in section 4, we discuss how to account for power dissipation when a cycle of input vectors are applied to a full scan circuit. Polynomially solvable problems, solutions and experimental results are discussed in section 5. Intractability issues of dynamic burn-in problems are discussed in section 6.

2 Power Dissipation Model

The two components of power dissipated in a CMOS circuit [4] are: Static dissipation due to leakage current or other current drawn continuously from the power supply (P_{st}) and dynamic dissipation due to (i) switching transient current (P_{sc}) and (ii) charging and discharging of load capacitances (P_d). The total power dissipation P_{total} is given by :- P_{total} = P_{st} + P_d + P_{sc}. As in [6] P_{st} and P_{sc} are neglected.

P_d is power required to charge and discharge the output capacitive load of every gate. P_d is approximated as follows:

$$P_d = 1/2 \times C \times V_{DD}^2 \times N_G \times f \tag{1}$$

where C = output capacitance; N_G = total number of gate output transitions ($1 \rightarrow 0$ or $0 \rightarrow 1$); and f = repetition frequency.

Equation (1) implies that power is dissipated at a node when the input vector is changed from T_i to T_{i+1}. Let $P_C(T_i, T_{(i+1)})$ be the total power dissipated in C when inputs change from T_i to T_{i+1}. Then,

$$P_C(T_i, T_{(i+1)}) = \sum_{j \in SetofNodes} 1/2 \times C_j \times V_{DD}^2 \times N_{G_j} \times f \tag{2}$$

Thus power dissipated at a node is proportional to the number of transitions (N_{G_j}) at that node. This depends on the gate delays and sequence of input vectors applied. Next we discuss how to compute the the transitions at the other nodes.

Under the zero-delay model, all gates are assumed to have zero delay. The four possible transitions 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. Drawback of this model is that it neglects glitches at the output of gates and consequently the power dissipation is underestimated.

3 A Taxonomy of Stress Tests Related Problems

Typically burn in is performed for an extended period, usually several hours. So, while selecting vectors for Burn-in, we will be more concerned with the ability of the resulting sequence to dissipate power rather than its length.

What is a good measure of stress during burn in ? In other words, given a sequence of vectors we would like to know its effectiveness in stressing the circuit. One possible measure is the average switching where the transitions at all nodes in the circuit are considered. The other extreme of this is to compute individual sequences for each node. The input sequence for a node would maximize the average switching activity at that node, thus stressing the node to the maximum. Following example shows that the former approach may not be the best approach.

Consider the scan circuit C shown in Figure 1. Inputs x_3 and x_4 come from the scan latches F_1 and F_2. Consider the test set $T = \{t^1 = 0001, t^2 = 0000, t^3 = 1011, t^4 = 1111\}$. Inputs assignments given in the order $\langle x_1, x_2, x_3, x_4 \rangle$. Table 2 shows the number of transitions at the output of gates g_1, g_2, g_3 and g_4 for all combinations of transitions of vectors from test set T under the zero delay model. Scan-in consists of two clock cycles. Assume that primary inputs x_1, x_2 change during the second clock cycle.

Graph G in Figure 1 represents the total number of transitions in circuit C during scan-in of each pair of test vectors. Each node in G corresponds to a test vector and each edge weight $w(i, j)$ corresponds to the number of transitions during an application of vector t^i followed by t^j under the zero delay model. It can be observed that $\langle t^2, t^4, t^2 \rangle$ constitutes a cycle with the maximum average weight among all possible cycles in G.

Now consider the number of transitions at the output of gate g_4. Only one vector pair causes output of gate g_4 to switch viz. ($t^4 \to t^2$). If vectors are cycled such that this order is not part of the cycle then output of g_4 will never be stressed. We consider this an unacceptable stress test. Thus we would like to choose a cycle of vectors such that cycling through it would maximize the average power dissipation in all parts of the circuit.

Typically during scan in, the values of the primary inputs are kept constant until the scan part of the vector is completely scanned in [5]. If this constraint is dropped then primary input vector can be switched during any clock cycle during the scan in process. Let *switching time* denote the clock cycle at which the primary inputs are switched.

Consider the transition 0000 $\to$ 1111 in the circuit in Figure 1(a). Here primary inputs switch from 00 to 11. Scan in consists of two cycles in which 00 is scanned out and 11 is scanned in. Table 3 compares the number of transitions at the output of gate g_4 when (a) primary input switches in the first clock cycle and; (b) primary input switches in the second clock cycle. First two rows indicate that there are 2 transitions in case (a) while the last two rows indicate that there is none in case (b).

Now consider transitions 0001 $\to$ 0000. Table 4 (case (a)) shows that there is no transition at the output of any gate during scan in. However, if primary input vector 11 is applied during the first cycle, 2 transitions take place at the output of each of the gate g_1, g_3 and g_4 (case (b)). Note that primary input switches multiple times (00 $\to$ 11, 11 $\to$ 00) during scan in.

Tables 3 and 4 indicate that the time at which primary input vector switches and the number of times it switches can affect the transitions at the output of gates.

So far the examples considered used vectors from a given test set only. This restriction is important in the case of monitored bun-in. But it is not required in the case of dynamic burn-in. Additional test vectors, not belonging to test set, can be used which will maximize the transitions at the gate outputs. Consequently, we can define a number of problems depending on whether the primary inputs switch once or multiple times and whether the input vectors are selected from the test set. All these variations are summarized in Figure 2.

Figure 2 shows the taxonomy of burn-in problems described above. Note that classification based on switching of primary inputs and that due to selection of vectors are orthogonal to each other. For example, a problem can be considered where primary input is switched only once and primary input vectors as well as scan vectors may or may not be selected from the test set. Problem with single switching (SS) and selection of vectors from Test Vectors Only (TVO) is denoted by SS-TVO. Problems SS-TVP, SS-AV, MS-TVO, MS-TVP and MS-AV are similarly denoted in Figure 1. Problems of type SS-TVO, SS-TVP, MS-TVO and MS-TVP where the applied vectors involve all the test vectors, may or may not be with additional vectors, are useful in monitored burn in. On the other hand, problems SS-AV and MS-AV are useful during dynamic burn in.

4 Accounting for Power Dissipation in Full Scan Circuits

Block diagram of Full Integrated Scan is shown in Figure 3 [5]. C is the combinational part of the circuit while R is the test register where the tests are scanned serially. Note that as a new vector is scanned in, the input to the combinational circuit C changes.

Next we divide the circuit nodes into different categories whose characteristics can be exploited while constructing vector sequences that maximize switching activity at their output. Consider a node that has only primary inputs in its fanin. e.g. node g_1 in Figure 1(a). Such a node does not depend upon state variables. Vectors (v_1, v_2) should be found such that set one of the vectors sets the node to 1 and the other sets it to 0. Cycling $\langle v_1, v_2 \rangle$ continuously would stress the node to its maximum. We call such a node, a **c-node**.

Consider a node that has only state input in its fanin. Here the problem reduces to finding a cycle $\langle v_1, ..., v_k \rangle$ such that average switching activity due to application of v_1 followed by v_2 and so on followed by v_k is maximum. We call such a node, an **s-node**. we call a node that has primary input as well as state input variables in its fanin, an **h-node**. The following notations are used in the rest of the paper.

Q	=	a sequential circuit
n	=	number of primary inputs

m	=	number of primary outputs
PI	=	$\{x_1, \ldots, x_n\}$ primary inputs of Q
PS	=	$\{y_1, \ldots, y_m\}$ present state inputs of Q
NS	=	$\{Y_1, \ldots, Y_m\}$ next state outputs of Q
$S(t^i, t^j)$	=	$\langle s_1, \ldots, s_{2m} \rangle \quad = \quad \langle Y_1^i, \ldots, Y_m^i, y_1^j, \ldots, y_m^j \rangle$
$S(k : l)$	=	$\langle s_k, \ldots, s_l \rangle$
t^i	=	$\langle PI^i @ PS^i \rangle$,(@ denotes concatenation)
	=	$\langle x_1^i, \ldots, x_n^i @ y_1^i, \ldots, y_m^i \rangle$ be a test vector, where PI^i are primary inputs and PS^i are present state variables of t^i

Let $I(t^i, t^j)$ denote the $(m+1)$ clock cycle interval while t^j is scanned in and its response is latched into the scan chain. During the first m clock cycles PS^j is scanned in and NS^i is scanned out in test register R. In $(m+1)^{st}$ clock cycle output of C, i.e. NS^j is latched to R. Next, we describe power dissipated in a set of c-nodes, s-nodes and h-nodes separately.

(1) Power dissipated in c-nodes: Let x be a c-node in scan circuit Q and assume that the primary inputs change from v_i to v_j. Since x is a c-node, power dissipated in x does not depend upon sequential input, hence only primary input is considered. Let $P_x(v_i, v_j)$ denote power dissipated in x when primary inputs change from v_i to v_j.

$$P_x(v_i, v_j) = \begin{cases} 1 & \text{if } v_i \rightarrow v_j \text{ causes output of gate } x \text{ to switch} \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

Let $X = \{x_1, ..., x_l\}$ denote a collection of c-nodes. Power dissipated in X when primary input changes from v_i to v_j is denoted by:

$$P_X(v_i, v_j) = \sum_{k=1}^{l} P_{x_k}(v_i, v_j) \qquad (4)$$

(2) Power dissipated in s-nodes: Let x be an s-node of Q. Consider the interval $I(t^i, t^j)$. Since x does not depend upon primary input, it can be ignored. In the first cycle, input to C is $S(1 : m)$. Similarly, in the k^{th} cycle the input to C is $v(k) = S(k+1 : k+m-1)$. Then

$$P_x(t^i, t^j) = \sum_{k=1}^{l} P_x(v(k), v(k+1)) \qquad (5)$$

If $X = \{x_1, ..., x_l\}$ denote a collection of s-nodes, power dissipated during $I(t^i, t^j)$ is given by:

$$P_X(t^i, t^j) = \sum_{t=1}^{l} P_{x_k}(t^i, t^j) \qquad (6)$$

(3) Power dissipated in h-nodes: Consider interval $I(t^i, t^j)$. Let $\langle v_1 = PI^i, v_2, ..., v_{m-1}, v_m = PI^j \rangle$ be a set of primary input applied to C during this interval in the given order. Input to C during k^{th} cycle is $v(k) = v_k @ S(k+1 : k+m-1)$. Power dissipated during $I(t^i, t^j)$ is given by:

$$P_x(t^i, t^j) = \sum_{k=1}^{l} P_x(v(k), v(k+1)) \tag{7}$$

If $X = \{x_1, ..., x_l\}$ denote a collection of h-nodes, power dissipated during $I(t^i, t^j)$ is given by:

$$P_X(t^i, t^j) = \sum_{r=1}^{l} P_{x_r}(t^i, t^j) \tag{8}$$

Given a node x in circuit Q, we would like to choose a sequence of vectors $\langle s^1, ..., s^k \rangle$ such that average power dissipated in node x given by the expression $\frac{1}{k}((\sum_{i=1}^{k-1} P_x(s^i, s^{i+1})) + P_x(s^k, s^1))$ is maximized. Note that $k = 2$ in case x is a c-node. Here $P_x(s^i, s^{i+1})$ corresponds to equations (3), (5) or (7) according to whether x is a c-node, s-node or h-node respectively. In case of a collection X of nodes of a particular type, $P_x(s^i, s^{i+1})$ is replaced by $P_X(s^i, s^{i+1})$ as defined by equations (4), (6) and (8).

5 Polynomially Solvable Monitored Burn-in Problems

In this section, we show that optimal solutions to most problems of interest in monitored burn-in can be obtained in polynomial time. These problems fall under the category in which (1) Vectors are chosen from a given set of test vectors only (TVO) and ; (2) Primary input is assumed to remain constant during scan-in of a new vector (single switching or SS). Next, we describe problems and algorithms for c-nodes, s-nodes and h-nodes.

5.1 c-nodes

Problem of computing a pair (v_1, v_2) that causes gate output of a c-node to switch is simple if a stuck at test set for the circuit is available. A solution can be obtained by simulating the set of test vectors and identifying a vector that sets the node to 0 and another that sets the node to 1. Since a c-node is independent of sequential input, multiple switching is not relevant in the context of c-nodes. In general, addition of new vectors (not in test set) is not going to improve the quality of the solution as there is just one transition possible under the zero delay model. Unless the fault at the gate output is untestable the vectors that cause the transition can be obtained from the test set. Thus variants of the problems for c-nodes, where selection of vectors need not be from test set e.g. TSP and VA, are not relevant in the context of a single c-node.

The simulation based approach mentioned above can be extended to multiple c-nodes case. Given a set of c-nodes, simulation of all pairs of test vectors would give the number of gate outputs that switch due to the pair of input vectors. If there are l test vectors, choosing the best pair would take $O(l^2)$ time.

5.2 s-nodes

Consider an s-node x and a test set $T = \{t^1, ..., t^l\}$. A complete graph $G = (V, E)$ is constructed such that v_i corresponds to vector t^i and weight function on the edges is defined as $w : E \rightarrow R$ where $w(v_i, v_j) = P_x(t_i, t_j)/F$. $P_x(t_i, t_j)$ is given by equation (5). F represents the number of flip flops. Weight function w represents the average number of transitions during scan-in of t^i and scan-out of t^j per clock cycle. We would like to compute a cycle $\langle v_{i_1}, ..., v_{i_k}\rangle$ such that $\frac{1}{k}((\sum_{r=1}^{k-1} w(v_{i_r}, v_{i_{r+1}})) + w(v_{i_k}, v_{i_1}))$ is maximized.

For example, consider the s-node g_2 in Figure 1 and the test set shown in Table 2. The complete graph in Figure 5 represents the transitions at the output of g_2 for all possible combinations of test vector transitions. Maximum average weight of a cycle in this graph is 2. In fact, there are a number of cycles with average weight 2. e.g. $\langle t^2, t^4\rangle$, $\langle t^2, t^3\rangle$, $\langle t^1, t^3\rangle$, $\langle t^1, t^4\rangle$ etc. Note that if all the edge weights are negated in the original graph then a cycle with maximum average weight in the original graph corresponds to a minimum average weight cycle in the new graph. For example, if all the edge weights of graph in Figure 5 are negated then minimum average weight of any cycle is -2. Thus computation of maximum average weight cycle and minimum average weight cycle in a directed weighted graph are equivalent problems.

Karp studied the problem of computing minimum edge weight cycle in a directed weighted graph and proposed an algorithm which yields optimal solution in time $O(|V||E|)$ and $O(|V|^2)$ space [7]. The algorithm is a modification of the Bellman-Ford single-source shortest path algorithm and it is based on the following important observation

Observation 5.1 *[7] It can be assumed that every vertex is reachable from every other vertex in the graph. If this were not the case, graph can be partitioned into strongly connected components each having this property. let s be an arbitrary (source) vertex. Let $\delta_k(s, v)$ denote total weight of the shortest path from s to v such that the path has exactly k edges. If μ denotes the weight of a cycle with minimum average weight then*

$$\mu = \min_{v \in V} \max_{0 \leq k \leq n-1} \frac{\delta_n(s, v) - \delta_k(s, v)}{n - k}$$

Based on the above observation, Bellman-Ford algorithm for single-source shortest path can be modified to get the required algorithm. For sake of completeness, we give the algorithm below.

Algorithm Optimal (G, w, s)

(1) Initialize-Single-Source (G, s);

(2) **for** $(i \leftarrow 1$ to $n)$

(3) **for** (each edge $(u, v) \in E$)

(4) **if** $(\delta_k(s, v) > \delta_{k-1}(s, u) + w)$

(5) $\delta_k(s, v) = \delta_{k-1}(s, u) + w$

(6) **end for**

(7) **end for**

(8) **for** $(v \in V)$

(9) **for** $(k \leftarrow 0$ to $n - 1)$

(10) $t = \frac{\delta_n(s,v) - \delta_k(s,v)}{n-k}$

(11) **if** (t > max)

(12) max = t

(13) **end for**

(14) **if** (max > min)

(15) min = max

(16) **end for**

(17) output min

Since the graph under consideration is always complete, Karp's algorithm runs in $O(n^3)$ time where n is the number of test vectors for the circuit. Next, we illustrate the algorithm with a hypothetical graph. Consider the graph in Figure 6. The graph has 3 nodes viz. $\{s, v_1, v_2\}$. Let s be the source vertex. At the end of one iteration of the for loop in steps 2-7, we have $\delta_1(s, s) = \infty$, $\delta_1(s, v_1) = 4$, $\delta_1(s, v_2) = 5$. At the end of second iteration we have, $\delta_2(s, s) = 7$, $\delta_2(s, v_1) = 6$, $\delta_2(s, v_2) = 10$ and after third iteration, $\delta_3(s, s) = 9$, $\delta_3(s, v_1) = 11$, $\delta_3(s, v_2) = 12$. If $\mu(v)$ denotes the term $\max\limits_{0 \le k \le n-1} \dfrac{\delta_n(s, v) - \delta_k(s, v)}{n - k}$, then $\mu(s) = 3$ for k = 0, $\mu(v_1) = 4.5$ for k = 1 and $\mu(v_2) = 4.5$ for k = 1. Thus it can be observed that weight of the minimum average weighted cycle is 3 viz. $\langle s, v_2, v_1 \rangle$.

Experiments were performed on ISCAS89 benchmark circuits [8] on SUN sparcII machine. Table 5 gives the number of c-nodes, s-nodes and h-nodes in the ISCAS89 circuits. Experimental results for the optimal algorithm are given in Table 6. For each circuit, sets of 50 s-nodes were grouped together and maximum weight cycle was computed using the optimal algorithm. part1, part2 etc. denote the partitions of s-nodes into 50 nodes each. For each partition, length and weight denote the length and the weight respectively of the optimal cycle. Following reasons explain why runs for larger circuits (s5378 onwards) did not complete.

There are two main drawbacks of the optimal algorithm. (1) It requires construction of a complete graph which involves n^2 edge weight computations. Each edge weight computation consists of F logic simulations where F is the number of flip flops in the scan chain. For large n and F the computation time is very large. (2) Running time $O(n^3)$ will be too large for large n. i.e. large circuits. This motivates the study of heuristics that do not require construction of a complete graph and run faster. Next, we present a randomized greedy heuristic for the same problem.

Algorithm GreedyRandomized (k, factor)

(1) source $\leftarrow$ pick a random vertex
(2) current-node $\leftarrow$ source
(3) path $\leftarrow$ current-node
(4) **while** (length(path) < $n-1$){
(5) **for** ($i \leftarrow 1$ to k)
(6) $v \leftarrow$ pick i^{th} distinct random vertex
(7) **if** ($v \in$ path) { /* a cycle is formed */
(8) cycle-cost $\leftarrow$ cost of the cycle due to
the back edge (current-node,v)
(9) **if** (cycle-cost < best-cycle-cost)
(10) best-cycle-cost $\leftarrow$ cycle-cost
(11) }
(12) **else** { /* update path */
(13) path-cost $\leftarrow$ cost of the path if v is added to it
(14) **if** (path-cost < best-path-cost)
(15) best-path-cost $\leftarrow$ path-cost
(16) }
(17) **end for**
(18) **if** (best-cycle-cost < factor * best-path-cost)
(19) terminate
(20) **else**{
(21) store best-cycle-cost
(22) path $\leftarrow$ path $\cup$ $\{v\}$
(23) }

(24)}

The procedure takes two parameters viz. k and factor. k denotes the number of edge weights computed for each node before making the greedy choice of next vertex in the path. The heuristic works as follows: It starts with a random source vertex as a path and then at each step of iteration (steps 4-24) picks k vertices at random with replacement. Edge weights for these vertices are computed. Average cycle cost is computed if a vertex belongs to the path and average path cost is computed otherwise. best cycle is accepted only if it is much better than the best path i.e. if average path < factor $\times$ average cycle. k was chosen to be 25 while factor was set to 0.6. Time complexity of this algorithm is $O(n)$.

Experimental results of GreedyRandomized algorithm are given in Table 7. For each ISCAS89 benchmark circuit, 10 iterations of this randomized heuristic were performed with random starting vertex. For each partition, denoted by part1, part2 etc., best weight denotes the best cycle weight obtained among the 10 iterations and worst weight denotes the worst weight among the 10 iterations. worst weight represents how bad a solution can get if the heuristic is run only once. length represents the cycle length for the best cycle. For example, in s298 for first set of 50 s-nodes, weight of the optimal cycle is 40.11 and its length is 10 (see Table 6). By running greedy heuristic 10 times the best solution computed was 35.68 with length 2 and 33.18 was the worst weight obtained in any individual run.

Circuit s5378 has 2465 s-nodes. We partitioned them into sets of 100 nodes each. Results of the first 20 partitioned are tabulated in Table 7. These runs took 4370 minutes; using one run of the greedy heuristic.

Table 8 compares the weights of the best and worst cycles obtained during greedy heuristic with the optimal weight. greedy best % denotes the average of all percentages (from all partitions) of best weight cycles with respect to the optimal weight. For example, in circuit s298 first partition has best percentage $(35.68/40.11) \times 100 = 88.95$ and second partition has best percentage $(16.96/16.96) \times 100 = 100$. 92.24 is the average of these two percentages which is listed in Table 8. optimal time denotes the user time in seconds for optimal algorithms. greedy avg time denotes the average time per iteration of the greedy heuristic in seconds. While comparing performance of a deterministic algorithm with a randomized algorithm following norms should be noted. If the best solution obtained during all the iterations is to be compared with the solution obtained by deterministic algorithm then the total time for all iterations should be considered. However, if worst solution obtained during all the iteration is compared with the solution obtained by deterministic algorithm then average time of an iteration of the randomized algorithm can be compared with that of deterministic algorithm.

5.3 h-nodes

If vectors chosen are restricted to test set only and if primary inputs switch at the last clock cycle during scan-in then computation of the cycle with maximum activity is similar to the s-node case. In case of s-nodes primary input change does not matter however, in case of h-nodes, during the last clock cycle change in primary input should be considered. Same algorithms were run on ISCAS89 circuits and results are tabulated in Tables 9 and 10. The performance and timings are compared in Tables 11.

6 Comments On the Intractability of Dynamic Burn-in Problems

Consider a c-node in a full-scan circuit. As stated in section 5, two stuck-at test vectors, one of which sets the node to 0 and the other that sets the node to 1 would suffice to generate maximum activity at the node. However, a c-node may not have a stuck-at test vector for two reasons. (1) The node may not be sensitizable [5] or; (2) the fault may not be propagated to a primary output. In case (1) no activity can be generated at the node. However, in case (2) a pair of vectors may exist that causes the output of the gate to switch but a stuck-at test vector may not exist. This motivates the need for a method of generating activity at the output of such c-nodes. A direct reduction from satisfiability [9] would show that this problem is intractable.

Note that in case of multiple c-node case the problem is at least as hard as the single c-node case. In fact, multiple c-node case is equivalent to computation of worst case power dissipation in a combinational circuit when the activity in all the c-nodes is considered together. We feel that the techniques of Davadas et. al. [10] and Kriplani et. al. [11] can be used in this context.

In case of s-nodes an optimal algorithm was discussed which computes a cyclic sequence with maximum average activity. It is an interesting problem to see if addition of new vectors (not be from the test set) would increase the average power dissipation significantly. However, we conjecture that this problem is intractable. It can be shown for h-node the problem of computing stress cycles, when inputs are not restricted to a test set, is at least as hard as that for c-nodes even when the number of scan latches is just one.

Notwithstanding these difficulties in computing cyclic tests for dynamic burn-in it is interesting to determine if fast heuristics that compute suboptimal solutions that stress the circuit more than in monitored burn-in can be developed. Heuristics for these problems are beyond the scope of this paper.

7 Conclusions

In this paper we presented a systematic approach towards generating cyclic stress tests for Full Integrated Scan circuits. These tests can be used during monitored and dynamic burn-ins. Optimal algorithm and fast greedy heuristic are compared in terms of quality of solution obtained and running times. We feel that the greedy heuristic or its variations can compute cyclic stress tests for large circuits where optimal algorithm takes too much time. This paper motivates the study for good heuristics for computing cyclic stress tests for dynamic burn-in problems.

References

[1] R. Parkar, "Bare die test," in *IEEE Multi-Chip Module Conference*, pp. 24–27, 1992.

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

[3] M. Campbell, "Monitored burn-in (a case study for in-situ testing and reliability studies)," in *IEEE International Test Conference*, pp. 518–523, 1984.

[4] H. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A systems perspective*. Addison-Wesley Publication Company, second ed., 1992.

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

[6] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *ACM/IEEE International Conference on Computer Aided Design*, pp. 402–407, 1992.

[7] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, Massachusetts: The MIT Press, 1991.

[8] 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.

[9] M. R. Gary and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Fransisco: W. H. Freeman, 1979.

[10] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits using boolean function manipulation," *IEEE Trans. Computer Aided Design*, vol. CAD-11, pp. 373–383, March 1992.

[11] H. Kriplani, F. Najm, P. Yang, and I. Hajj, "Resolving signal correlations for estimating maximum currents in CMOS combinational circuits," in *ACM/IEEE 30th Design Automation Conference*, pp. 384–388, 1993.

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

Transition	g_1	g_2	g_3	g_4
0001 → 1111	1	1	1	0
1111 → 0001	1	1	1	0
1011 → 1111	1	0	0	0
1111 → 1011	1	0	0	0
0000 → 1011	0	1	1	0
1011 → 0000	0	1	1	0
0000 → 0001	0	0	0	0
0001 → 0000	0	0	0	0
0000 → 1111	1	1	1	0
1111 → 0000	1	1	2	1
0001 → 1011	0	1	1	0
1011 → 0001	0	1	1	0

Table 2: Transitions for circuit in Figure1

Transition	cycle number	number of transitions
0000 → 1101	1	1
1101 → 1111	2	1
0000 → 0001	1	0
0001 → 1111	2	0

Table 3: Transitions at the output of g_4 when primary inputs are switched in the first cycle and in the second cycle

Transition	cycle number	Total number of transitions
0001 → 0000	1	0
0000 → 0000	2	0
0001 → 1100	1	3
1100 → 0000	2	3

Table 4: Total number of transitions in the circuit of Figure 1 in case of single switching and multiple switching

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

Table 5: c-nodes, s-nodes, h-nodes in ISCAS89 circuits

circuit	part1		part2		part3		part4	
	length	weight	length	weight	length	weight	length	weight
s208.1	34	2.50						
s298	10	40.11	10	16.96				
s344	22	51.00	14	1.17				
s349	2	42.13	26	1.73				
s382	8	32.81	36	39.48				
s386	80	5.25						
s420.1	76	3.19						
s444	36	31.98	20	43.95	32	6.90		
s510	65	0.00						
s526n	66	47.95	14	53.14	15	10.54		
s526	50	48.31	72	53.17	22	10.48		
s641	40	42.95						
s713	54	39.16						
s820	14	9.60						
s832	24	10.80						
s838.1	152	6.00						
s1423	14	16.90	16	16.87	76	56.15	76	10.04
s1488	130	35.83						
s1494	142	32.00						

Table 6: Maximum weighted cycles for s-nodes by optimal algorithm

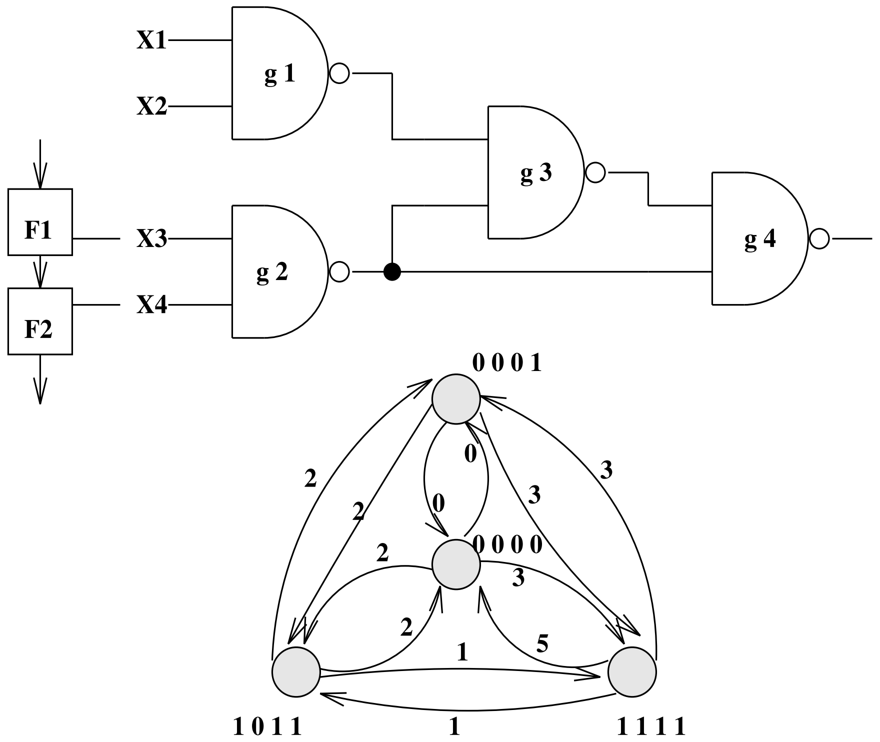

Figure 1: Full Integrated Scan Circuit and Transitions Graph

circuit	part1/5/9			part2/6/10			part3/7			part4/8		
	length	best	worst	length	best	worst	length	best	worst	length	best	worst
s208.1	2	2.50	2.12									
s298	2	35.68	33.18	2	16.96	14.91						
s349	2	42.13	38.50	2	1.67	1.63						
s382	2	31.26	27.97	2	39.38	39.38						
s386	2	4.83	4.17									
s420.1	2	3.09	2.73									
s444	2	30.14	28.07	2	43.95	41.57	2	6.90	6.17			
s510	2	0.00	0.00									
s526	2	47.88	44.44	2	51.45	46.86	2	10.48	9.48			
s526n	2	47.45	41.64	3	49.60	46.71	3	10.52	9.99			
s641	2	42.95	39.76									
s713	3	38.88	35.80									
s820	3	8.80	7.20									
s832	3	8.80	8.16									
s838.1	4	5.24	4.91									
s1423	2	16.49	15.33	2	16.43	15.42	3	54.55	51.93	2	9.92	9.57
s1488	2	33.75	31.94									
s1494	2	31.67	27.78									
s5378	4	45.24	45.24	3	65.06	65.06	2	51.03	51.03	2	58.15	58.15
	2	42.71	42.71	2	63.62	63.62	2	46.04	46.04	2	52.01	52.01
	2	50.32	50.32	4	68.10	68.10	3	32.38	32.38	3	59.27	59.27
	3	60.89	60.89	3	56.90	56.90	2	61.98	61.98	2	52.99	52.99
	4	52.07	52.07	3	31.28	31.28	4	51.89	51.89	2	76.78	76.78

Table 7: Results of the greedy algorithm on ISCAS89 circuits

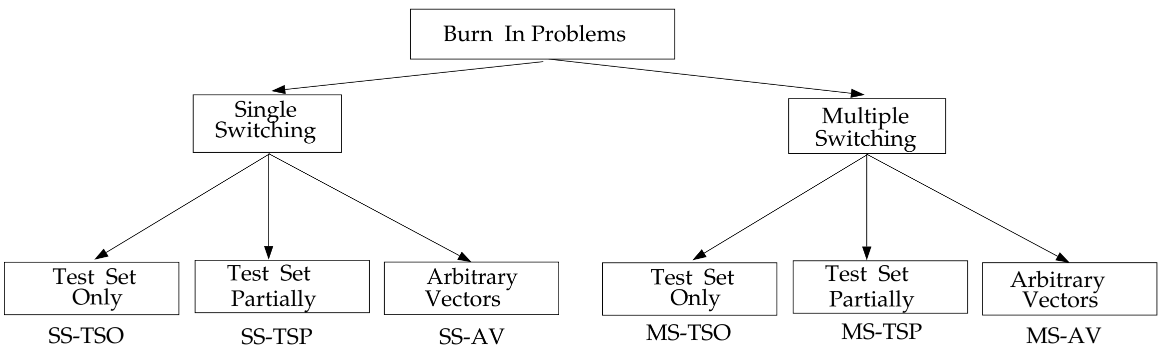

Figure 2: Taxonomy of Burn-In Problems

circuit	greedy best %	greedy worst %	optimal time	greedy avg time
s208.1	100.00	84.80	3.28	0.30
s298	92.24	84.26	34.41	3.10
s349	99.86	91.50	30.74	2.49
s382	97.72	77.38	73.74	6.60
s386	92.00	79.43	32.91	2.67
s420.1	96.87	85.58	59.13	5.12
s444	97.78	75.23	115.29	10.40
s526	98.08	87.85	463.44	31.90
s526n	96.36	83.55	399.65	29.44
s641	100.00	92.57	59.87	5.35
s713	99.28	91.42	47.21	4.10
s820	91.67	75.00	185.48	14.46
s832	81.48	75.56	208.34	15.64
s838.1	87.33	81.83	918.61	69.50
s1423	97.43	55.58	3568.37	224.73
s1488	94.19	89.14	474.57	36.83
s1494	98.97	86.81	546.12	40.99

Table 8: Performance and timing comparison between optimal and greedy algorithm (s-nodes)

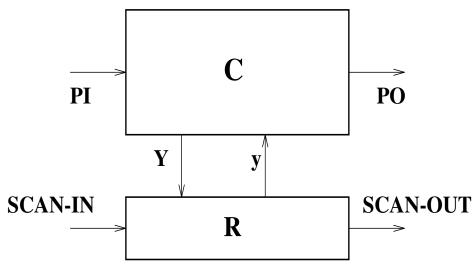

Figure 3: Scan Structures

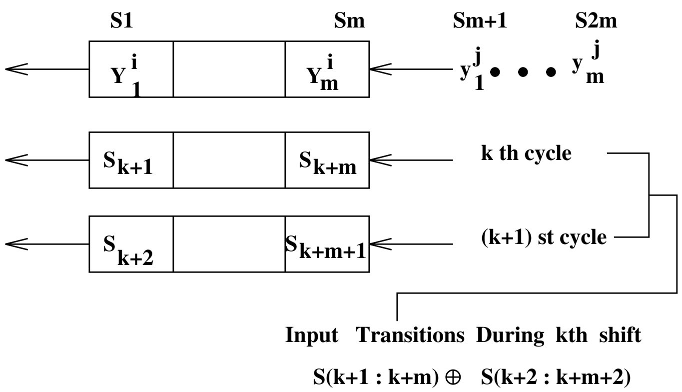

Figure 4: Scan Register Shifting

circuit	part1/5/9		part2/6/10		part3/7		part4/8	
	length	weight	length	weight	length	weight	length	weight
s208.1	33	12.96	35	8.50				
s298	36	16.39						
s349	4	21.63	6	9.53				
s382	34	28.38						
s386	78	19.50	78	15.50	21	4.44		
s420.1	74	9.25	76	17.44	76	11.97	75	1.75
s444	36	28.07	6	2.08				
s510	15	7.89						
s526	48	20.29	21	2.86				
s526n	16	20.55	68	2.99				
s641	38	33.82	60	30.97	4	26.37	60	32.24
	33	17.74	60	17.21				
s713	54	33.92	54	42.50	54	28.47	24	35.75
	26	23.53	26	22.45	54	5.12		
s820	120	13.70	12	15.20	122	0.00		
s832	80	13.10	124	15.40	128	0.00		
s838.1	152	9.62	152	10.23	152	19.83	152	14.34
	152	13.66	150	6.25	153	7.62		
s1423	12	23.89	76	22.43	76	27.28	16	30.03
	14	22.80	76	23.17	46	30.11	72	15.05
	14	13.99	16	2.91				
s1488	132	27.22	38	15.58	132	22.25	39	14.39
	36	14.44	38	17.58	132	15.17		
s1494	141	27.22	138	14.33	48	22.83	142	18.50
	48	13.42	138	15.33	142	18.00		

Table 9: Opitmal algorithm for h-nodes

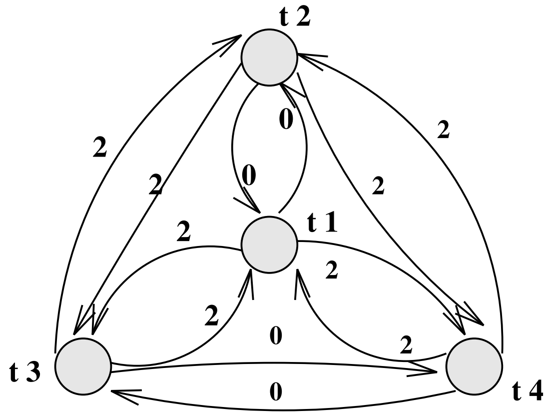

Figure 5: Transitions at gate g_2

circuit	part1/5/9			part2/6/10			part3/7			part4/8		
	length	best	worst	length	best	worst	length	best	worst	length	best	worst
s208.1	3	12.96	11.75	3	8.50	7.81						
s298	2	16.39	13.93									
s349	2	21.63	19.76	2	9.53	8.57						
s382	2	27.55	24.04									
s386	2	18.25	13.72	3	13.22	10.50	2	4.08	2.75			
s420.1	2	8.62	7.24	2	17.44	14.16	2	11.31	9.06	2	1.75	1.33
s444	2	28.07	25.14	3	2.08	2.02						
s510	3	7.17	5.50									
s526n	2	20.19	17.19	2	2.93	2.62						
s526	3	19.37	17.60	2	2.76	2.51						
s641	2	31.95	24.68	2	28.42	23.93	2	24.11	21.04	2	30.47	24.61
	2	16.68	14.72	2	16.84	14.14						
s713	3	33.47	25.87	2	42.50	29.58	2	26.37	22.82	2	35.00	29.09
	2	23.53	19.19	3	21.79	18.95	2	5.05	4.11			
s820	2	11.20	8.77	2	13.70	10.63	2	0.00	0.00			
s832	2	10.60	8.90	2	13.90	11.05	2	0.00	0.00			
s838.1	2	9.62	6.25	3	8.70	6.73	2	19.44	16.02	2	13.42	8.65
	2	13.36	11.70	2	5.55	4.40	2	7.45	6.47			
s1423	2	23.82	22.38	2	22.43	19.35	2	27.09	24.61	2	29.82	28.05
	2	22.80	18.97	2	22.88	18.39	2	29.74	27.80	2	13.79	12.43
	2	13.99	11.36	2	2.85	2.73						
s1488	2	27.17	19.27	2	12.42	8.92	2	19.25	13.87	2	13.50	9.47
	2	13.17	10.10	2	17.58	11.11	3	13.50	9.96			
s1494	2	24.50	16.96	3	12.11	10.25	2	20.00	13.25	2	16.00	11.00
	3	11.11	8.83	2	15.33	10.44	2	12.92	9.92			

Table 10: Maximum weighted cycles for h-nodes by greedy algorithm

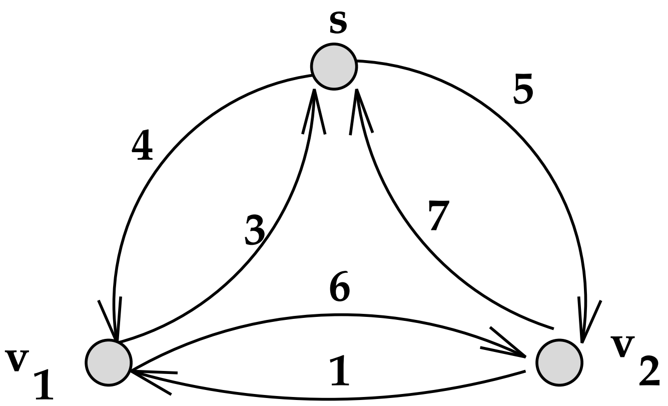

Figure 6: Graph for the illustration of optimal algorithm

circuit	greedy best %	greedy worst %	optimal time	greedy avg time
s208.1	100.00	91.15	9.41	0.92
s298	100.00	84.99	12.76	1.19
s349	100.00	90.92	15.04	1.37
s382	97.08	84.71	18.59	1.75
s386	90.14	68.38	84.77	7.24
s420.1	96.81	57.04	346.75	30.16
s444	100.00	90.08	43.73	4.05
s526n	98.22	84.15	218.32	19.27
s526	95.59	86.87	247.73	21.86
s641	93.76	75.50	465.69	42.39
s713	97.90	72.82	465.75	44.12
s820	86.16	60.69	369.94	28.77
s832	85.96	62.46	406.26	30.73
s838.1	95.08	49.11	8502.12	652.00
s1423	98.84	76.87	13314.53	1145.74
s1488	92.07	57.48	2427.41	188.87
s1494	86.38	57.24	2767.86	209.40

Table 11: Performance and timing comparison between optimal and greedy algorithm (h-nodes)