An Efficient Test Vector Ordering Method for Low Power Testing†

X. Kavousianos[1], D. Bakalis[2,3], M. Bellos[2,3] and D. Nikolos[2,3]
[1]*Computer Science Dept., University of Ioannina, 45110 Ioannina, Greece*
[2]*Computer Engineering & Informatics Dept., University of Patras, 26500 Patras, Greece*
[3]*Research Academic Computer Technology Institute, 61 Riga Feraiou Str., 26221 Patras, Greece*
kabousia@cs.uoi.gr, bakalis@cti.gr, bellos@ceid.upatras.gr, nikolosd@cti.gr

Abstract

This paper presents a novel test vector ordering method for average power consumption minimization. The proposed method orders the test vectors taking into account the expected switching activity at the primary inputs and at a very small set of internal lines of the Circuit Under Test. The computational time required by the proposed method is very small while the power reduction achieved is very close to the best, with respect to power reduction, most time-consuming method. Experimental results show that apart from average power reduction, the proposed method achieves significant peak power reduction too.

1. Introduction

During the last decade, several power optimization techniques targeting minimal switching activity at circuit nodes have been proposed at all levels of the design hierarchy so as to reduce power dissipation. On the other hand, the test efficiency has been shown to have a high correlation with the toggle rate; hence, in test mode, in order to excite many potential faults by relatively short test sequences, the switching activity at circuit nodes is often several times higher than the switching activity during normal operation. The elevated average power dissipation during testing can be responsible for several kinds of problems: decreased overall yield, decreased reliability and system life cycle and increased product costs [1].

The focus of this paper is on the problem of minimizing power dissipation by ordering suitably the vectors of the test set. Several test vector ordering techniques have been already proposed in the open literature [2-5]. Although these techniques are normally used in external testing, they can also be used in deterministic BIST as well [1]. In this paper a new test vector ordering technique is proposed whose computational time is very small while power reduction approaches that of the best, most elaborate methods [3,5].

† This research was financially supported by the Public Benefit Foundation "Alexander S. Onassis" via its scholarships programs and by the State Scholarships Foundation of Greece via its postdoctoral research scholarships program.

2. Prior work and motivation

Test vector ordering techniques [2-5] are based on the construction of a complete weighted graph, called Transition Graph, where each node of the graph corresponds to a test vector. A weight $w_{i,j}$ is assigned to each edge $e_{i,j}$ connecting graph nodes v_i and v_j which is indicative of the energy consumed in the circuit due to the application of the test vector pair (v_i, v_j). Then the problem accounts for finding a Hamiltonian path in the transition graph such that the sum of weights is minimized. This is equivalent to the well-known Traveling Salesman Problem (TSP), which is NP-complete, therefore all proposed methods resort to heuristics. What differentiates the already known methods is what the weight represents.

If the weight is set equal to the energy consumed by the circuit under test (CUT) due to the application of a specific pair of test vectors or to the number of transitions activated in the CUT assuming a realistic delay model, then the graph is directed. On the contrary, it has been proven in [3] that under the general delay model or the zero delay model, the number of transitions in the CUT caused by test vector pair (v_i, v_j) is equal to those caused by vector pair (v_j, v_i). Therefore $w_{i,j} = w_{j,i}$ and the graph is undirected.

When weights are set equal to the number of transitions in the CUT and the general delay model is assumed [3], $t(t-1)/2$ logic/timing simulations are required for the construction of the transition graph, t being the number of test vectors. This makes the method inapplicable for large circuits with many test patterns. In order to reduce the time required to construct the transition graph, the adoption of the zero delay model was proposed in [5]. Unfortunately, although the simulations now run faster, their number remains $t(t-1)/2$, thus the problem of long computational times is only moderated. The time required for the construction of the transition graph can be reduced significantly if weight $w_{i,j}$ is set equal to the Hamming distance between test vectors v_i and v_j [2]. Since this technique does not take into account the structure of the CUT, the reduction in the total number of transitions is usually significantly lower than the reduction achieved in [3]. In fact, depending on the structure of the circuit, a transition at some primary input of a circuit may cause

IEEE COMPUTER SOCIETY

more transitions at internal lines than a transition at some other primary input. The authors of [4] took into account the structure of the CUT in the calculation of the weights in the transition graph and thus achieved larger reduction of the switching activity than that obtained in [2]. However, the switching activity reduction obtained in [4] is usually significantly lower than that achieved in [3].

3. Proposed method

The large power reduction achieved by the method proposed in [3] is mainly attributed to the exhaustive count of the transitions at all circuit lines for every test pair. However, the calculation of $w_{i,j}$ for every possible pair of test vectors requires $O(t^2)$ logic/timing simulations [3], where t is the cardinality of the test set. When the test set is large the required computational time is prohibitively long. Adopting the zero delay model, as in [5], the count of transitions caused at all CUT lines by a pair of test vectors can be done without simulating the CUT for every test vector pair. The CUT can be simulated for every test vector and the value of each node of the CUT can be stored. Then the number of the transitions $w_{i,j}$ caused by pair (v_i, v_j) can be calculated by the sum $\Sigma(v_i^f \oplus v_j^f)$ over all nodes f of the CUT, where v_i^f is the value (0 or 1) of the CUT line f when the test vector v_i is applied to the primary inputs of the CUT. This method although requires only $O(t)$ simulations is unrealistic for modern circuits due to its prohibitively large requirements in memory storage.

In the following we present a method according to which the CUT is simulated for every test vector but the values of only a very small number of selected internal lines of the CUT are stored. Weights derived for the primary inputs of the CUT and the selected internal lines are then used for constructing the transition graph and solving the test vector ordering problem. As we will show, the proposed method achieves a power reduction very close to that of [5] in very short execution time and with small memory requirements even for large circuits.

Our Test Vector Ordering algorithm begins with the calculation of initial weights of all circuit lines. Then, the selection of a subset of lines follows with the appropriate update of their weights. Finally, the transition graph is constructed and a test vector ordering heuristic is applied. The next subsections describe each step in more detail.

3.1 Weight calculation and internal line selection

The selection of the internal lines is based on a procedure, which gradually reduces the set of internal lines of the circuit until a predefined number of them is left. For that reason the weight of each selected line must be representative of the switching activity caused by the dropped lines in its fanout cone. When a line is dropped, its weight is shared among lines in its fan-in logic cone, which have not been dropped. The percentage of the weight of the dropped line l_i, received by line l_j, is proportional to the possibility a transition at line l_j to cause a transition at line l_i. Note that when a selected line has a large weight, then there is a strong possibility a transition at this line to cause a large number of transitions at lines in its fanout cone. Therefore, if we compute the weights only at the selected lines, then we can have an estimation of the number of transitions at all circuit lines. The weights of the selected internal lines along with the weights of the primary inputs can then be taken into account for test vector ordering.

The first step of the weight calculation and line selection procedure is to assign initial weights to all circuit lines. Consider the set IL of all internal lines of the circuit. Let $P(l)$ denote the signal probability at line l. Then, under the temporal independence assumption, the transition probability at line l is defined as $P_{tr}(l)=2P(l)(1-P(l))$ and can be regarded as an indication of the number of transitions that may occur at this line. The signal probability of a primary input of the CUT is calculated from the test set by dividing the number of test vectors that apply the logic value 1 at this primary input by the cardinality of the test set. The signal probability $P(l)$ of each internal line can be estimated by using the Full Range Cutting Algorithm [6]. The transition probability of a line is indicative of the number of transitions that appear at the line in a time interval. Thus we consider the transition probability of each line as its initial weight.

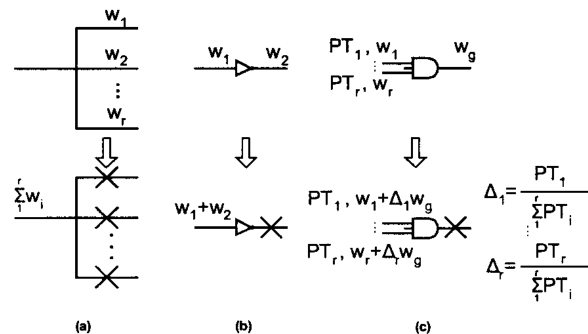

Figure 1. Internal line reduction procedures

The second step is to drop all the fanout branches, by summing their weights and assigning the result to the fanout stem, as shown in Figure 1.a. This is justified by the fact that every transition at a fanout stem implies a transition at all fanout branches. Similarly, the outputs of all gates with single input are dropped as shown in Figure 1.b, since a transition at the output implies a transition at the input. In the case that the input of the gate is a fanout branch, which has been already dropped, then the weight of the output is added directly to the fanout stem.

Let *LinesToSelect* be a user-defined parameter which specifies the number of lines of the circuit that will be finally selected so as their weights are taken into account in the test vector ordering procedure. The procedure iterates until *LinesToSelect* lines are left. At each step the output of the gate with the smaller weight is selected and marked for

IEEE COMPUTER SOCIETY

dropping. A small weight indicates a small number of transitions and thus the error caused by dropping the line is minimal. Moreover the estimation error caused by sharing a small weight to other lines is also low. The weight of the marked gate output is shared among the inputs of the gate proportionally to the transition probability (*PT*) of each input as shown in Figure 1.c. The reason behind this decision is that transitions at the output of each gate are caused by transitions at the inputs of the gate. An input with large transition probability is usually expected to cause more transitions at the output of the gate than another input with smaller transition probability. Therefore, we assume that the contribution of each input line to the transitions of the output line, is equal to its transition probability divided by the sum of the transition probabilities of all inputs. In the extreme case, where an input receives a constant logic value (i.e. the transition probability of the input is 0), then all transitions at the output are caused by transitions at the remaining inputs of the gate.

In the case that one or more of the inputs of a gate have been already dropped, the weight of the output must be propagated backwards until it is shared among not dropped lines. The circuit is considered to be divided into logic levels, where a gate belongs to level k only if it is driven by gates belonging to levels $k-1$, $k-2$, ... A line resides at level k if it is the output of a gate belonging to level k. The algorithm considers one level at a time beginning from the last and moving towards the first. When at level k one or more lines are marked their weight is propagated backwards to level $k-1$ as described in Figure 1. Then their weight is set to 0 and the marks are removed from these lines. If a line at level $k-1$, receives the propagated weight, while it is already dropped then it is marked in order to propagate the received weight backwards when level $k-1$ is considered. When a not dropped line is reached, the backward propagation for this path terminates. The backward propagation continues with levels $k-1$, $k-2$, ... until no marked lines exist. In this way, the weight of each dropped line is propagated backwards and shared to the closest not dropped lines.

3.2 Construction of the transition graph

Consider a set of test vectors $V=\{v_1, v_2, ..., v_n\}$. We construct a complete undirected graph $G=(R,E)$, where each vertex $r_a \in R$ corresponds to test vector v_a. Each undirected edge $(r_a, r_b) \in E$ represents a pair of test vectors (v_a, v_b) and a weight is calculated for this edge based on a) the weight of the primary inputs and the selected internal lines, and b) the transitions at these lines when test vector v_b follows v_a.

Each test vector is simulated and the logic value of each selected internal line is recorded. Let v_{ij} denote the logic value of the j-th bit of vector v_i and s_{ik} denote the logic value of the selected internal line k when test vector v_i is applied at the circuit inputs. If w_j, w_k are the corresponding weights of primary input line j and an internal line k respectively, as computed at the previous section, then the extended weighted hamming distance of the test vector pair (v_a, v_b) is defined as follows:

$$\mathrm{EWHD}(v_a, v_b) = \sum_{j \in PrimaryInputs} (v_{aj} \oplus v_{bj}) W_j + \sum_{k \in SEL} (s_{ak} \oplus s_{bk}) W_k$$

where *SEL* is the set of the selected internal lines. EWHD(v_a, v_b) is used as the weight of edge (r_a, r_b) in the transition graph. After constructing the transition graph, a Nearest Neighbor [7] approach is used in order to determine the order of the test vectors that minimizes the power dissipated during testing.

4. Evaluation and comparisons

In order to evaluate the effectiveness of the proposed method and compare it with the already known methods we performed a series of experiments on the non-redundant version of all ISCAS'85 benchmark circuits and the combinational part of the two largest ISCAS'89 benchmark circuits assuming the zero delay model. It has been shown in [8] that there is a correlation between the energy dissipation of a circuit assuming a zero delay and the energy dissipation assuming a general delay model. Hence, using a zero delay approximation is reasonable for comparisons. Experiments were carried out on a 933MHz Intel Pentium III IBM PC-compatible computer with 512MB of RAM. Compacted test sets produced by the Synopsys ATPG tool were used. All test sets achieve complete fault coverage for single stuck at faults. We implemented in C programming language procedures for calculating the weight for each pair of test vectors according to our method (Extended Weighted Hamming Distance, EWHD), the method given in [5] (number of transitions at all circuit lines per test vector pair, TrperTVP), the method given in [2] (Hamming Distance, HD) and the method given in [4] (Weighted Hamming Distance, WHD). All methods mentioned above use the same greedy heuristic for finding a Hamiltonian path of minimum cost at the transition graph. In each case, we run the greedy heuristic 10 times assuming each time a different randomly selected test vector as the initial vector (v_a). The best result among them was selected.

Table 1 summarizes the results. The first two columns correspond to the name and test set length of each benchmark circuit. The column with the name "ATPG_Tr" shows the total number of transitions when the test vectors are applied in the random order in which they are derived by the ATPG tool. The columns with names "HD", "WHD" and "TrperTVP" show the reduction of the transitions of each method compared to the ATPG_Tr case. It is clear that in all cases the method given in [5] dominates the other two. The Weighted Hamming distance method performs better than the Hamming distance method. For the proposed method we run 3 experiments for each circuit selecting 1%, 5% and 10% of the internal lines.

Table 1. Reductions on total number of transitions

Circuit	Vectors	ATPG_Tr	Savings (%) HD [2]	WHD [4]	TrperTVP [5]	Proposed (% of Internal Lines) 1%	5%	10%
c432	52	6098	23.5	24.1	42.1	41.8	41.6	42.4
c499	59	6610	40.5	41.1	53.1	43.2	53.1	53.1
c880	49	14836	21.2	22.3	29.7	25.5	27.1	29.3
c1355	86	31369	45.6	45.7	50.9	49.6	50.6	49.8
c1908	117	71063	30.9	33.4	38.9	36.1	38.5	38.8
c2670	92	57731	19.9	25.2	32.3	30.8	31.9	32.0
c3540	206	206780	27.2	31.8	43.5	41.2	43.4	43.2
c5315	114	217608	10.6	15.4	20.8	19.7	20.4	20.5
c6288	27	53271	17.2	19.8	20.3	17.9	20.1	19.9
c7552	162	418988	23.6	27.2	35.2	33.3	35.0	35.1
s38417	319	4507743	21.7	22.5	26.6	26.3	26.5	26.6
s38584	313	3722435	28.1	29.4	35.3	35.2	35.2	35.3

Table 2. Peak power reductions

Circuit	ATPG_Tr	Savings (%) HD [2]	WHD [4]	TrperTVP [5]	Proposed (% of Internal Lines) 1%	5%	10%
c432	180	14.4	19.4	25.6	35.6	33.3	37.2
c499	211	25.6	25.6	34.6	37.9	37.9	46.9
c880	413	9.7	13.1	20.1	24.5	23.2	18.2
c1355	509	18.5	26.1	24.6	31.2	13.4	19.6
c1908	883	22.2	19.1	32.4	28.1	24.0	34.5
c2670	931	9.7	14.5	25.5	20.7	4.7	11.2
c3540	1355	12.5	14.2	12.4	23.6	26.1	15.4
c5315	2305	5.5	11.8	17.6	15.1	12.8	15.3
c6288	2513	10.3	15.0	14.2	15.0	15.0	13.7
c7552	3145	-0.1	4.7	19.3	7.2	16.0	6.7
s38417	16015	6.7	6.1	8.7	8.0	12.1	10.3
s38584	16682	1.8	4.9	8.9	10.5	9.2	9.2

Table 3. Computational time comparison (sec)

Circuit	Vectors	HD [2]	WHD [4]	TrperTVP [5]	Proposed (% of Internal Lines) 1%	5%	10%
c2670	92	<1	<1	10	1	1	1
c3540	206	<1	<1	114	1	1	1
c5315	114	<1	<1	80	2	1	1
c6288	27	<1	<1	4	3	1	1
c7552	162	<1	1	256	3	3	3
s38417	319	<1	33	26136	193	188	181
s38584	313	<1	28	30949	224	223	217

From the results it is obvious that in most cases the reduction achieved by the proposed method approaches the reduction of the TrperTVP method even by selecting just the 1% of the internal lines.

In our experiments the instantaneous power is estimated as the number of transitions caused by test vector pair (v_i, v_j). The peak power is the maximum of the instantaneous power over all test pairs of the test vector sequence. In Table 2 we can see the peak power reduction attained by each method. The test vector ordering methods given in [2-5] as well as the proposed one try to find a Hamiltonian path with the minimal sum of weights, which implies energy minimization. Since the cardinality of the test set is kept constant, the average power reduction is equal to the energy reduction. As a by-product of the ordering methods we expect reduction of the peak power in many cases. We can see that peak power reduction is achieved in all cases except for one in the HD method. The peak power reduction achieved by our method is in many cases higher than that obtained by the other methods.

Table 3 shows the computational time required for the construction of the graph by each method for the larger ISCAS85 and the two ISCAS89 circuits. As it is expected, the TrperTVP method requires long computational time that grows exponentially as the circuit size and the number of test vectors increases. The HD and the WHD methods are less computationally expensive than the proposed method but, as shown in Table 1, the power reductions achieved are less than those of the proposed method. Therefore, the proposed method is the most power-time efficient method. Results in Table 3 indicate also that the computational time of the proposed method decreases when the percentage of selected internal lines increases. This is justified by the fact that when the percentage of selected internal lines increases then the computational time required for back-propagating the weights of the dropped lines on the remaining lines decreases due to less number of internal lines that have to be dropped. However in this case, the memory required for the proposed method increases since more internal lines have to be recorded during logic simulation.

5. Conclusions

In this paper we have proposed a new test vector ordering technique for reducing the average power during test application. As we have shown energy as well as peak power dissipation are also reduced. The great advantage of the proposed method is that, although this method requires small computational time, it achieves power reduction close to that of the most elaborate methods [3, 5] which require $O(t^2)$ simulations where t is the test set cardinality and hence cannot be applied when the circuits and/or the test sets are large.

6. References

[1] P. Girard, "Survey of Low-Power Testing of VLSI Circuits", IEEE Design & Test of Computers, pp. 82-92, May-June 2002.

[2] P. Girard et al., "Reducing Power Consumption during Test Application by Test Vector Ordering", IEEE Int. Symposium on Circuits and Systems, pp. 296-299, 1998.

[3] V. Dabholkar et al., "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", IEEE Trans. on CAD, vol. 17(2), pp. 1325-1333, Dec. 1998.

[4] P. Girard et al., "A Test Vector Ordering Technique for Switching Activity Reduction during Test Operation", 9th Great Lakes Symp. on VLSI, pp. 24-27, 1999.

[5] Z. Luo et al., "Test Power Optimization Techniques for CMOS Circuits", 11th Asian Test Symposium, pp. 332-337, 2002.

[6] P. H. Bardell, W. H. McAnney, J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Willey and Sons, 1987.

[7] D. Johnson and L. McGeoch, "The Traveling Salesman Problem: A case study", In Local Search in Combinatorial Optimization (eds. E. Aarts, J. Lenstra) J. Wiley and Sons, 1997.

[8] A. Shen et al., "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks", Proc. of Int. Conference on CAD, pp. 402-407, 1992.

0-7695-2097-9/04 $20.00 © 2004 **IEEE**

IEEE COMPUTER SOCIETY