In Proceedings of the IEEE/ACM International Conference on VLSI Design (VLSI), pages 37-41, January 1999

Assignment and Reordering of Incompletely Specified Pattern Sequences Targetting Minimum Power Dissipation

Paulo Flores, José Costa, Horácio Neto, José Monteiro and João Marques-Silva
Technical University of Lisbon/IST
Cadence European Laboratories/INESC
Rua Alves Redol, 9
1000 Lisboa, Portugal
{pff,jccc,hcn,jcm,jpms}@inesc.pt

Abstract

For a significant number of electronic systems used in safety-critical applications circuit testing is performed periodically. For these systems, power dissipation due to Built-In Self Test (BIST) can represent a significant percentage of the overall power dissipation. One approach to minimize power consumption in these systems consists of test pattern sequence reordering. Moreover, a key observation is that test patterns are in general expected to exhibit don't cares, which can naturally be exploited during test pattern sequence reordering. In this paper we develop an optimization model and describe an efficient algorithm for reordering pattern sequences in the presence of don't cares. Preliminary experimental results amply confirm that the resulting power savings due to pattern sequence reordering using don't cares can be significant.

1. Introduction

Many circuits today include on-chip structures that enable circuit self-testing, known as built-in self-test (BIST) [1]. Initially designed to make the testing of the circuits out of the fabrication line easier, they allow for the periodic testing of the circuit. This can be especially important for circuits used in safety-critical or mobile devices. Clearly the penalty to pay is the extra circuitry required for BIST. One approach to reduce this overhead is to use a simple linear feedback shift register (LFSR) to generate a pseudo-random input sequence, which is run until a given fault coverage has been achieved [1, 4]. The disadvantage is that for high fault coverages the run time may become too long. A different approach is to use an automated test-pattern generator (ATPG) tool to obtain a (ideally minimum) set of test patterns necessary for the desired fault coverage. Then, the BIST structure reduces to a counter-type finite state machine (FSM) that generates each of these patterns sequentially [4]. Even though this latter solution in general requires larger area, it is also clear that it provides shorter test sequences, thus being the option of choice for specific applications [1, 4] where power and not area is the most important design goal. Moreover, the increased use of periodic testing in safety-critical devices raises concerns about the power that is consumed during this process. Consequently, techniques for reducing the power dissipation during testing are particularly relevant for these devices.

In this paper we address the problem of power reduction during testing. Even though solutions for solving this problem consist of reordering sequences of completely specified test patterns [5], one might expect the potential existence of don't cares in test patterns to help further reduction in power. The main purpose of this paper is to propose solutions for this problem and provide comprehensive empirical evidence that the existence of don't cares in test patterns can in fact play a significant role in reducing power dissipation during testing.

2. Power Reduction with Completely Specified Test Patterns

2.1. Power Dissipation Model

It has been shown [6] that during normal operation of well designed CMOS circuits the switching activity power accounts for over 90% of the total power dissipation. Thus power optimization techniques at different levels of abstraction target minimal switching activity power. The model for power dissipation for a gate *i* in a logic circuit is simplified to:

$$P = \frac{1}{2} \cdot C_i \cdot V_{DD}^2 \cdot f \cdot N_i \quad (1)$$

where V_{DD} is the supply voltage, f is the frequency of operation, C_i is the node capacitance and N_i is the node switching activity.

Both simulation-based (e.g., [3]) and probabilistic (e.g., [7]) techniques have been proposed for the computation of N_i. Simulation-based techniques use a logic or timing simulator. The circuit is simulated with a *sufficiently large* number of randomly generated input vectors to obtain an average transition count at every gate in the circuit. Simulation-based techniques can be very efficient for loose accuracy bounds. Increasing the accuracy may require a prohibitively high number of simulation vectors.

Given some statistical information of the inputs, probabilistic methods propagate this information through the logic circuit obtaining statistics about the switching activity at each node in the circuit. Only one pass through the circuit is needed thus making these methods potentially very efficient. Still, modeling issues like correlation between signals can make these methods computationally expensive.

2.2. Model for Completely Specified Test Patterns

For the testing of the circuit, the only requirement is that all the test-patterns generated by the ATPG are applied to the circuit. Thus, one degree of freedom that can be explored is the *order* by which these patterns are applied.

Let $\{T_1, T_2, \ldots, T_m\}$ be a given sequence of *completely specified* test patterns. The problem of power reduction during testing can be formulated as the identification of a permutation $\langle i_1, \ldots, i_m \rangle$ such that the overall power consumption is minimized. This problem can be naturally reduced to the (euclidean) traveling salesperson problem (TSP) [9].

Moreover, the power consumption between every possible input-vector pair (T_k, T_l) can be heuristically approximated by the Hamming distance between the input vectors. The argument is that by minimizing the switching activity at the inputs we will also be minimizing the switching activity on internal nodes in the circuit. Although this is not always true (one transition in a given input may cause many transitions in internal nodes, whereas several inputs changing may cause fewer transitions), it is a good approximation for typical circuits, as confirmed by the results presented in Section 6.

Finally, we note that even though the euclidean traveling salesman problems is NP-hard, several efficient polynomial-time approximation algorithms exist [9].

3. Reordering Test Patterns with Don't Cares

In general, ATPG algorithms attempt to generate test patterns with a maximal number of don't cares, so that compaction of test patterns becomes facilitated. Hence, power reduction techniques for circuit testing should address the potential advantages of exploiting the don't cares in the test set. We have resorted to two different ATPG algorithms, ATALANTA [12] and MTP [8]. ATALANTA heuristically generates test patterns with don't cares, whereas MTP implements a formal model for the computation of test patterns with the maximum number of don't cares.

The existence of test pattern with don't cares implies that the Hamming distances between test patterns become conditional, and depend on the final assignments of the unspecified bits. Let us consider the following test set $\{T_1 = \langle 1, X, 0, 0\rangle; T_2 = \langle X, 1, 0, 1\rangle; T_3 = \langle 0, X, 1, 1\rangle\}$. Depending on the values specified to the don't care bits, the Hamming distance from T_1 to T_2 can range from 1 to 3.

In the next section we propose a formal optimization model for reducing the sum of the Hamming distances in pattern sequences, hence with clear potential application to minimizing power during BIST. However, the proposed optimization model denotes a complex optimization problem, and consequently we then propose heuristic algorithms for approximating the solution to this problem.

4. A Formal Model for Pattern Sequence Reordering Using Don't Cares

In this section we derive a formal integer linear optimization model for pattern reordering under the assumption that patterns exhibit don't cares, which is also valid for completely specified patterns. The cost function assumed is given by the sum of the Hamming distances between each pair of patterns, which will be henceforth referred to as the *Hamming cost* for the pattern sequence. As empirically confirmed in Section 6, we assume that a reduction in the sum of the Hamming distances accurately measures the reduction in power associated with the sequence of patterns.

For the case where the patterns are fully specified we can always map an instance of the TSP into an instance of integer linear programming (ILP), in particular into one where the variables assume binary values (i.e., 0-1 ILP) [14]. Consequently, our goal is to modify this model in order to also capture don't care conditions. The resulting model basically allows for conditional weights (i.e., conditional Hamming distances) between each pair of vertices (i.e., patterns). We start by reviewing a 0-1 ILP model for the TSP, following [14], and then proceed to develop an optimization model in the presence of don't cares.

4.1. A 0-1 ILP Model for the TSP

Let $T_i = \langle b_{i1}, \ldots, b_{in} \rangle$ and $T_j = \langle b_{j1}, \ldots, b_{jn} \rangle$ denote two patterns, and let c_{ij} denote the Hamming distance between T_i and T_j. Further, let x_{ij} denote a Boolean variable such that $x_{ij} = 1$ provided T_j immediately follows T_i in the sequence of patterns, and $x_{ij} = 0$ otherwise. Finally, let $V = \{1, \ldots, m\}$ denote a set of vertices where each i is associated with pattern T_i. Consequently, from [14], the resulting instance of TSP can be polynomially formulated as follows,

$$\text{minimize} \qquad \sum_{i,j} c_{ij} \cdot x_{ij} \tag{2}$$

subject to the following constraints,

$$\begin{aligned}
&\sum_{\{i\,:\,(i,j)\,\in\, V\times V\}} x_{ij} = 1 \quad j \in V \\
&\sum_{\{j\,:\,(i,j)\,\in\, V\times V\}} x_{ij} = 1 \quad i \in V \\
&u_i - u_j + m \cdot x_{ij} \le m - 1 \quad (i,j) \in V \times V,\ i \neq 1,\ j \neq 1 \\
&x_{ij} \in \{0,1\},\ u_i \in R \qquad i, j \in V
\end{aligned} \tag{3}$$

where the constraints $u_i - u_j + m \cdot x_{ij} \le m - 1$ guarantee that no subtours will be selected, and where each variable u_i is a real number. Hence, only complete tours satisfy the constraints. (An elegant proof of this fact can be found in [14]).

4.2. An ILP Model for Pattern Reordering Using Don't Cares

Assuming that each pattern can exhibit don't care bits

then, as mentioned in Section 3, the distance c_{ij} between test patterns is a conditional number, whose final value is imposed by how the don't care bits are actually assigned. Hence, we modify the cost function to consider the conditional costs,

$$\text{minimize} \qquad \sum_{i,j} s_{ij} \tag{4}$$

Next we show how each conditional cost s_{ij} is defined. We start by introducing a Boolean variable,

$$d_{ijk} = (b_{ik} \oplus b_{jk}) \wedge x_{ij} \qquad i,j \in V, k \in \{1, \ldots, n\} \tag{5}$$

where k denotes each of the possible bit positions. We note that d_{ijk} assumes value 1 if and only if T_j follows T_i in the pattern sequence and bit k in T_i differs from bit k in T_j. In addition, this condition can be represented in CNF format [15] as follows,

$$\begin{aligned} \varphi_{ijk} = {} & (\neg x_{ij} \vee \neg b_{ik} \vee b_{jk} \vee d_{ijk}) \wedge \\ & (\neg x_{ij} \vee b_{ik} \vee \neg b_{jk} \vee d_{ijk}) \wedge (\neg d_{ijk} \vee x_{ij}) \\ & \wedge (\neg d_{ijk} \vee b_{ik} \vee b_{jk}) \wedge (\neg d_{ijk} \vee \neg b_{ik} \vee \neg b_{jk}) \end{aligned} \tag{6}$$

Moreover, and using the straightforward mapping of [8], these clauses can be written as linear inequalities,

$$\begin{aligned} \varphi_{ijk} = {} & (-x_{ij} - b_{ik} + b_{jk} + d_{ijk} \geq -1) \\ & \wedge (-x_{ij} + b_{ik} - b_{jk} + d_{ijk} \geq -1) \wedge (-d_{ijk} + x_{ij} \geq 0) \\ & \wedge (-d_{ijk} + b_{ik} + b_{jk} \geq 0) \wedge (-d_{ijk} - b_{ik} - b_{jk} \geq -2) \end{aligned} \tag{7}$$

For each test pattern T_i, if bit b_{ik} is a don't care, then $b_{ik} \in \{0, 1\}$ is one of the problem variables. Otherwise, the value of b_{ik} is specified by pattern T_i. We can now redefine the integer-valued cost s_{ij} between T_i and T_j as follows,

$$s_{ij} = \sum_{k=1}^{n} d_{ijk} \tag{8}$$

where clearly $s_{ij} \in \{0, \ldots, n\}$. This definition of s_{ij} as well as the above constraints complete the formulation of the model. Using (3), (7) and (8) the resulting ILP model becomes,

$$\text{minimize} \qquad \sum_{i,j} \sum_{k} d_{ijk}$$

subject to,

$$\begin{aligned} & \sum_{\{i\,:\,(i,j) \in V \times V\}} x_{ij} = 1 && j \in V \\ & \sum_{\{j\,:\,(i,j) \in V \times V\}} x_{ij} = 1 && i \in V \\ & u_i - u_j + m \cdot x_{ij} \leq m - 1 && (i,j) \in V \times V, i \neq 1, j \neq 1 \\ & x_{ij} \in \{0,1\}, u_i \in R && i,j \in V \\ & \varphi_{ijk} && i,j \in V, k \in \{1, \ldots, n\} \\ & b_{ik}, d_{ijk} \in \{0,1\} && i,j \in V, k \in \{1, \ldots, n\}, b_{ik} \notin T_i \end{aligned} \tag{9}$$

where the variables s_{ij} were replaced by their definition in (8), and where $b_{ik} \notin T_i$ denotes that b_{ik} is an unassigned bit for pattern T_i.

It can be showed that the solution to ILP (9) denotes an assignment to the don't care bits and a reordering of the pattern sequence which minimizes the overall sum of Hamming distances in the selected ordering of patterns.

The proposed model denotes a complex optimization problem, clearly no easier than TSP. Consequently, and in order to evaluate the practical usefulness of pattern sequence reordering in the presence of don't cares, we propose in the next section different heuristic algorithms for computing approximated solutions to the pattern reordering problem.

5. Power Reduction Algorithms

As described in Section 2.2, for completely specified test patterns, the straightforward representation of the power reduction problem as an instance of the TSP problem immediately yields a wealth of approximation algorithms [9]. Our approach is to modify an existing approximation algorithm for the TSP instead of solving the model proposed in the previous section with an ILP solver. We choose to adapt the 2-opt [9] local search approximation algorithm for the TSP, that is described below. The resulting algorithm is organized as follows:

1. Use a dedicated algorithm for computing a test set where each test pattern contains don't cares. Either MTP [8] or ATALANTA [12] can be used.
2. Apply a heuristic procedure for identifying an initial tour. Several different heuristics are described below.
3. Use a modified 2-opt local-search approximation algorithm for the TSP to reorder the test patterns. Repeat this step while the tour cost can be reduced.

The following initial ordering heuristics have been implemented (which will henceforth be referred to as **H1** through **H5)**:

1. Randomly order the test patterns.
2. Order test patterns by decreasing order of don't cares in each test pattern. By choosing for the first test patterns those with more don't cares one can expect that the distances between the first test patterns be the lowest possible.
3. This heuristic starts by applying heuristic 2. Afterwards, greedily select the next test pattern as one that minimizes the distance from the current test pattern. This heuristic goes one step further in minimizing the distances between the first test patterns by choosing the second best test pattern, and then the third best, and so on.
4. In this heuristic for each bit position the don't care bits are set to the bit that occurs more often. By using this approach the test patterns are expected to become more similar between each other. Next an ordering is made that approximates Gray coding. This approach attempts to order the test patterns in such a way that the average distances between test patterns is minimized.
5. The last heuristic sets the don't cares in the same manner in the heuristic 4. Afterwards, with all the test patterns specified, the Christofides TSP approximation algorithm is used for defining the initial tour. This

heuristic permits using a TSP approximation algorithm in a tour where the test patterns are expected to be similar to each other.

After having the initial tour of the test patterns the following modified 2-opt [9] is applied:

1. Evaluate the tour cost by specifying the don't care bits which minimize the distance between consecutive test patterns.
2. Reverse the action taken in Step 1 by getting the test patterns with don't cares.
3. For every pair of test patterns (T_i and T_j), cut the link between those test patterns and the next ones (T_{i+1} and T_{j+1}), and link T_i with T_{j+1} and T_j with T_{i+1}. For this new ordering obtain the tour cost as in Step 1.
4. If the *lowest* tour cost found in Step 3 is lower than the initial tour cost then keep the order for that lowest tour cost and repeat Step 1 for that ordering. Otherwise the algorithm terminates.

Finally, the test sequence considered by the power estimator is given by the result of the modified 2-opt with the don't care bits specified in such a way that the Hamming distance between consecutive test patterns is minimized.

6. Experimental Results

This section includes results of applying the algorithm described in the previous section to the MCNC benchmark circuits [10] and to the ISCAS benchmark circuits [2]. The ATPG tool ATALANTA [12] was used on all the experiments. ATALANTA was used to generate both completely and incompletely specified test patterns.

The experimental procedure consisted of first generating a set of completely-specified test patterns and subsequently reordering them such that the Hamming cost was minimized. For this minimization procedure, the Christofides approximation algorithm [9] was used. All the results in this paper are compared to the Hamming cost and power consumption under this input pattern sequence.

Afterwards, the set of incompletely specified test patterns was generated. The algorithm described in the previous section was used to generate the best ordering and don't-care assignment for the different initial ordering heuristics proposed.

6.1. MCNC Benchmarks

We first present in Table 1 results for the MCNC benchmark circuits on the reduction of the Hamming cost by exploiting the don't cares in the incompletely specified set of test patterns, which is the figure of merit that we are targeting directly. For each of the different heuristics, the percentage savings relative to the optimal sequence of completely specified pattern sequence is shown. It can be observed that for many of the examples significant reductions in the Hamming cost is obtained. Still in Table 1 we give the CPU time in seconds used by the ordering and assignment algorithm under the different heuristics on a Sun Ultra I with 384MB of main memory.

The results for power reduction in test sequences for the same circuits are shown in Table 2. The columns labeled ***completely specified*** indicate the percentage power savings that result from ordering a sequence of completely specified test patterns (**#PR**), and the number of computed test patterns (**#TP**). The columns labeled ***incompletely specified*** indicate the power savings from exploiting the don't cares in incompletely specified test patterns over an *already ordered* sequence of completely specified test patterns. In all cases a power estimator tool integrated in SIS was used for estimating the actual power dissipation from applying the test sequences [7].

Circuit	H1		H2		H3		H4		H5	
	%	CPU	%	CPU	%	CPU	%	CPU	%	CPU
9symml	7.7	17.3	10.5	17.9	16.1	2.0	14.7	18.4	14.7	4.1
alu4	24.3	153.1	22.7	156.9	30.5	16.2	22.7	129.0	29.0	47.6
cht	59.0	0.0	60.3	0.0	59.7	0.0	59.0	0.0	59.0	0.0
cm138a	18.0	0.0	16.0	0.0	16.0	0.0	16.0	0.0	14.0	0.0
cm150a	61.4	1.4	65.5	1.5	65.5	0.2	64.7	1.0	71.3	1.0
cm163a	40.5	0.0	48.6	0.0	43.2	0.0	45.9	0.0	48.6	0.0
cmb	32.6	0.1	32.6	0.0	32.6	0.0	32.6	0.0	32.6	0.0
comp	67.2	12.3	67.9	13.0	70.7	4.8	67.2	12.5	65.8	5.0
comp16	50.8	131.0	59.3	98.7	61.7	11.8	50.3	98.1	60.7	22.1
cordic	56.2	3.9	61.1	3.1	61.1	0.7	59.2	3.5	61.1	0.7
cu	48.3	0.2	48.3	0.1	52.6	0.1	50.5	0.1	50.5	0.0
majority	14.0	0.0	16.0	0.0	14.0	0.0	16.0	0.0	14.0	0.0
misex1	43.4	0.0	39.1	0.0	47.8	0.0	47.8	0.0	43.4	0.0
misex2	72.9	1.6	96.0	1.46	96.0	0.4	76.5	1.9	74.4	0.7
misex3	27.0	815.1	24.1	744.5	35.8	67.6	25.8	667.7	28.7	309.2
mux	67.9	1.2	66.2	1.3	69.5	0.5	68.7	1.2	67.9	0.6
pcle	47.6	0.0	49.5	0.0	47.6	0.0	49.5	0.0	49.5	0.0
pcler8	59.7	0.2	58.7	0.2	59.7	0.1	60.8	0.2	61.9	0.1
term1	71.3	3.7	73.0	4.2	74.7	2.3	74.4	4.5	74.4	2.5
too_large	57.7	817.1	60.8	745.8	63.3	217.5	58.6	604.9	63.3	335.5
unreg	64.1	0.0	63.2	0.0	63.2	0.0	64.1	0.0	63.2	0.0

Table 1: Hamming cost reduction and CPU times for the MCNC benchmark circuits

As can be readily concluded, large power savings ranging from 30% to 60% are achieved in most cases. This is particularly significant since these results measure the percentage power savings over the already ordered sequence of test patterns. Finally, we note that the number of test patterns (**#TP**) does not change significantly (especially for usage in BIST) from completely specified to incompletely specified test patterns. In addition, the results from Table 1 and Table 2 indicate that the measured reduction in power dissipation correlates well in most circuits with the achieved reduction in Hamming cost.

6.2. ISCAS Benchmarks

The results in the previous section validate the proposed approach for reducing power dissipation for medium-size circuits. For the ISCAS benchmarks we noticed that the proposed (and non-optimized) 2-opt algorithm would take a couple of hours of CPU time for the examples with a larger number of test patterns, and would take more than 24 hours of CPU time for C7552. As a

Circuit	Completely specified		Incompletely specified versus ordered completely specified					
	# TP	% PR	# TP	% PR (power reduction)				
				H1	H2	H3	H4	H5
9symml	78	43.9	80	3.8	7.2	15.3	11.1	17.1
alu4	100	29.0	128	12.2	11.7	18.3	13.7	20.4
cht	17	5.6	10	25.3	27.9	24.7	17.9	23.0
cm138a	12	36.3	12	20.9	20.7	11.0	16.2	17.5
cm150a	34	16.5	39	38.6	46.2	53.6	42.4	45.3
cm163a	15	15.6	14	31.2	34.7	40.0	35.4	42.7
cmb	30	43.2	27	16.9	17.7	21.9	13.8	15.1
comp	56	27.1	60	57.0	56.9	60.6	58.7	57.7
comp16	72	38.9	99	40.6	47.6	44.2	46.6	42.3
cordic	43	36.6	47	49.3	56.2	61.5	54.0	59.4
cu	27	35.6	26	23.4	34.3	36.4	13.1	30.3
majority	11	36.1	11	9.6	15.9	5.1	10.9	4.6
misex1	18	14.3	17	36.2	26.5	24.9	26.7	33.0
misex2	47	20.5	37	41.7	48.7	52.6	52.5	51.6
misex3	154	38.3	178	21.0	24.6	27.9	19.1	24.5
mux	35	20.5	38	28.7	31.2	36.9	41.4	32.0
pcle	17	16.6	20	28.1	37.1	47.2	31.2	42.7
pcler8	19	20.4	21	35.4	23.7	43.2	27.5	37.8
term1	43	14.4	43	43.6	49.5	47.2	39.6	46.9
too_large	103	32.1	146	36.1	41.2	49.9	42.1	46.3
unreg	15	15.8	10	12.4	12.5	12.0	11.2	17.2

Table 2: Power reduction results for the MCNC benchmarks

result, we modified the 2-opt algorithm so that only 500 links were examined at each iteration of the 2-opt algorithm (instead of a number that is quadratic in the number of test patterns). The results obtained with this modified 2-opt algorithm are shown in Table 3. As can be concluded, once again, large power savings can be obtained by generating test pattern with don't cares, reordering the test sets and specifying the unassigned bits so that the dissipated power is minimized.

Furthermore, we noticed that the percentage power savings in general increases as the size of the circuit and number of test patterns increases. Hence, for large circuits we expect the proposed power reduction algorithm to lead to similar or greater power savings. Regarding the heuristics proposed in Section 5 for constructing the initial tour, the results do not identify a clear best heuristic, even though the greedy heuristic **H3** performs better in most cases.

7. Conclusions

In this paper we propose an optimization model for reducing the Hamming cost in pattern sequences, and describe a heuristic algorithm for obtaining approximated solutions. This algorithm is then applied to reducing the power dissipation during testing by exploiting don't cares in test pattern sequences. We provide experimental evidence that exploiting don't cares in test sequences can lead to very significant savings in dissipated power. In designs where periodic testing is required, these power reduction techniques may play a key role in the design of BIST hardware.

Circuit	Completely specified		Incompletely specified versus ordered completely specified					
	# TP	% PR	# TP	% PR (power reduction)				
				H1	H2	H3	H4	H5
c432	58	10.4	75	48.4	45.9	49.0	45.1	51.3
c499	60	30.5	61	5.3	5.9	7.8	5.7	5.1
c880	51	27.4	79	33.5	38.6	49.6	35.5	48.5
c1355	94	14.2	96	58.1	64.5	68.6	63.2	66.9
c1908	128	18.7	175	20.3	21.7	42.4	30.8	43.4
c2670	117	16.9	156	37.2	40.7	44.9	35.2	45.6
c3540	159	28.9	253	13.5	8.1	18.8	14.2	20.1
c5315	116	11.0	158	47.5	45.5	51.2	44.8	51.5
c6288	25	18.6	54	54.0	54.9	55.6	56.3	50.9
c7552	217	15.1	347	39.7	49.6	64.1	52.7	67.8

Table 3: Power reduction results with modified 2-opt algorithm

References

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

[2] F. Brglez and H. Fujiwara, "A Neutral List of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN," in *Proceedings of the International Symposium on Circuits and Systems(ISCAS)*, 1985.

[3] R. Burch, F. Najm, P. Yang and T. Trick, "McPOWER: A Monte Carlo Approach to Power Estimation," in *Proceedings of the International Conference on Computer-Aided Design*, November 1992.

[4] K. Chakrabarty, B. T. Murray, J. Liu and M. Zhu, "Test Width Compression for Built-In Self Testing", in *Proceedings of the International Test Conference*, October 1997.

[5] S. Chakravarty and V. Dabholkar, "Minimizing Power Dissipation in Scan Circuits During Test Application," in *International Workshop on Low-Power Design*, April 1994.

[6] A. Chandrakasan, T. Sheng and R. Brodersen, "Low Power CMOS Digital Design," in *IEEE Journal of Solid State Circuits*, no. 4, vol. 27, pp. 473-484, April 1992.

[7] J. C. Costa, J. C. Monteiro and S. Devadas, "Switching Activity Estimation using Limited Depth Reconvergent Path Analysis", in *Proceedings of the International Symposium on Low Power Design*, 1997.

[8] P. Flores, H. Neto and J. Marques Silva, "An Exact Solution to the Minimum-Size Test Pattern Problem,", in *IEEE International Conference on Computer Design*, October 1998.

[9] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, 1996.

[10] IWLS'89 Benchmark Suite, available from http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth89/.

[11] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 1, pp. 4-15, January 1992.

[12] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report No. 12_93, Department of Electrical Engineering, Virginia Polytechnic Institute and State University, 1993.

[13] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.

[14] J. P. M. Silva and K. A. Sakallah, "Robust Search Algorithms for Test Pattern Generation," in *Proceedings of the Fault-Tolerant Computing Symposium*, June 1997.