Stress Tests for Dynamic Burn-in of Full Scan Circuits

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

Abstract

Dynamic burn-in 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. We show that the problems are computationally difficult. In addition, we demonstrate experimentally that good heuristics can be developed to compute stress test for dynamic burn-in of full scan circuits.

1 Introduction

Growing size and complexity of VLSI circuits is making production of reliable chips a challenging task. Environmental stress testing is an effective method for improving product reliability [1]. Products often have hidden defects which cause field failures. Of particular concern are failures that occur soon after the end user starts using it ("infant mortality"). Stress testing is a process in which stress stimuli are applied to products to expose such latent defects.

Stress tests can be applied in a variety of ways such as burn-in, power cycling, temperature cycling, voltage variations, clock variations etc [1]. *Burn-in* is the process of testing a product for an extended period at an elevated temperature. It is one of the most common form of stress testing [2, 1]. 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 [3].

Dynamic burn-in and monitored burn-in are widely used [4, 5]. One of a number of scenarios where burn-in is used is depicted in Figure 1. This figure, taken from [5], depicts MCM production flow. Highlighted blocks show the places where burn-in and testing are performed. It can be seen that burn-in is performed at two places, once on the bare die and then on the assembled module. In addition to MCMs, burn-in used to separate *infant mortalities* in many other applications [1].

The objective of this research is to compute good stress tests that are to be used during burn-in. We assume that during burn-in stress conditions are induced by raising temperature and applying good set of vectors that generate high switching activity in the targeted part of the circuit. We had proposed the use of cyclic sequences [6] which makes it possible to apply them repetitively for an extended period. The idea is to maximize switching activity during application of these cyclic sequences. The issue that we are interested is how to compute such a sequence.

A systematic study of the problem of computing cyclic sequences in Full Scan circuits to maximize average switching activity is presented in [6]. In addition to presenting a taxonomy of burn-in problems, it is shown in [6] 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 in [6] for computing stress tests for monitored burn-in.

Monitored burn-in has several drawbacks. (i) It requires a tester throughout the burn-in process. The cost of tying down a tester for a long period could be prohibitively large. (ii) 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. (iii) Since selection of vectors is not restricted to test vectors in dynamic burn-in, higher level of activity can be can be generated.

In this paper, we focus on computing good cyclic stress tests for dynamic burn-in. First, we show 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. This is done by classifying the nodes of the circuits and developing different heuristics for each class of nodes. We show that these heuristics are faster and work considerably better than the optimal algorithm used for monitored burn-in [6].

The paper is organized as follows. Section 2 gives the background and notation used in the paper. In section 3, measure of goodness used to compare the cyclic sequences is discussed along with the problem definition. Heuristics for computing stress tests are presented in section 4 and experimental results are discussed in section 5. Intractability of dynamic burn-in problems is discussed in Appendix.

2 Background

2.1 Power Dissipation Model

We briefly describe the power dissipation model used in the paper. For more details please refer to [6]. Let P_d be the power required to charge and discharge the output capacitive load of every gate. P_d is approximated as

$$P_d = 1/2 \times c_i \times v_{dd}^2 \times n_i \tag{1}$$

where c_i = output capacitance of gate i; n_i = number of transitions ($1 \rightarrow 0$ or $0 \rightarrow 1$) at the output of gate i per unit time.

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 a circuit C when inputs change from t_i to t_{i+1}. Then,

$$P_C(t_i, t_{i+1}) = \sum_{j \in SetofNodes} 1/2\, c_j \times v_{dd}^2 \times n_j \tag{2}$$

Thus power dissipated at a node is proportional to the number of transitions per unit time (n_j) at that node. This depends on the gate delays and sequence of input vectors applied. The former is approximated by delay models. We use zero delay model under which all gates are assumed to have zero delay.

2.2 Power Dissipation in Full Scan Circuits

Block diagram of Full Integrated Scan is shown in Figure 2 [7]. 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. Equations that take into account the power dissipation during monitored burn-in have been derived in [6]. They can be easily modified for dynamic burn-in. For the sake of completeness we briefly describe the equations in Appendix A.

We divide the circuit nodes into different categories whose characteristics is exploited while constructing cyclic sequences that maximize switching activity at their output. Consider a node that has only primary inputs in its fanin. The output of this node does not depend on the state variables. We call such a node *c-node*. Nodes which have only scan latches in their fanin are called *s-nodes* and nodes having both, primary inputs and scan latches in their fanin are called *h-nodes*(meaning hybrid nodes).

3 Problem Definition

Determining measure of goodness of a cycle is an important issue. In [6], average number of nodes switched per clock of the cyclic sequence was used as the measure. It 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 will lead to a 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 [6], 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 average activity in a node ? In other words, a threshold for good average activity needs to be determined such that nodes with average activity below this threshold could be said to have poor switching. 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 in 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.

In this paper, we propose a two threshold approach. We associate two thresholds t_1 and t_2 with every cyclic sequence C. This divides the targeted nodes into three categories; (1) nodes with average activity less than t_1; (2) nodes with average activity between t_1 and t_2 and; (3) nodes with average activity more than t_2. We denote the three sets as $m_1(C)$, $m_2(C)$ and $m_3(C)$ respectively. We say that cyclic sequence C_1 is better than C_2 if either $m_1(C_1) < m_1(C_2)$ or if $m_1(C_1) = m_2(C_2)$ and $m_2(C_1) < m_2(C_2)$.

We would like to compute a cyclic sequence such that maximum number of nodes have switching activity more than t_2 and possibly all the nodes have switching activity more than t_1.

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 a best cyclic sequence is NP-hard. The proofs of these propositions are given in Appendix B.

Notwithstanding these difficulties 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 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 nodes pair 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 4.5-4.18) 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 pair (c, v) switches (step 4.8). These nodes are classified into 10 Bins (Bin[1]-[10]). Bin[1] stores the number of nodes that pair (c, v) switches and having average switching activity between 0 and 0.1 Bin[2] stores the number of nodes with activity between 0.1 and 0.2 and so on. (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 4.10 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 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)
(4.9) /* Evaluate the vector v */
(4.10) weight $\leftarrow \sum_{i=1}^{10}$ BinWeight$[i] *$ Bin$[i]$
(4.11) **if** (weight $>$ MaxWeight)
(4.12) MaxWeight $\leftarrow$ weight
(4.13) BestVector $\leftarrow v$
(4.14) **endif**
(4.15) **endfor**
(4.16) UpdateNodeTransitions (CurrentVector, BestVector)
(4.17) CurrentVector $\leftarrow$ BestVector
(4.18) **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 [8]. 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. 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 3 and Table 4. 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 run 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 given by optimal algorithm for generation cyclic stress tests for monitored burn-in described in [6] with k-lookahead heuristics. Results obtained by the Monitored burn-in heuristic are shown in Table 5. It can be seen that in general lookahead heuristics give 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. Another important point to note is the running time for monitored burn-in heuristic. It takes time order of magnitude more than that of lookahead heuristics. In fact for circuit s5378 cycles were obtained for only 3 out of 24 groups of s-nodes and that took 5640 minutes. On the other hand, reasonable 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 5.

Results obtained for h-node heuristic are presented in Table 7. By observing the second column of Table 7 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 are presented to compute good cyclic sequences. 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 will be directed towards computing better cycles for h-nodes.

References

[1] H. A. Chan, P. J. Englert, M. A. Oien, and S. Rajaram, "Environmental stress testing," *AT&T Technical Journal*, pp. 77–85, April 1994.

[2] F. Jensen and N. Petersen, *Burn-in: An Engineering Approach to the Design and Analysis of Burn-in Procedures*. John Wiley & Sons, 1982.

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

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

[5] Y. Zorian, "Tutorial: MCM testing strategies," in *International Test Conference*, 1995.

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

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

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

Appendix A: Power dissipation in c-nodes, s-nodes, and h-nodes

Power dissipated in c-nodes, s-nodes and h-nodes can be given by the following equations.

(1) Power dissipated 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 power dissipated 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) Power dissipated 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, power dissipated during the shifting of complete cycle is:

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

(3) Power dissipated 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

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

Appendix B: Intractability of Dynamic Burn-in Problems

Solutions to the monitored burn-in problems can be used as stress tests during dynamic burn-in. However, it is possible to obtain better stress tests for dynamic burn-in when choice of vectors is not restricted to the given test set. To illustrate this, consider a c-node in a full-scan circuit. 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 [7] or; (2) it may be not be possible to propagate the fault 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 pair of stuck-at test vectors may not exist. This motivates the study of generating activity at the output of such c-nodes. However, we show that the problems of computing cyclic stress tests are NP-hard for c-nodes, s-nodes and h-nodes. All these problems can be shown to be as hard as satisfiability problem. For the sake of completeness, we give the formulation of satisfiability problem [9].

Given a product of sum formula $\mathcal{F}$ consisting of m clauses and a set of n literals $\mathcal{U}$. Let each clause Cl_i have k_i literals such that each literal is of the form x or $\overline{x}$ where $x \in \mathcal{U}$. $\mathcal{F}$ can be represented as

$$\begin{aligned} \mathcal{F} &= \bigwedge_{i=1}^{m} Cl_i, \\ Cl_i &= \bigvee_{j=1}^{k_i} x_j^i \text{ such that } x_j^i \in \mathcal{U} \vee \overline{x_j^i} \in \mathcal{U} \end{aligned}$$

SAT : Given a boolean formula $\mathcal{F}$ of the above form, find an assignment $A \in \{1, 0\}^n$ to literals $x_1, \ldots, x_n$ such that $\mathcal{F}$ is satisfied.

Stress Tests for c-nodes We study the time complexity of generating stress tests for dynamic burn-in for a given c-node in a full-scan circuit.

$\mathcal{D}_{\mathsf{c-node}}$**:** Given a full scan circuit Q and a c-node x, compute a pair of vectors (v_1, v_2) such that $P_x(v_1, v_2)$ is maximized.

Proposition: Problem $\mathcal{D}_{\mathsf{c-node}}$ is NP-Hard.

Proof: Given an instance of SAT, an instance of $\mathcal{D}_{\mathsf{c-node}}$ can be constructed as follows. Given a CNF formula $\mathcal{F}$, a boolean circuit $\mathcal{C}$ is constructed as shown in Figure 3. $\mathcal{C}$ contains m OR-gates - one for each clause Cl_i in $\mathcal{F}$ and one AND-gate corresponding to the conjunction of the clauses. Each OR-gate has the literals in the corresponding clause as the inputs.

Consider the output node f of $\mathcal{C}$. We show that the problem of finding a pair of vectors (v_1, v_2) such that $P_f(v_1, v_2)$ (i.e. activity at the c-node f) is maximized is at least as hard as finding a satisfying assignment for $\mathcal{F}$. This is same as showing that if a solution to the instance of $\mathcal{D}_{\mathsf{c-node}}$ is obtained, solution to the instance of SAT can be obtained in polynomial time. Consider a solution (v_1, v_2) of $\mathcal{D}_{\mathsf{c-node}}$. If $P_f(v_1, v_2)$ is 1 then either v_1 or v_2 should set f to 1 which means $\mathcal{F}$ is satisfiable. On the other hand, if $P_f(v_1, v_2)$ is 0 then there are only two possibilities: either $\mathcal{F}$ evaluates to true for *all* vectors $v \in \{0, 1\}^n$ (in which case $\mathcal{F}$ is a tautology) or $\mathcal{F}$ evaluates to false for *all* vectors $v \in \{0, 1\}^n$. In either case, choose any vector, say $v = 0^n$. If $\mathcal{F}(v)$ is true then $\mathcal{F}$ is satisfiable. If it is false then it is not satisfiable. This concludes the proof. □

Note that in case of multiple c-node case the problem is at least as hard as the single c-node case.

Stress Tests for s-node Consider a vector v of length n. Number the n elements from left to right by 0 through $n-1$ and let $v[i]$ denote the i^{th} element of v. A non-negative integer p, $p < n$ is a *period* of v if $v[i] = v[i+p]$, for $i \in [0, n-p)$. In terms of scan-in/scan-out, a vector with period p, after p cyclic shifts results in the same vector in the scan register. By definition, an n element vector has period n. For example, vector 010010 has periods 3 and 6. Next, let $P_x(v^i)$ denote number of transitions at node x during the i^{th} cyclic shift of v.

$\mathcal{D}_{\mathsf{s-node}}$**:** Given a full-scan circuit Q, and an s-node x, find a vector v with a period p such that average number of switching at x, given by the following term, is maximized.

$$P_x(v) = \frac{1}{p}\sum_{i=1}^{p} P_x(v^i)$$

Next we show that $\mathcal{D}_{\mathsf{s-node}}$ is NP-Hard. We show this by showing a special case of $\mathcal{D}_{\mathsf{s-node}}$ is NP-Complete. Consider the following decision problem which is a special case of this problem.

$\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$**:** Consider a full-scan circuit Q and an s-node x. Does there exist a vector v such that one shift of v with appropriate shift-in (0 or 1) switches x ?

Note that this problem is indeed a special case of $\mathcal{D}_{\mathsf{s-node}}$. If average activity of the optimal solution is non-zero then there must exist one vector whose shift switches x. Hence answer to $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is affirmative. Similarly, if the average activity of the optimal solution is zero then the answer to $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is no. Next, we show $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is NP-Complete.

Lemma: $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is NP-Complete.

Proof: It can be seen that given a vector v it can be verified in polynomial time whether v is a solution to $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ and hence the problem belongs to $\mathcal{NP}$. We reduce SAT to $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ and show $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is at least as hard as SAT. Given an instance of SAT, we construct an instance of $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ similar to the one shown in Figure 3 with a few differences. Each input is associated with a scan latch and all scan latches are connected to form a scan chain. The order of the scan latches is not important. Note that f is an s-node under this construction.

Let the solution to the instance of $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ be affirmative. In that case, there exists at least one vector v such that shifting in either 0 or 1 switches the s-node f. Then $\mathcal{F}$ is satisfiable since either v or v with one shift should satisfy $\mathcal{F}$. Now consider the other case where solution to $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is negative. Note that any vector v can be obtained from any other other vector v' by shifting it at most n times. Hence there cannot be any disjoint partitions among the vectors such that vectors in one partition after shifting produce vectors in the same partition. This leaves only two possibilities similar to the proof of proposition in section 6.1. viz. (i) $\mathcal{F}$ is tautology and (ii) Complement of $\mathcal{F}$ is tautology. And in either case it can be easily found out whether $\mathcal{F}$ is satisfiable. Thus $\mathcal{D}^{\mathsf{special-case}}_{\mathsf{s-node}}$ is at least as hard as SAT. $\square$

From the Lemma follows the result:

Proposition: $\mathcal{D}_{\mathsf{s-node}}$ is NP-Hard.

Intractability of h-node case is obvious since c-node and s-node cases are special cases of h-node case.

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 1: 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 2: c-nodes: Heuristic-1

circuit	30	30-60	60	avg cycle	time
				length	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

Table 3: s-nodes: Heuristic-2 with 2-lookahead

circuit	30	30-60	60	avg cycle	time
				length	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 4: s-nodes: Heuristic-2 with 4-lookahead

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 5: s-nodes: Monitored burn-in results

circuit	30	30-60	60	avg cycle	time
				length	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 6: s-nodes: Combining monitored burn-in and Heuristic-2

circuit	30	30-60	60	avg cycle	time
				length	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	-

Table 7: h-nodes: 2-lookahead heuristic

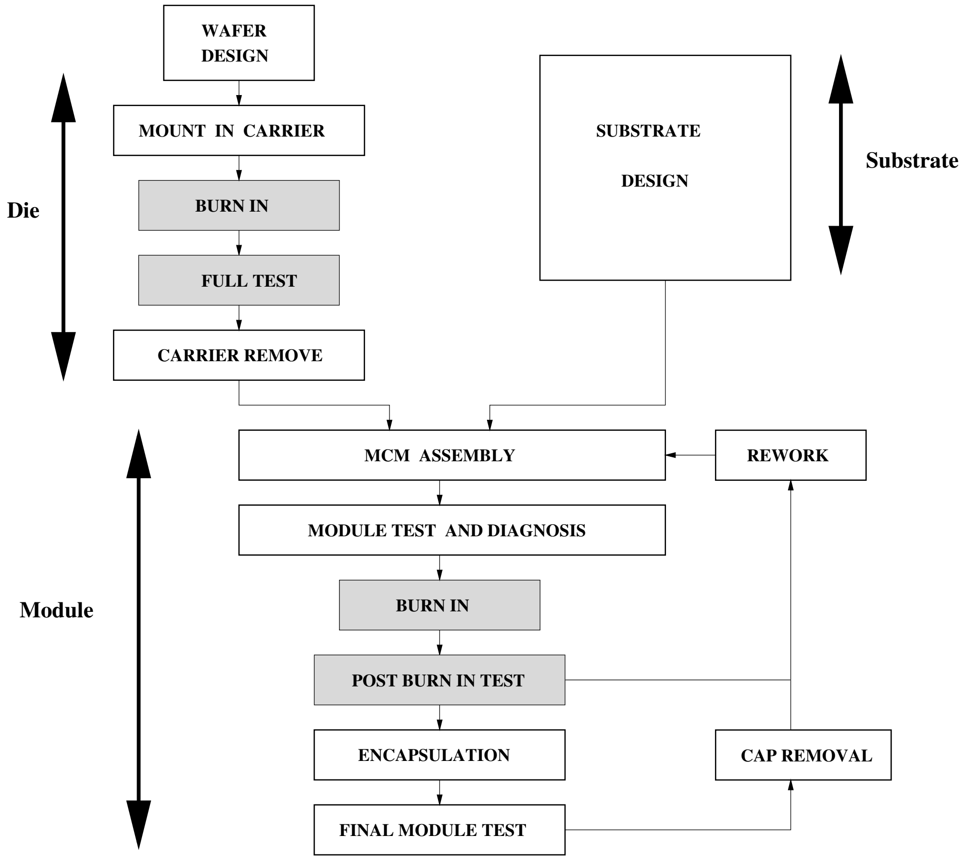

Figure 1: MCM Production Flow

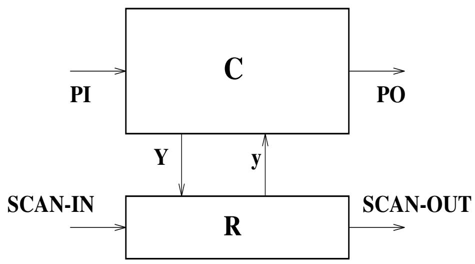

Figure 2: Scan Structures

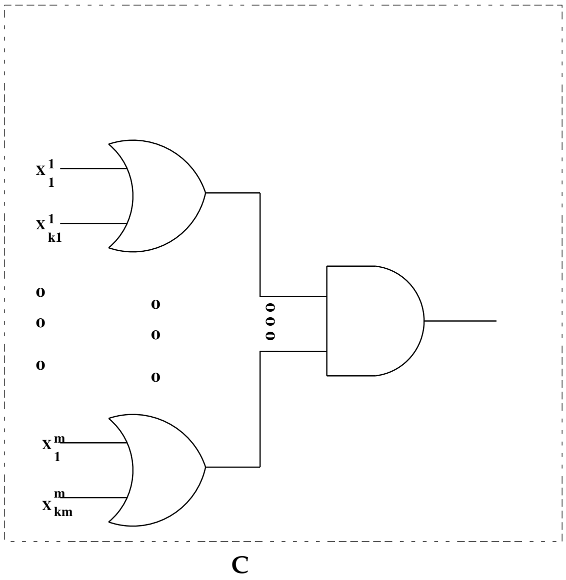

Figure 3: Instance of $\mathcal{D}_{\mathsf{c-node}}$