Simultaneous Test Pattern Compaction, Ordering and X-Filling for Testing Power Reduction

Ju-Yueh Lee[1], Yu Hu[1], Rupak Majumdar[2] and Lei He[1]
1. Electrical Engineering Department
2. Computer Science Department
University of California, Los Angeles *

ABSTRACT

Minimizing the power dissipation in scan-based testing is an important problem. We provide for the first time an optimal formulation for the problem of simultaneously compacting, ordering, and X-filling a set of test patterns such that the fault coverage is maintained but the (overall or peak) power dissipation is minimized. We model the problem as a sequence of Pseudo-Boolean optimization problems. We give a scalable implementation of the optimization problem based on window-based local search. In contrast to the traditional technique of sequentially optimizing for compaction, ordering, and X-filling, we experimentally demonstrate that our simultaneous optimization can reduce power dissipation by 47% on ISCAS'89 benchmark circuits presented heuristics to minimize test power by simultaneously test pattern ordering and X-filling.

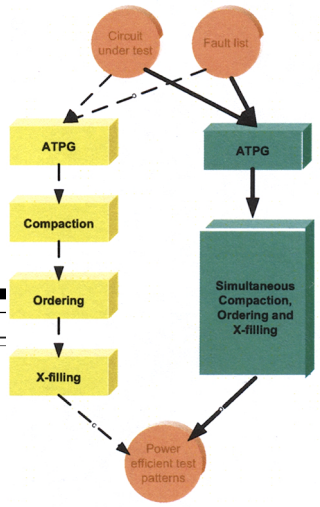

Figure 1: Conventional flow (left branch) vs. proposed flow (right branch) for low power ATPG

1. INTRODUCTION

The power dissipation of integrated circuits (ICs) in scan-based testing can be several times higher than that in functional modes [1, 2], for example, due to simultaneous testing of multiple modules which do not operate at the same time in functional mode or due to high switching rates during scan shift/capture. With the shrinking geometries and lower voltage thresholds in modern circuits, this excessive power dissipation during test can have an impact on digital IC reliability, resulting in power-driven failures and false failures at final test [3]. For example, excessive power dissipation can cause structural damage to the silicon or the package, and excessive peak power dissipation can cause large voltage drops which lead to erroneous data in test mode only. This has led to substantial research effort in minimizing power dissipation during scan testing.

Numerous algorithms have been presented for test mode power reduction [4, 5, 6], and surveys for recent advances in DFT and test pattern generation for test power optimization can be found in [7, 8]. Particularly, various optimizations are proposed for test pattern compaction, ordering, or X-filling, which are three major design freedoms in the power-aware test pattern generation. In order to reduce test data volume when testing, static test pattern compaction has been proposed [9] to reduce both test data volume and power dissipation. The power-aware test pattern ordering problem has been attacked by [10] for the combinational circuit testing and by [11] for the scan-chain based sequential circuit testing. In addition, power-aware test pattern X-filling techniques have been presented by [12, 13, 14, 6], which minimize capture power by filling don't-care bits to minimize the bit difference between pseudo primary inputs (PPIs) and pseudo-primary outputs (PPOs). Moreover, [15]

*This research is partially supported by the NSF awards CCR-0306682, CNS-0702881 and CCF-0702743, and a UC MICRO grant sponsored by Actel. Address comments to lhe@ee.ucla.edu.

However, most previous papers perform the three optimizations for ATPG patterns —compaction, ordering, X-filling— sequentially (see Figure 1, left branch). This sequential process can cause loss of optimality: the power dissipation of the resulting solution can be higher than the power dissipation of an "optimal" test sequence, even if each phase is performed optimally. In this paper, we present an *optimal* algorithm for ATPG power minimization, by *simultaneously* optimizing compaction, ordering, and X-filling for (peak or overall) power dissipation. The proposed simultaneous compaction, ordering, X-filling (COX) algorithm can be applied right after a conventional ATPG tool, and it preserves the test coverage with negligible impact on the test data volume and test time. Our algorithm is based on Pseudo-Boolean (PB) optimization, and we propose local window-based heuristics to cope with the high runtime complexity of the PB optimization problem. Experimental results show that the proposed COX algorithm achieves 47% power saving compared to the conventional sequential power-aware ATPG flow. A comparison between the conventional power-aware ATPG flow and the proposed flow is as shown in Figure 1.

The remainder of this paper is organized as follows. Section 2 introduces background concepts. Section 3 presents the problem formulation and the proposed algorithms. Section 4 gives the exper-

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.

imental results and the paper is concluded in Section 5. To the best of our knowledge, this is the first work to consider simultaneous compaction, ordering, and X-filling for test mode power reduction.

2. PRELIMINARIES

2.1 Scan-Based Testing

Figure 2 illustrates basic scan testing of circuits. Testing happens in two modes: scan shift mode and capture mode. During the scan shift mode, a test vector is shifted into the scan flip flops (SFFs) serially. After the whole test vector is shifted into position, the circuit under test (CUT) turns into capture mode, in which the primary inputs (PIs) and pseudo-primary inputs (PPIs) from SFFs are fed into the combinational portion of the circuit and the response is propagated to the primary outputs (POs) and pseudo-primary outputs (PPOs). After capture, the CUT again runs in scan shift mode, this time shifting out the response captured by the SFFs and shifting in the next test vector at the same time.

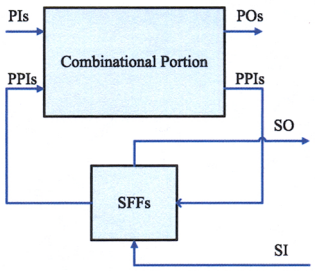

Figure 2: Scan-chain based testing procedure.

Given a circuit with K primary inputs, $\overrightarrow{I} = (i_1, \cdots, i_K)$, N gates $G = g_1, \cdots, g_N$, and a set of stuck-at faults $F = \{f_1, \cdots, f_n\}$, an ATPG tool can produce a set of test patterns, $T = \{\overrightarrow{t_1}, \cdots, \overrightarrow{t_m}\}, (m \leq n)$, which detects all faults in F. Each test pattern, $\overrightarrow{t_i}$, is a sequence of K ternary values

$$\overrightarrow{t_i} = (t_i^1, t_i^2, \cdots, t_i^K),$$

where t_i^j denotes the value for jth input bit in the ith test pattern, which can be the constant 0, the constant 1, or a don't-care x. A typical ATPG tool, e.g., ATALANTA, usually generates a set of test patterns with don't-cares, which can be assigned to either 0 or 1 without affecting the detection of faults.

There is a natural ordering on test patterns defined as follows. A test pattern $\overrightarrow{t_a}$ is *covered* by test pattern $\overrightarrow{t_b}$, written $\overrightarrow{t_a} \dashv \overrightarrow{t_b}$, iff

$$t_a^j = t_b^j \text{ for all } t_a^j \neq \text{x}. \quad (1)$$

For example, test pattern "0xx1" covers test pattern "00x1" but not "10x1". If two test patterns cover each other, they are *compatible. Test pattern compaction* is the problem of reducing the number of test patterns without affecting the fault coverage. This can be achieved by merging compatible test patterns and reducing redundant test patterns (i.e., patterns which are covered by other test patterns).

A test vector $\overrightarrow{s_i} \in \{0,1\}^K$ is an instantiation of a test pattern $\overrightarrow{t_i} \in \{0,1,\text{x}\}^K$ if $\overrightarrow{s_i}$ can be obtained by filling all don't-cares in $\overrightarrow{t_i}$. This operation is called *X-filling*. It follows that $\overrightarrow{s_i} \dashv \overrightarrow{t_i}$. For example, possible X-fillings for test pattern "0xx1" are "0001", "0011", "0101", "0111". Since the scan-in power depends on the number of bit switches in each test vector, different X-filling may result in significantly different switching power. In this example, when the last bit of the test pattern is shifted in, "0101" has 1 bit switch more than the other three vectors.

In scan-chain-based testing, a sequence of test vectors (derived from test patterns generated by an ATPG tool) are scanned in sequentially to detect a specific set of faults. The procedure to decide the order of these test vectors applied during the scan-chain testing is called *test pattern scheduling*. For example, consider the following four test patterns "0110", "1001", "0100", "1000". Among 4!=24 different orders, the one leads to the least bit switches is 0100 → 0110 → 1001 → 1000, which has only 1 bit switch (0110 → 1001) between consecutive test patterns. On the other hand, the one leads to the most bit switches is 0100 → 1000 → 1001 → 0110, which has 3 bit switches (every transition contains a bit switch).

2.2 Power Modeling

Assuming that the clock period is sufficiently small, it is sound to interpret the average dynamic power dissipation over a clock cycle as the instantaneous dynamic power during that clock cycle [16]. In the remainder of this paper, instantaneous dynamic power is simply referred to as power. Also, in our power modeling we assume that the SFFs in Figure 2 are scan-hold flip-flops (SHFFs)[17], which can reduce scan shift power by isolating the combinational portion during scan shifting.

DEFINITION 1. *(Scan-in Register Energy) The switching energy consumed by scan-chain registers during the course of scan-in of the test vectors, $\overrightarrow{s_1}, \cdots, \overrightarrow{s_n}$, is calculated as:*

$$E_{reg}(\overrightarrow{s_1}, \cdots, \overrightarrow{s_m}) = \sum_{i=1}^{m} \sum_{j=1}^{K-1} w_{ij} \cdot |s_i^j - s_i^{j+1}|, \quad (2)$$

where w_{ij} is the weight to characterize the loading capacitance of the scan-chain registers and the weighted transition count (WTC) [18], which has been normalized based on the K.

DEFINITION 2. *(Capture Energy) The switching energy consumed during the course of the test vector change, e.g., $\overrightarrow{s_i} \rightarrow \overrightarrow{s_{i+1}}$, is calculated as:*

$$E_{cap}(\overrightarrow{s_i} \rightarrow \overrightarrow{s_{i+1}}) = \sum_{g \in G} C_g \cdot |v_g(\overrightarrow{s_i}) - v_g(\overrightarrow{s_{i+1}})| \quad (3)$$

where C_g is the loading capacitance of gate or register g in the circuit which is calculated based on Weighted Switching Activity (WSA) [19]. $v_g(\overrightarrow{s_i})$ and $v_g(\overrightarrow{s_{i+1}})$ are logic values at gate g under test vector $\overrightarrow{s_i}$ and $\overrightarrow{s_{i+1}}$, respectively. For a circuit and a given test vector $\overrightarrow{s_i}$, the logic value $v_g(\overrightarrow{s_i})$ of each gate g can be obtained by performing one pass of logic simulation.

Suppose the number of test vectors remaining after test compaction is $\hat{m}$. The *overall testing power dissipation* for test vector set $\overrightarrow{s_1}, \cdots, \overrightarrow{s_{\hat{m}}}$ is

$$P(\overrightarrow{s}, \hat{m}) = \frac{E_{reg}(\overrightarrow{s_1}, \cdots, \overrightarrow{s_{\hat{m}}}) + \sum_{i=1}^{\hat{m}-1} E_{cap}(\overrightarrow{s_i} \rightarrow \overrightarrow{s_{i+1}})}{\hat{m}}. \quad (4)$$

The *peak testing power dissipation* for test vector set $\overrightarrow{s_1}, \cdots, \overrightarrow{s_{\hat{m}}}$ is

$$P(\overrightarrow{s}) = \max\Big(\max_i \Big(\sum_{j=1}^{K-1} w_{ij} \cdot |s_i^j - s_i^{j+1}| \Big), \max_i (C_g \cdot |v_g(\overrightarrow{s_i}) - v_g(\overrightarrow{s_{i+1}})|) \Big) \quad (5)$$

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.

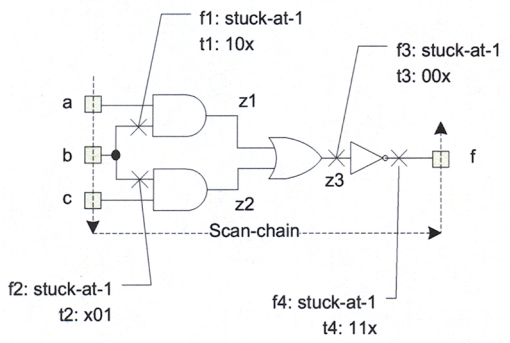

Figure 3: A sub-circuit (assuming inputs and output are registered and can be scanned for testing) to compute $f(a,b,c) = \bar{b} + \bar{a} \cdot \bar{c}$ and the test patterns generated by the ATPG tool (ATALANTA) to detect 4 stack-at faults.

Our goal is to minimize the overall or peak testing power dissipation while retaining fault coverage.

2.3 Pseudo-Boolean Constraint Problem

A *Pseudo-Boolean (PB) Constraint* [20] is an inequality $C_0p_0 + C_1p_1 + \cdots + C_{n-1}p_{n-1} \geq C_n$, where p_i is a literal and C_i is an integer coefficient for each i. A true literal is interpreted as value 1, a false literal as 0. An *objective function* is a sum of weighted literals on the same form as PB constraints. The *Pseudo-Boolean Constraint Problem* is to find a satisfying assignment to a set of PB-constraints that minimizes a given objective function.

A PB-constraint problem can be solved by general 0-1 integer linear programming (ILP) solvers (e.g., mosek [21]) or by dedicated SAT-based PB-solvers represented by minisat+ [22]. Our experimental results show that the PB-based problem formulation proposed in this paper is more favorable to PB-solvers than general ILP solvers because of its structural nature.

3. PROBLEM FORMULATION

3.1 COX Problem and Motivation

The simultaneous test pattern compaction, ordering and X-filling (COX) problem is formulated as follows.

FORMULATION 1. *(COX problem) Given a set of test patterns $\overrightarrow{t_1}, \cdots \overrightarrow{t_m}$, find an ordered set of test vectors $\overrightarrow{s_1}, \cdots \overrightarrow{s_{m'}}$ such that (1) each $\overrightarrow{s_i} \in \{0,1\}^K$ (i.e., contains no* x *terms), (2) $m' \leq m$, and (3) fault coverage is preserved (i.e., for each $\overrightarrow{t_i}$ there is some $\overrightarrow{s_j}$ such that $\overrightarrow{s_j} \dashv \overrightarrow{t_i}$), and (4) the testing power dissipation objective (either overall dissipation or peak dissipation) is minimized.*

The following example shows that *simultaneous* test pattern compaction, ordering and X-filling can result in lower overall power dissipation than traditional, *sequential* optimizations. Figure 3 shows a simple combinational circuit, with four stuck-at faults, $f_1, \cdots, f_4$, and four test patterns $\overrightarrow{t_1}, \cdots, \overrightarrow{t_4}$, to detect them. For the sake of simplicity, only the scan-in power is considered for minimization.

According to Table 1, which shows the fault coverage for each test pattern, one test pattern may cover multiple faults, and therefore test pattern compaction is available in this example. Based

test pattern	fault coverage
$\overrightarrow{t_1} = 10x$	f_1, f_2
$\overrightarrow{t_2} = x01$	f_1, f_2, f_3
$\overrightarrow{t_3} = 00x$	f_2, f_3
$\overrightarrow{t_4} = 11x$	f_4

Table 1: Fault coverage of each test pattern for the circuit shown in Figure 3.

on the test pattern compaction algorithm presented in [9], a compatibility graph (Figure 4(a)) can be first constructed to represent the fault coverage relationship among different test patterns and the test pattern compaction can be done by merging compatible test patterns, which results in one test pattern reduction as shown in Figure 4(b). After the compaction, don't-cares in each test patterns are filled (e.g., using minimum transition fill (MT-fill) [9] shown in Figure 4(c)) so that the total number of bit switches inside each test pattern is minimized. The ordering is then performed by solving a Traveling Salesman Problem (TSP) on the complete graph shown in Figure 4(c), where the weight on an edge denotes the number of switch between two X-filled test patterns [15]. The resulting ordering is shown in Figure 4(d), i.e., the scan-in sequence is $111 \rightarrow 101 \rightarrow 000$. In contrast, the best solution found by the simultaneous approach (which is detailed in Section 3.2) is shown in Figure 4(e), i.e., the scan-in sequence is $001 \rightarrow 111 \rightarrow 100$.

Now we evaluate the power dissipation by the test vectors returned by the sequential approach and the simultaneous approach, respectively. For the sake of simplicity, the scan-in energy can be approximated by the total number of bit switches in the test sequence, which includes both the internal the bit switches in each test vector and the mutual bit switches between two adjacent test vectors. It is seen that the test sequence generated by conventional sequential flow (as mentioned above) has 3 bit switches but the simultaneous optimization has 2 bit switches. Intuitively, the simultaneous optimization algorithm searches a larger solution space which leads to better results, compared to the conventional sequential approach, even if each individual optimization step (i.e., compaction, ordering and X-filling) is optimal. Our experimental results confirm that the proposed simultaneous algorithm can reduce power (on the average by 47% on our benchmarkes) compared to the conventional sequential approach.

3.2 COX Algorithm

We now formulate the simultaneous test pattern compaction, ordering and X-filling problem to a Pseudo-Boolean (PB) optimization problem [20], which results in an optimal test vector generation for power minimization. We will show that the proposed PB-based formulation is general enough to consider various objectives for power minimization.

The COX problem for minimizing overall test power dissipation is formulated as the following optimization problem. The variables in the problem are s_i^j for $i = 1, \ldots, m$ and $j = 1, \ldots, K$. The optimization problem is:

$$\begin{array}{lll} \text{minimize} & P(\overrightarrow{s}, \hat{m}) & \\ \text{subject to} & \text{CNF}_G(\overrightarrow{s_i}) & \text{for } i = [1,m], \\ & \bigvee_{j=1}^{\hat{m}} (\overrightarrow{s_j} \dashv \overrightarrow{t_i}) & \text{for } i = [1,m], \\ & s_i^j \in \{0,1\} & \text{for } i = [1,m], j = [1,K], \end{array} \tag{6}$$

where $\hat{m}$ is the number of test patterns after compaction, The first constraint $\text{CNF}_G(\overrightarrow{s_i})$ is the characteristic function in CNF representation for the given gate network, which constrains the logic value $v_g(s_i)$ for each gate $g \in G$ and input vector s_i. The second constraint guarantees that each fault is covered by at least one

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.

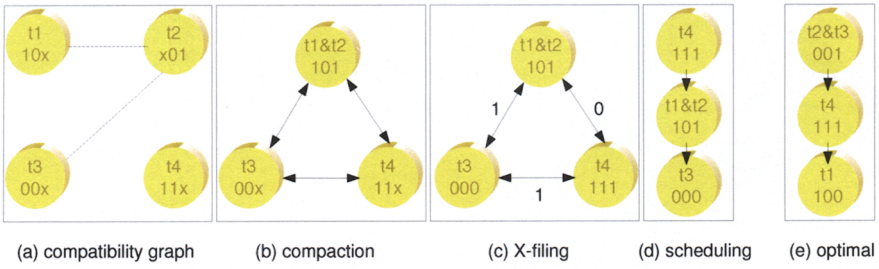

Figure 4: The computational procedure of power-aware ATPG.

generated test vectors after ordering and X-filling.

Notice that the objective in (6) is nonlinear due to the presence of the unknown $\hat{m}$ in the denominator. However, since the number of final test vectors should be limited for the purpose of testing time control, the feasible assignments for $\hat{m}$ must be in within the following range.

$$\hat{m} \in [m^*, \lceil \beta \cdot m^* \rceil]$$

where m^* is the minimal number of test vectors obtained by an optimal compaction algorithm, and $\beta \geq 1$ is a constant which is set as 2 in our experiments. Therefore we can linearize Formulation (6) by solving a sequence of PB optimization problems. In each of these problems the value of $\hat{m}$ is fixed as a constant $\hat{m}_p = m^*, \cdots, \lceil \beta \cdot m^* \rceil$, and there are $\hat{m}_p \cdot K$ variables s_i^j for $i = 1, \ldots, \hat{m}_p$ and $j = 1, \ldots, K$. We give the details of the optimization problem below.

The optimization problem (6) can be solved by solving a sequence of Pseudo-Boolean constraint problems

$$\begin{aligned}
&\text{minimize} && P(\overrightarrow{s}, \hat{m}_p) && (7)\\
&\text{subject to} && \text{CNF}_G(\overrightarrow{s_i}) && \text{for } i \in [1,m], \quad (8)\\
& && \bigvee_{i=1}^{\hat{m}_p} (\overrightarrow{s_i} \dashv \overrightarrow{t_j}), && \text{for } j \in [1,m] \quad (9)\\
& && s_i^j \in \{0,1\}, && \text{for } i \in [1,m], j \in [1,K] \quad (10)
\end{aligned}$$

The number of constraints in the PB optimization problem is dominated by (9), which has $O(K \cdot m^2)$ clauses, and the number of variables is $O(K \cdot m)$.

3.3 Example

Consider the example shown in Figure 3 to illustrate the proposed PB-based formulation. Since the minimal test vector number is 3 after compaction, we fix $\hat{m}_p = 3$. To compact, order and X-fill $\overrightarrow{t_1}, \cdots, \overrightarrow{t_4}$, the following 9 variables are needed.

$$\overrightarrow{s_1} = (s_1^1, s_1^2, s_1^3) \in \{0,1\}^3 \quad \overrightarrow{s_2} = (s_2^1, s_2^2, s_2^3) \in \{0,1\}^3$$
$$\overrightarrow{s_3} = (s_3^1, s_3^2, s_3^3) \in \{0,1\}^3$$

Based on (7), the objective function is

$$E_{reg} = \sum_{i=1}^{3} \sum_{j=1}^{2} |s_i^j - s_i^{j+1}|$$

$$E_{cap} = \sum_{i=1}^{2} \sum_{g \in G} C_g \cdot |v_g(\overrightarrow{s_i}) - v_g(\overrightarrow{s_{i+1}})|$$

where $G = \{z_1, z_2, z_3, f\}$. In this example, the power consumption in combinational portion during scan shifting is eliminated by replacing SFFs with SHFFs. Notice that $|x_1 - x_2| = x_1 \oplus x_2$ can be linearized by introducing new variable y, so that $y = x_1 \oplus x_2$, which can be rewritten in CNF as

$$(\neg x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee x_2 \vee y) \wedge \quad (11)$$
$$(\neg x_1 \vee x_2 \vee \neg y) \wedge (x_1 \vee \neg x_2 \vee \neg y).$$

The logic value constraints (8) are as follows.

$$\begin{aligned}
&(s_i^1 \vee \neg v_{z_1}^i) \wedge (s_i^2 \vee \neg v_{z_1}^i) \wedge (\neg s_i^1 \vee \neg s_i^2 \vee v_{z_1}^i) \wedge\\
&(s_i^3 \vee \neg v_{z_2}^i) \wedge (s_i^2 \vee \neg v_{z_2}^i) \wedge (\neg s_i^3 \vee \neg s_i^2 \vee v_{z_2}^i) \wedge\\
&(\neg v_{z_1}^i \vee v_{z_3}^i) \wedge (\neg v_{z_2}^i \vee v_{z_3}^i) \wedge (v_{z_1}^i \vee v_{z_2}^i \vee \neg v_{z_3}^i) \wedge\\
&(\neg v_{z_3}^i \vee \neg v_{z_4}^i) \wedge (v_{z_3}^i \vee v_{z_4}^i)
\end{aligned}$$

where v_g^i denotes $v_g(\overrightarrow{s_i})$, and the above four constraints represents the characteristic function to compute the logic value of `AND` gates z_1 and z_2, `OR` gate z_3 and `INV` gate f, respectively, under test vector $\overrightarrow{s_i}$, for $i = 1, \cdots, 3$.

Based on (9), the following constraints are needed to cover all faults,

$$\begin{aligned}
&((s_1^1 \wedge \neg s_1^2) \vee (s_2^1 \wedge \neg s_2^2) \vee (s_3^1 \wedge \neg s_3^2)) \wedge\\
&((\neg s_1^2 \wedge s_1^3) \vee (\neg s_2^2 \wedge s_2^3) \vee (\neg s_3^2 \wedge s_3^3)) \wedge\\
&((\neg s_1^1 \wedge \neg s_1^2) \vee (\neg s_2^1 \wedge \neg s_2^2) \vee (\neg s_3^1 \wedge \neg s_3^2)) \wedge\\
&((s_1^1 \wedge s_1^2) \vee (s_2^1 \wedge s_2^2) \vee (s_3^1 \wedge s_3^2))
\end{aligned}$$

After solving the above PB constraint problem, the optimal solution returned by PB solver is

$$\overrightarrow{s_1} = (0,0,1) \quad \overrightarrow{s_2} = (1,1,1) \quad \overrightarrow{s_3} = (1,0,0)$$

Particularly, three procedures, i.e., compaction, ordering and X-filing, are performed simultaneously and power is minimized by solving the PB constraint problem.

3.4 Speedup by Slicing Windows

In our implementation, minisat+ [22] is employed as the PB solver. Essentially the reasoning in a PB solving procedure includes solving multiple SAT instances iteratively. Due to the NP-completeness of SAT problem, the runtime of solving the proposed PB constraint problem could be prohibitively high (will be shown in Section 4).

To cope with the runtime issues of the proposed PB formulation, we present a *slicing window*-based heuristic to speedup the algorithm (see Alg. 1). Instead of solving all m test patterns at the same time, we each time consider α test patterns, which might be compacted, ordered, and X-filled, and relax these α test patterns according to the original test sequence. Here the relaxation procedure is done by finding and restoring the test vectors which can only be covered by these α test patterns by performing pattern compatibility or coverage checking. After the relaxation is done, we solve the PB problem for these relaxed test patterns to minimize power (see steps 4-8 in Alg. 1). We perform such window-based

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.

PB optimizations in iterations, which all m test patterns are passed by the slicing window in one iteration, until no improvement can be found between two successive iterations or the number of iterations exceeds the specified limit.

Algorithm 1 α-Slicing Windows

Require: $\overrightarrow{t_1}, \cdots, \overrightarrow{t_m}$, α
1: Initialize the ordering of $\overrightarrow{t_1}, \cdots, \overrightarrow{t_m}$
2: Initialize test sequence $T = \overrightarrow{t_1}, \cdots, \overrightarrow{t_m}$
3: **while** power consumption is less than that of the previous iteration and the number of iterations is below the specified limit **do**
4: **for** $i = 1, \cdots, m - \alpha$ **do**
5: Remove test vectors $\overrightarrow{t_i}, \cdots, \overrightarrow{t_{i+\alpha-1}}$ from T
6: Form a window of test vectors W = a set of test vectors in $\overrightarrow{t_1}, \cdots, \overrightarrow{t_m}$ for faults cannot be covered by T
7: Formulate and solve PB based on window W
8: Insert test vectors returned from PB solver into position i of T

3.5 Extension to Peak Power Minimization

For peak power minimization, we change the objective function in the optimization problem to (5). This objective is linearized by adding a new variable t and solving the optimization problem:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & \sum_{j=1}^{K-1} w_{ij} \cdot |s_i^j - s_i^{j+1}| \le t \\ & C_g \cdot |v_g(\overrightarrow{s_i}) - v_g(\overrightarrow{s_{i+1}})| \le t \\ & \text{all other constraints from (6)} \end{array} \quad (12)$$

Note that Formulation (12) is not a PB constraint problem due to the real-value variable t. However, we can still use PB solver to solve it by performing a binary search of t starting from a upper bound which can be obtained by performing the conventional sequential peak power optimization techniques. In each binary search iteration, t is set as a constant t_p and Formulation (12) becomes a PB constraint problem without an objective function. If the problem is SAT, a smaller t_p is tried. Otherwise, a larger t_p is used. In addition, Formulation (12) can be solved by a generic ILP solver, which is less efficient than SAT-based PB solver in our experiments due to the particular problem structure. The rest of the details, including the slicing window-based optimization, are identical.

4. EXPERIMENTAL RESULTS

The proposed PB-based algorithm is implemented in C++ and tested on ISCAS'89 benchmark circuits. Minisat+ [22] is used to solve the PB problem instances. For each benchmark circuit, we first use ATALANTA [23] to generate one test pattern for every fault by "-Z" option, which disables the test pattern compaction for the purpose of comparison. The COX takes the set of test patterns generated by ATALANTA as the input and perform simultaneous compaction, ordering and X-filing to minimize average power.

Figure 5 shows the number of remained test vectors in the course of slicing window iterations for benchmark $s298$ with window size equals to 4. The number of remained test vectors converges when 150 slicing windows are passed, which is done within 4 iterations in slicing window-based COX algorithm. Similar observations are obtained for other benchmark circuits, and we found that the results converge within 5 iterations for all benchmark circuits in our experiments. In the following experiments, we will show the results for both $\alpha = 2$ and $\alpha = 4$ for the comparison purpose, and we found that window size $\alpha = 2$ achieves the best tradeoff between the solution quality and the runtime.

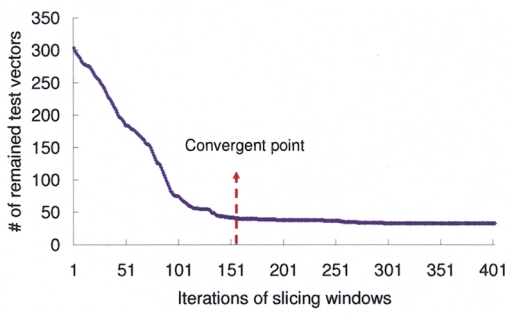

Figure 5: Convergency of the proposed slicing window-based heuristic (s298)

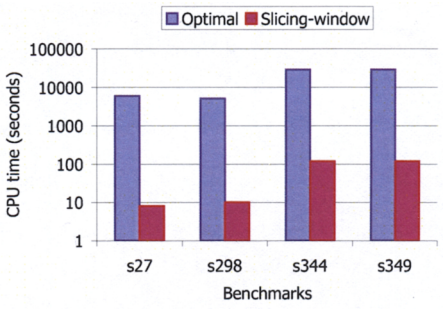

Figure 6: Runtime: optimal vs. slicing-window

We first compare the quality and runtime of optimal solution, i.e, PB-based algorithm without slicing window, and the slicing window-based heuristic with $\alpha = 2$. Due to the capability of the PB solver, four small circuits (s27, s298, s344, s349) are tested using 20 test patterns. The PB solver is set to timeout if no improvement is obtained in 1800 seconds. Three of these circuits are terminated by timeout and s298 finishes normally. For all four circuits, the optimal solution and the slicing window-based heuristic return the same results, while the former is three orders of magnitude slower than the latter as shown in Figure 6.

We then compare the proposed COX with the traditional sequential compaction, ordering and X-filing flow. Table 2 summaries the experimental results, including number of test vectors (column "test#"), power dissipation (column "power") and runtime (column "runtime"). Two window sizes, $\alpha = 2$ and $\alpha = 4$, are tested in the proposed COX algorithm (column "simu"). "-" indicates that the algorithm cannot finish within 30 hours. In the sequential flow (shown in columns "seq"), compaction is performed using ATALANTA [23] "-c" option. ATALANTA compacts test patterns using two different methods: reverse order compaction and shuffling compaction. First, test patterns are applied in the reverse order and fault simulated (reverse order compaction). Second, test patterns are shuffled randomly and fault simulated (shuffling compaction). During the fault simulations, all the test patterns which do not detect a new fault are eliminated [23]. Columns "orig" and "seq" under "test#" show the number of test vectors before and after ATALANTA compaction. Table 2 shows that the compaction in ATALANTA does not always minimize test set, and therefore our COX searches $\hat{m}_p \in [0.9m^*, 2m^*]$, where m^* is the number of test vectors compacted by ATALANTA. In addition, the ordering and X-filing are performed based on the TSP-based algorithm proposed in [15]. The power consumption in Table 2 is calculated based on (4) by assuming the uniform weight in (2) and uniform capacitance

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.

circuit characteristics				test#				average power			runtime(seconds)	
circuit	PI#	PO#	reg#	orig	simu(α=2)	simu(α=4)	seq	simu(α=2)	simu(α=4)	seq	α=2	α=4
s208	10	2	8	215	56	51	29	9.04	9.33	16.45	72	341
s208.1	11	1	8	217	60	63	35	9.02	9.14	14.86	72	341
s298	3	6	14	308	36	34	35	9.86	9.38	15.43	49	169
s344	9	11	15	342	32	27	22	11.56	12.22	20.82	54	1746
s349	9	11	15	348	27	28	22	10.96	11.61	20.45	51	2458
s382	3	6	21	399	51	48	32	12.67	11.22	21.75	84	779
s386	7	7	6	384	88	88	75	8.36	8.84	10.85	94	150
s400	3	7	21	418	53	47	33	11.94	10.96	22.18	95	828
s420	19	2	16	430	87	80	53	11.34	11.39	29.45	176	5364
s444	3	6	21	460	52	52	33	10.94	10.94	22.67	131	449
s510	19	7	6	564	88	91	60	12.47	11.57	22.52	64	1897
s526	3	6	21	554	82	79	69	12.40	12.41	21.55	106	1653
s526n	3	6	21	553	87	80	69	10.97	12.86	21.55	98	1242
s641	35	24	19	467	57	51	64	22.42	18.55	50.11	265	99008
s713	35	23	19	543	58	-	52	21.05	-	49.83	254	-
s820	18	19	5	850	135	-	111	12.39	-	16.77	218	-
s832	18	19	5	856	139	-	111	11.66	-	16.6	170	-
GeoMean				434	63	54	47	11.87	11.26	21.38	104	1156
Ratio					1.34	1.15	1	0.56	0.53	1	1	11×

Table 2: Comparison between COX ("simu") and the sequential flow ("seq")

for all gates in (3). Table 2 shows that the proposed COX algorithm reduces test mode power by 44% (47%) in the cost of 34% (15%) more test vectors for $\alpha = 2$ ($\alpha = 4$). Note that the runtime for the sequential algorithm is < 1 second. In addition, compared to $\alpha = 4$, $\alpha = 2$ achieves similar power dissipation with over $11\times$ speedup.

5. CONCLUSIONS AND FUTURE WORK

For the first time, an optimal formulation for simultaneous test pattern compaction, ordering and X-filling (COX) for test mode power minimization has been proposed. The formulation is based on a sequence of Pseudo-Boolean optimization problems. In addition, a window-based local search is implemented to cope with the runtime complexity due to the PB-based formulation. Experimental results show that the proposed COX algorithm reduces average power by 47% compared to the conventional sequential test pattern compaction, ordering and X-filling flow. Gearing the window-based local search, our COX algorithm runs three orders of magnitude faster.

Due to its high quality of power optimization, the proposed COX algorithm is particularly suitable to be applied after the conventional ATPG flow as a test pattern refinement for power optimization. Moreover, the proposed methodology can be applied for various test power optimization problems other than average and peak power minimization. For example, the PB-based formulation can be easily modified to handle average power minimization problem under the peak power constraint.

6. REFERENCES

[1] P. Girard, "Survey of low-power testing of vlsi circuits," *IEEE Dessign and Test of Computers*, vol. 19, pp. 80–90, May-June 2002.

[2] Y. Zorian, "A distributed bist control scheme for complex vlsi devices," in *VLSI Test Symposium*, pp. 4–9, April 1993.

[3] A. Uzzaman, P. Gallagher, and E. Malloy, "The need to address power during manufacturing test," *EDA DesignLine*, Oct 2008.

[4] S. Remersaro, X. Lin, Z. Zhang, S. Reddy, I. Pomeranz, and J. Rajski, "Preferred fill: A scalable method to reduce capture power for scan based designs," in *IEEE international Test Conference*, pp. 1–10, October 2006.

[5] J. Li, Q. Xu, Y. Hu, and X. Li, "On reducing both shift and capture power for scan-based testing," in *Asia and South Pacific Design Automation Conference*, pp. 653–658, March 2008.

[6] J. Li, Q. Xu, Y. Hu, and X. Li, "ifill: An impact-oriented x-filling method for shift- and capture-power reduction in at-speed scan-based testing," in *Design, Automation and Test in Europe*, pp. 1184–1189, March 2008.

[7] S. Ravi, "Power-aware test: Challenges and solutions," in *IEEE international Test Conference*, pp. 1–10, October 2007.

[8] L. H.-T. and J.-M. Li, "Simultaneous capture and shift power reduction test pattern generator for scan testing," *Computers and Digital Techniques, IET*, vol. 2, March 2008.

[9] R. Sankaralingam, R. Oruganti, and N. Touba, "Static compaction techniques to control scan vector power dissipation," in *VLSI Test Symposium*, pp. 35–40, April-May 2000.

[10] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "Reducing power consumption during test application by test vector ordering," *IEEE International Symposium on Circuit and Systems*, vol. 2, pp. 296–299, May-June 1998.

[11] V. Dabholdar, S. Chakravarty, L. Pomeranz, and S. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 1325–1333, December 1998.

[12] X. Wen, Y. Yamashita, S. Kajihara, L. Wang, K. Saluja, and K. Kinoshita, "On low-capture-power test generation for scan testing," in *Proc. of IEEE 23th VLSI Test Symposium*, pp. 265–270, November 2005.

[13] X. Wen, S. Kajihara, K. Miyase, T. Suzuki, K. Saluja, L. Wang, K. Abdel-Hafez, and K. Kinoshita, "A new atpg method for efficient capture power reduction during scan testing," in *Proc. of IEEE 24th VLSI Test Symposium*, May 2006.

[14] S. Kajihara, K. Ishida, and K. Miyase, "Test vector modification for power reduction during scan testing," in *Proc. of 20th IEEE VLSI Test Symposium*, pp. 160–165, April-May 2002.

[15] J. C. Costa, P. F. Flores, H. C. Neto, J. C. Monteiro, and J. P. M. Silva, "Power reduction in bist by exploiting don't cares in test patterns," *IEEE IWLS*, 1998.

[16] H. Mangassarian, A. Veneris, S. Safarpour, F. N. Najm, and M. S. Abadir, "Maximum circuit activity estimation using pseudo-boolean satisfiability," in *Proc. Design Automation and Test Conf. in Europe*, pp. 1538–1543, 2007.

[17] M. Bushnell and V. Agarwal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, 2000.

[18] Y. Wu and M.-T. Chao, "Scan-chain reordering for minimizing scan-shift power based on non-specified test cubes," in *IEEE VLSI Test Symposium*, pp. 147–154, April-May 2008.

[19] S. Remersaro, X. Lin, S. Reddy, I. Pomeranz, and J. Rajski, "Low shift and capture power scan tests," in *IEEE international VLSI Design Conference*, pp. 793–798, January 2007.

[20] N. Een and A. Biere, "Effective preprocessing in sat through variable and clause elimination," in *SAT*, 2005.

[21] http://www.mosek.com.

[22] http://minisat.se/MiniSat+.html.

[23] http://www.vtvt.ece.vt.edu/vlsidesign/downloadTools.php.

Authorized licensed use limited to: Univ of Calif Los Angeles. Downloaded on May 28, 2009 at 15:38 from IEEE Xplore. Restrictions apply.