Dynamic Power Minimization During Combinational Circuit Testing as a Traveling Salesman Problem

Artem Sokolov, Alodeep Sanyal, Darrell Whitley, Yashwant Malaiya
Department of Compute Science
Colorado State University
Fort Collins, CO 80523
{sokolov, alodeep, whitley, malaiya}@cs.colostate.edu

Abstract- Testing of VLSI circuits can cause generation of excessive heat which can damage the chips under test. In the random testing environment, high-performance CMOS circuits consume significant dynamic power during testing because of enhanced switching activity in the internal nodes. Our work focuses on the fact that power minimization is a Traveling Salesman Problem (TSP). We explore application of local search and genetic algorithms to test set reordering and perform a quantitative comparison to previously used deterministic techniques. We also consider reduction of the original test set as a dual-objective optimization problem, where switching activity and fault coverage are the two objective functions.

Keywords: Combinational circuit, testing, test vector, automatic test pattern generator (ATPG), fault coverage, traveling salesman problem (TSP), genetic algorithm application, weight-biased edge crossover, multi-objective genetic algorithm

1 Introduction

The growing size of very large scale integration (VLSI) circuits, high transistor density, and popularity of low-power circuit and system design are making minimization of power dissipation an important issue in VLSI design. The amount of heat generated limits the density of a chip. Reduction of power dissipation also permits the use of smaller package size. This can reduce weight of portable products and prolong battery life.

Power dissipation during test application also plays a key role. Often during testing, test patterns leading to much larger power dissipation are applied which never occur during the normal mode of operation of the given circuit. Excessive power dissipation during testing could damage the chip, or prevent periodic testing of such equipment. In the case of multi-chip modules, it has been observed that the potential advantages in circuit density and performance of the technology cannot be realized without access to fully tested, unpackaged integrated circuits or what are called "bare die". Absence of packaging precludes the use of traditional heat removal techniques during the bare die testing. In such cases, power dissipated during testing can adversely affect the overall yield, thus adding to the production cost [1].

Prior experimental results [1], [2], [3], [4] show that test vector ordering in the context of combinational and sequential circuit testing can reduce power dissipation by an order of magnitude compared to the original unordered test set generated by a traditional automatic test pattern generator (ATPG). The test set re-ordering problem can be reduced to the well-known traveling salesman problem (TSP), where the individual test vectors are the cities and the Hamming distance between any two test vectors is the distance between those two cities [1], [15].

Chattopadhyay and Choudhary have shown [2] that by sacrificing a small amount of fault coverage it is possible to select a representative subset of test vectors generated by an ATPG and achieve a further decrease in power dissipation. Fault coverage is defined as the number of faults detected divided by the total number of faults under a given fault model [20], [21].

One part of this work is focused on quantitative comparison between two search methods to TSP: a well-known 2-opt heuristic and a genetic algorithm-based approach. Selecting an appropriate crossover operator for genetic algorithms (GAs) is important when dealing with permutation optimization problems. Many scheduling problems [19] can be categorized as order-based, because relative position of elements in a permutation matters a great deal. The traveling salesman problem, on the other hand, is an adjacency-based problem where the focus shifts from relative ordering to adjacency. That is, for two elements A and B of a permutation, the fact that A comes before B is no longer as important as whether A is adjacent to B, usually due to some cost associated with going from A to B. We make use of weight-biased edge crossover [12] for test set reordering.

We also investigate a potential for combining 2-opt with the GA approach. This yields a marginal improvement in power dissipation on several benchmark circuits. All results are compared to deterministic techniques that have been previously applied to the problem of dynamic power minimization.

In the other part of this paper we revisit the idea of test set reduction by highlighting the trade-off between high fault coverage and low power dissipation. Deb's NSGA-II [13] is used to construct a *Pareto-front* of multiple subsets from the original generated test vectors. We argue that generating a Pareto-front provides more flexibility in selecting a proper solution for some particular application than generating a single solution using a fixed weighted objective function that combines fault coverage and power dissipation, as observed in [2].

2 Background and Related Work

The two components of power dissipated in a CMOS circuit [11] are i) *static dissipation* due to leakage current through the channel and the tunneling current through the gate oxide drawn continuously from the power supply (P_{st}), and ii) *dynamic dissipation* due to switching transient current (P_{sc}) and charging and discharging of load capacitances

π_1							π_2						
Test vector				Output vector			Test vector				Output vector		
i_3	i_2	i_1	i_0	k	f_0	f_1	i_3	i_2	i_1	i_0	k	f_0	f_1
0	0	0	1	0	0	0	1	0	0	0	0	0	1
1	1	1	1	1	1	1	0	0	0	1	0	0	0
0	0	1	1	1	1	0	0	0	1	1	1	1	0
1	0	0	0	0	0	1	1	1	1	1	1	1	1
Switching activity				2	2	3	Switching activity				1	1	2

Table 1: Switching activity for two test vector orderings

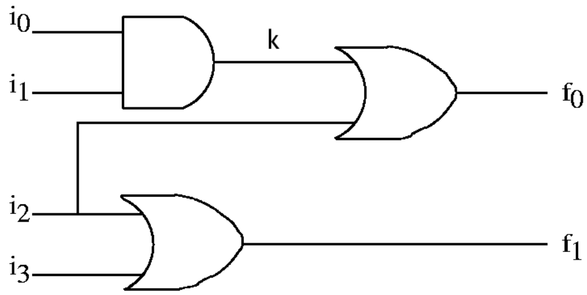

Figure 1: An example combinational circuit

(P_d). The total power dissipation (P_{total}) is given by

$$P_{total} = P_{st} + P_{sc} + P_d \tag{1}$$

As in [1], P_{st} and P_{sc} are not considered here.

P_d is the power required to charge and discharge the output capacitance load of every gate. P_d is approximated as follows [1]:

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

where C is the output capacitance, V_{DD} is the supply voltage, N_G is the total number of gate output transitions ($1 \to 0$ and $0 \to 1$), and f is the clock frequency.

Equation (2) 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 combinational circuit C when inputs change from T_i to T_{i+1}. Then

$$P_C(T_i, T_{i+1}) = \sum_{j \in SetofNodes} 1/2 \times C_j \times V_{DD}^2 \times N_{G_j} \times f \tag{3}$$

Thus, power dissipated at a node is proportional to the number of transitions (N_G) at that node. This depends on the gate delays. Under the zero-delay model, all gates are assumed to have zero delay. The four possible transitions and the corresponding logic levels in this model are:

- *static zero*: logic level remains zero
- *static one*: logic level remains one
- *rising*: logic level changes from zero to one
- *falling*: logic level changes from one to zero

The alternative is the general-delay model where gates can have arbitrary delays, and, therefore, takes the glitches into account. In this paper, we have assumed the zero-delay model.

Given a test set $T = \{t_1, t_2, ..., t_n\}$, the total switching activity of a combinational circuit C is a function of the relative ordering of the test vectors (t_i) applied to it.

To illustrate this point, let us take a look at an example combinational circuit shown in Figure 1. The circuit has four primary inputs (i_0, i_1, i_2 and i_3) and two primary outputs (f_0 and f_1). The only internal output of the circuit is denoted by k. A test set with four test vectors {0001, 0011, 1000 and 1111} is applied in two different orders: π_1 = {0001, 1111, 0011, 1000} and π_2 = {1000, 0001, 0011, 1111}. As observed in Table 1, the total switching activity of all the gate outputs (i.e. k, f_0, and f_1) for π_1 is 2+2+3 = 7 and that for π_2 is 1+1+2 = 4.

We can also evaluate the stuck-at fault coverage (defined in [21]) for this example circuit. For the first test sequence π_1, the fault coverage is 13/18 = 72.22%. The second test sequence will yield the exact same value because the exact same test vectors have been applied. Relative order of test vectors only affects switching activity in combinational circuits, not the fault coverage. Therefore, π_2 would cause less power dissipation compared to π_1 with the same fault coverage and stands as a better choice in this instance.

Power dissipation issues are addressed at various stages of circuit design. For example, circuit synthesis to reduce the average switching activity is described in [5]-[7], technology mapping that targets low power dissipation is considered in [8] and [9], and physical design for low power is considered in [10].

In the domain of circuit testing, low-power dissipation test methods have been investigated thoroughly for combinational and sequential circuits. Dabholkar et al. [1] have proposed several heuristics both for combinational circuits and scan-based sequential circuits. They show that computing an optimal order of the test vectors such that the switching activity of a combinational circuit is minimized is an NP-hard problem. They categorize their heuristics for combinational circuits as with or without repetition of test vectors. Christofides's heuristic and a greedy heuristic are used for the case of test vector reordering without repetition. Christofides's heuristic uses a minimum spanning tree based method to find a Hamiltonian path of the transition graph composed from the test set; the algorithm has $O(n^3)$ complexity. A greedy heuristic is also proposed which exhibits better running time for all the benchmark circuits. Kruskal's minimum spanning tree algorithm is used for the second case where repetition of test vectors is allowed. However, in the context of dynamic power minimization during testing, a test set without repetition of test vectors is always better than a test set with repetition in the sense that repeated vectors do not contribute to the increment of fault coverage but increase the total switching activity.

A genetic algorithm-based approach for combinational circuit testing was proposed by Chattopadhyay and Choudhary [2]. While the actual vector reordering was done using

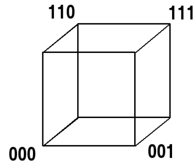

Figure 2: Vertices of a hypercube from the previous example

$O(n^2)$ Prim's Algorithm, the chromosomes of Chattopadhyay's genetic algorithm were used to represent subsets of the original test set. Using a number of operators, they were able to find a remarkably low value for switching activity at the cost of reduced fault coverage. More specifically, a 3-4% decrease in fault coverage has been demonstrated to yield a 70% reduction in switching activity [2].

3 Problem Definition

As pointed out by Latypov [15], reordering of a test set is the well-known traveling salesman problem. Dabholkar et al. give the formal definition [1] as follows: given a combinational circuit C and a set of input vectors $T = \{t_1, ..., t_n\}$, compute an optimal input order $\langle s_1, ..., s_n\rangle$ of T where $s_i = t_{\pi(i)}$, where π is a permutation of $\{1, ..., n\}$ such that

$$\sum_{i=1}^{n-1} P_C(s_i, s_{i+1}) \textit{ is minimized}$$

for the power dissipation function P_C (refer to equation (3)). The proof of NP-hardness for this problem is presented in [16].

Because power dissipation is directly proportional to switching activity, we can simplify the problem definition to the following model. Consider combining the output of all nodes into a single bitstring. This yields a hypercube vertex in n dimensions, where n is the bitstring length. The goal can then be expressed as finding the shortest path through the subset of vertices representing the outputs of all nodes of a circuit. Note that switching activity associated with going from test vector T_i to test vector T_{i+1} is the Hamming distance between two corresponding bitstrings. This distance is equivalent to the number of edges traversed when traveling from one vertex to the other.

Figure 2 ties these concepts with our example from the previous section. The output of all gates during application of four test vectors in Table 1 yields four vertices of a 3-dimensional cube: {000, 111, 110, 001}.

In the context of test set reduction, the problem definition becomes multi-objective with $P_d = \sum_{i=1}^{m-1} P_C(s_i, s_{i+1})$ being minimized for some subset of T with cardinality $m \leq n$, where the subset itself is chosen such that it provides maximum fault coverage while yielding the lowest value of P_d. Due to the trade-off involved, there is rarely a single answer to this multi-objective problem and a set of optimal solutions needs to be located.

4 Techniques Applied

A number of deterministic heuristics have already been applied to test vector reordering [1], [2]. While the three heuristics proposed by Dabholkar *et al.* [1] for combinational circuits show significant reduction in the amount of switching activity, the use of Prim's algorithm for test vector reordering by Chattopadhyay and Choudhary [2] turns out to be not the most efficient choice for this problem.

For Euclidean TSP problems where the triangle inequality holds, Prim's algorithm is used to construct a minimal spanning tree over the set of cities; the resulting spanning tree must be less than the optimal Hamiltonian circuit which visits all of the cities. A non-Hamiltonian circuit can be constructed that traverses each edge of the spanning tree twice: in effect, the first traversal goes "out" over the spanning tree, and the second traversal returns to the origin. Using the triangle inequality, the non-Hamiltonian circuit can be converted into a Hamiltonian circuit by dropping cities from the route if they have already been visited. Denote the cost of this solution by C_s. Let C_m denote the cost associated with the minimal spanning tree, and let C^* denote the optimal solution to the TSP. The advantage of this method is that it guarantees a solution that is within a factor (i.e., a ratio bound) of 2 of the optimal solution for Euclidean TSP problems [22]:

$$C_m < C^* \leq C_s \leq 2C_m$$

However, for the general TSP (where the triangle inequality does not hold), one can show that this method *cannot* guarantee a ratio bound on the quality of the optimal solution unless $P = NP$. As pointed out by Dabholkar *et al.*, the triangle inequality does hold for test vector reordering problem if the zero-delay model is assumed, though the property is not guaranteed to hold for the general-delay model.

The major contribution of this paper is application of two search techniques to the problem of dynamic power minimization. All previous work on test vector reordering in combinational circuits has utilized algorithms that constructed a path through the corresponding TSP graph one task at a time in a deterministic fashion [1], [2]. We present a comparative study between these deterministic techniques and two search algorithms that start with some initial random path and iteratively refine it. The first approach we consider is a well-known 2-opt heuristic. This heuristic is the most basic form of local search for TSP problems and, therefore, serves as a good baseline for comparison. The second approach is a specialized crossover operator for a genetic algorithm framework. The method has been demonstrated to perform very well on TSP benchmarks by its authors [12]. Following is a more detailed description of each technique.

4.1 2-opt Heuristic

The 2-Opt local search method is a very general and robust local search operator for the TSP. The idea behind 2-opt is quite simple. Starting with some initial path, a pair-wise edge comparison is performed to decide whether two edges should swap vertices. For instance, in Figure 3, the decision is based on sums $BC + FG$ and $BF + CG$ for two edges BC and FG. If $BF + CG < BC + FG$, then vertices C and F are swapped to yield two new edges: BF and CG. To preserve all other edges, the order of traversal for all vertices between C and F is reversed. Such pair-wise edge comparison is performed iteratively for all possible pairs of edges until the path converges to a locally optimal solution.

When the TSP problem is Euclidean, the application of 2-Opt until a local optimum is found guarantees that the resulting Hamiltonian circuit has no crossed edges. Con-

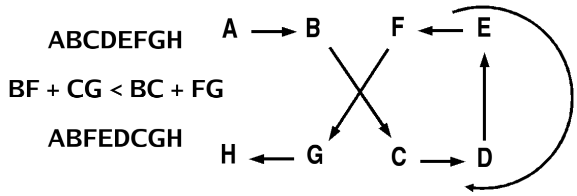

Figure 3: Example of 2-opt heuristic

verting a minimal spanning tree into a Hamiltonian circuit will typically leave crossed edges, so that the resulting solutions are not locally optimal under 2-Opt. When the TSP problem is non-Euclidean, it is no longer meaningful in the general case to refer to crossed edges; nevertheless the solutions produced by local search using 2-Opt are still locally optimal under the same neighborhood structure.

It is easy to see that the complexity of a single 2-opt iteration is $O(n^2)$ partial (or edge) evaluations, where n is the path length. Every edge has to be compared to every other edge and the number of edges is linearly proportional to the path length. However, there is no upper bound on the number of iterations and the search can potentially take $O(n!)$ time. For our experiments, we have implemented the next-descent version of 2-opt, where an iteration is completed at the first encounter of an improving move and the search is terminated when all edge pairs have been compared with no improvement. In our experience, 2-opt converged to a local optimum in approximately $2n$ moves for smaller circuits and $3n$ moves for larger ones. The number of iterations was equal to the number of moves due to the next-descent nature of the algorithm.

4.2 Weight-Biased Edge Crossover

Julstrom and Raidl proposed the Weight-Biased Edge Crossover operator for solving TSP problems [12]. For the purposes of our work we have concentrated on its greedy version (GX1 in [12]). Our choice was guided by the fact that GX1 was not only able to locate better solutions on TSP benchmarks [17] but also showed the fastest convergence rate over its random and heuristically-guided counterparts (see comparison to IX, RX, and TX operators in [12]). The latter feature is particularly important when dealing with expensive objective functions.

Weight-Biased Edge Crossover (WBEX) takes two parent permutations and generates a child in the following fashion. Pick a random vertex as a starting point. Generation of a path is then an iterative process where at every iteration the shortest edge from the two parents connecting the current vertex to an unvisited one is chosen. If there is no such edge, an unvisited vertex is chosen at random.

Generating a new child is approximately $3n$ edge evaluations, where n is the chromosome length. To be exact, adding the first vertex to the child involves evaluating at most 4 edges — two from each parent — to find the shortest edge. Adding all consecutive vertices requires evaluation of at most 3 edges, because the tour cannot go back. A total of $n-1$ vertices need to be added to the child, which yields an upper bound of $3*(n-2)+4 = 3n-2$ evaluations.

4.3 Multi-objective GA

Another goal of this paper is to explicitly demonstrate the trade-off between low switching activity and high fault coverage. Test set reduction is a problem where the goal is to find and order a subset of the original generated test vectors that minimizes switching activity while maintaining high fault coverage. The two extremes are, of course, the entire original test set (high fault coverage and high switching activity) and the null set (zero switching activity and zero fault coverage). The previous work on test set reduction [2] has assumed fixed weight values for the two objectives. We argue that optimality of a solution depends, in some sense, on application. If a subset A results in lower switching activity as well as lower fault coverage than subset B, deciding whether A is better than B is not obvious. Therefore, it might be desirable to present a set of optimal solutions that the end-user will be able to choose from.

We have used Deb's NSGA-II [13] to generate sets of solutions. NSGA-II is a fast and elitist genetic algorithm framework designed for dealing with multi-objective optimization problems. It involves no additional parameters and any traditional single-objective GA framework can be easily scaled up to it. While highly competitive, NSGA-II is not the only technique capable of approaching multi-objective problems. A number of other techniques have been proposed in the literature, both in the context of genetic algorithms and evolutionary strategies [26], [24], [25]. Following is a brief description of how NSGA-II works.

Given two solutions A and B to a multi-objective problem, we can say that A *dominates* B only if it is better or equal to B with respect to every objective and strictly better than B in at least one objective. In our case, A dominates B if A yields lower switching activity **and** higher or the same fault coverage than B, or A yields lower or the same switching activity **and** higher fault coverage. Using this definition, one can now extract a set of non-dominated individuals from a population. This set of solutions is the *first non-dominated Pareto-front*. All remaining solutions are dominated by one or more members of the first Pareto-front and can, therefore, be considered worse.

We can further extract the *second non-dominated Pareto-front* from the remaining portion of the population. All solutions in the second front are worse than members of the first Pareto-front but better than everything that doesn't belong to the first or the second fronts. Continuing to extract one non-dominated Pareto-front after another, we can group the entire population into fronts with a natural ordering: individuals belonging to a front with lower index are considered more fit.

It is common to find multiple fronts over the first few generations. However, as the search progresses and the entire population converges to the first Pareto-front, further ordering of individuals requires an additional metric. NSGA-II employs *crowding distance* to sort individuals within the same front. To compute crowding distance for an individual, we average the distances to its immediate neighbors along the same front in every dimension (dimensions correspond to objective functions). Larger crowding distance represents more fit individuals.

The intuition behind this metric is to favor sparsely populated regions of a front when selecting individuals for recombination. By participating in recombination, these individuals are more likely to create offsprings that will "fill in the gaps" in the front yielding a more diverse set of solutions. Diversity is an important issue in any multi-objective optimization problem. Uniformly distributed solutions along the optimal Pareto-front provide more flexibility to the end-user than, for example, one dense cluster of solutions.

5 Experimental Results

5.1 Test Set Reordering

The first set of experiments dealt exclusively with test set reordering. No reduction was performed at this stage. We used ATALANTA [14] — an automatic test pattern generator — in its default mode to generate a compacted test set with very high fault coverage for ISCAS85 combinational benchmark suite. A short summary of this suite is given in Table 2. The last column presents the average number of test vectors generated by ATALANTA (these averages were computed by running ATALANTA 30 times for every circuit).

A fault simulator HOPE [18] (designed to evaluate test coverage) was run on a modified circuit definition — where all outputs of internal gates have been declared as primary to retrieve outputs of internal gates as well corresponding to every test vector. After computing the transition matrix for TSP, we applied the next-descent implementation of 2-opt and a generational GA using switching activity as the objective function. The experiments using the GA utilized a population size of 100, two hundred generations, tournament selection, WBEX, and no mutation. The results over 30 trials can be found in Table 3, where the best technique for every benchmark problem has been highlighted in bold. The average percent improvement over the total switching activity of an unordered set by using the highlighted technique is given in the last column. Although WBEX generally outperforms 2-opt, it does so with statistical significance under $p < 0.05$ only for c432, c499, and c1355.

In our experience, 2-opt runs significantly faster than the GA-based approach because the former does not have the overhead associated with maintaining a population. All n^2 edges of the TSP can be computed once at the beginning and stored internally as a matrix. Performing an edge evaluation, associated with going from one test vector to another, implies nothing more than accessing an entry of that matrix.

Table 4 shows how our results compare to deterministic heuristic-based methods used in the literature [1], [2]. The genetic algorithm-based approach and 2-opt prove to be competitive, particularly on the smaller benchmark circuits. Not surprisingly, Prim's algorithm demonstrated the worst performance and stands as a poor choice for test set reordering.

Besides running 2-opt and WBEX separately, we have also combined the two approaches (column WBEX + 2-opt in Table 3) by applying 2-opt to every individual in the population at every generation. Such combination yields marginal improvement for several benchmarks, though insignificant under $p < 0.05$ conditions.

To investigate the implications of this combination, we have selected one of the benchmark problems where WBEX and WBEX+2-opt have demonstrated strikingly similar performance, namely c2670. For this circuit we plotted the amount of improvement 2-opt has provided to the population. The improvement values have been averaged over all 100 individuals at every generation and appear in Figure 4.

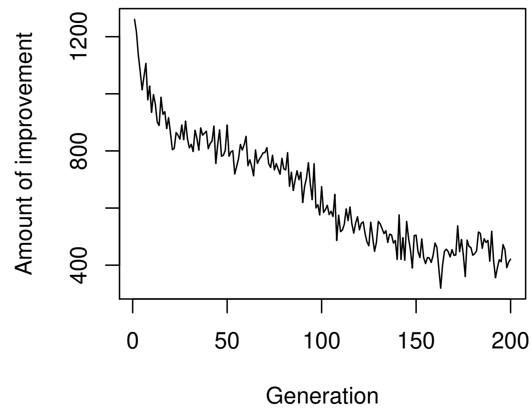

Figure 4: Additional improvement by 2-opt to individuals in a population. The value at every generation is the average improvement over the entire population.

There are two things to be noted from Figure 4. First, a progressive decay in the amount of improvement is clearly visible. This is intuitive since convergence closer to a global optimum would naturally leave less room for improvement. The second thing to note is the vertical axis scale. The biggest amount of improvement has been observed during population initialization (not plotted in Figure 4), which is to be expected as the individuals are generated randomly. However, throughout generations the improvement in switching activity remained consistently less than 1200 transitions or approximately .3% for this benchmark problem. This suggests that either solutions generated by WBEX fall very close to local optima or that basins of attraction are rather shallow. Either case is characterized by almost non-existent difference in performance between WBEX and WBEX+2-opt.

5.2 Test Set Reduction

To address the problem of test set reduction we have modified our GA framework to NSGA-II with individuals represented as subsets of the original compacted test set generated by ATALANTA. The two objective functions used were (1 - Fault Coverage) and switching activity. To compute fitness of an individual, HOPE simulator [18] was run to determine the fault coverage and 2-opt was used to find a locally optimal ordering from which switching activity could be computed. We borrowed Chattopadhyay's crossover and mutation operators [2]. The former is an equivalent of single-point crossover [23] where each of the parents is split in two, and the parts are recombined to produce two children. Special care is taken to ensure no duplicate test vectors occur. The mutation operator either removes or appends new vectors from the original test set. When using a population size of 100, we noticed that mutation did not yield any significant improvement in performance.

Circuit	# inputs	# outputs	# inverters	# other gates	# test vectors
c432	36	7	40	120	49
c499	41	32	40	162	53
c880	60	26	63	320	53
c1355	41	32	40	506	85
c1908	33	25	277	603	117
c2670	233	140	321	872	107
c3540	50	22	490	1179	149
c5315	178	123	581	1726	118
c6288	32	32	32	2384	31
c7552	207	108	876	2636	207

Table 2: Specification of the ISCAS85 benchmark circuits

Circuit	% Fault	Switching Activity				
	Cov.	Unordered	2-opt	WBEX	WBEX + 2-opt	Ave. % Red.
c432	99.2	2831 ± 160.7	1803 ± 89.87	**1751 ± 97.21**	1795 ± 96.71	38.15
c499	98.9	4233 ± 96.32	3178 ± 51.00	**3109 ± 54.46**	3153 ± 52.11	26.55
c880	100	7445 ± 466.6	5504 ± 293.1	**5416 ± 293.7**	5481 ± 277.6	27.25
c1355	99.5	16457 ± 211.2	12793 ± 179.7	**12622 ± 172.6**	12684 ± 174.0	23.30
c1908	99.5	41506 ± 1303	28731 ± 886.2	28420 ± 919.0	**28394 ± 886.3**	31.59
c2670	95.7	46223 ± 1631	31624 ± 962.3	31492 ± 883.9	**31488 ± 864.2**	31.88
c3540	96.0	85635 ± 2289	55937 ± 1373	55903 ± 1433	**55388 ± 1364**	35.32
c5315	98.9	109769 ± 3508	89309 ± 2918	**88840 ± 2915**	88852 ± 2838	19.07
c6288	99.6	30376 ± 2110	26734 ± 1731	**26499 ± 1704**	26695 ± 1716	12.76
c7552	98.3	289289 ± 8286	198799 ± 5700	201741 ± 5962	**198140 ± 5516**	31.51
		Average				27.74

Table 3: Results on test set reordering over 30 trials in the mean ± st.d format. The smallest mean value for each circuit is highlighted in bold.

Circuit	Total Switching Activity					
	2-opt	WBEX	Christofides's [1]	Greedy [1]	Kruskal's [1]	Prim's [2]
c432	1803	1751	3004	2464	2102	2911
c499	3178	3109	5450	4920	4340	4264
c880	5504	5416	8918	8183	6762	7657
c1355	12793	12622	13949	13315	11008	15169
c1908	28731	28420	25608	24251	20265	32918
c2670	31624	31492	25296	21917	17174	43374
c3540	55937	55903	72458	66443	58089	63600
c5315	89309	88840	107004	93573	79180	103058
c6288	26734	26499	23970	20792	16036	28288
c7552	198799	201741	163742	142641	120763	234536

Table 4: Comparison of evolutionary methods to deterministic heuristics

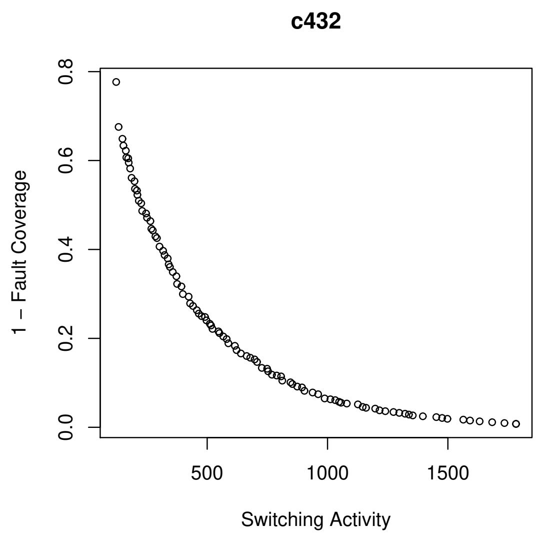

Figure 5: An example Pareto front

Figure 5 shows the results obtained from a population size of 100 after running NSGA-II framework for 200 generations on c432. Every point on a graph corresponds to a particular test vector subset. Note that a Pareto-front represents a gradient of solutions and is better suited for the end-user — who can choose a particular solution depending on his/her application — than a single solution obtained through fixed weighting of the objective functions. The granularity of this gradient can be controlled by modifying the population size. Larger values of the population size will yield more points along a Pareto-front and, thus, a finer gradient.

6 Conclusion and Future Work

This work has touched on a number of important points relating to dynamic power dissipation minimization during combinational circuit testing. We have performed a quantitative comparison between two TSP techniques well suited for this problem, demonstrating that a GA-based approach with proper crossover operator is able to find better solutions than 2-opt heuristic and requires fewer edge evaluations for larger circuits. The second contribution of this paper was introduction of Pareto-fronts to the problem of test set reduction. Our results were close to but not better than what reported in [2] for this problem. We believe that this is due to a much smaller and compact set of test vectors we used in our experiment. It would be interesting to see Chattopadhyay's experiments [2] rerun in a NSGA-II framework to generate a Pareto-front of optimal solutions rather than a single point.

We are currently extending this work to scan-based sequential circuits [21]. Testing a scan-based sequential circuit involves useless power dissipation during the scan-in and the scan-out of the states before and after the application of the test. This power dissipation can be decreased through scan-latch reordering.

We are also experimenting with more accurate estimates of power consumption. Load capacitance of a gate (as defined in Equation 2) depends on the number of gates connected to its output and, therefore, can be approximated by its fan-out count. The load capacitance provides weighting on switching activity of a gate and can potentially serve as a more accurate objective function. We found no literature that takes this into consideration.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0117209.

Bibliography

[1] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. M. Reddy. Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No. 12, December 1998, pp. 1325-1333*

[2] S. Chattopadhyay and N. Choudhary. Genetic Algorithm based Approach for Low Power Combinational Circuit Testing. *16th IEEE International Conference on VLSI Design, pp. 552-557, January 4-8, 2003*

[3] P. Flores, J. Costa, H. Neto, J. Monterio, and J. Marquessilva. Assignment and Reordering of Incompletely Specified Pattern Sequences Targeting Minimum Power Dissipation. *12th International Conference on VLSI Design, pp. 37-41, January 1999*

[4] F. Corno, P. Prinetto, M. Rebaudengo, and M. S. Reorda. A Test Pattern Generation Methodology for Low Power Consumption. *16th IEEE VLSI Test Symposium, pp. 453-457, April 1998*

[5] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer. On average power dissipation and random pattern testability of CMOS combinational logic networks. *Proc. ACM/IEEE Int. Conf. Computer Aided Design, 1992, pp. 402-407*

[6] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. *Proc. ACM/IEEE Int. Conf. Computer Aided Design, 1992, pp. 253-259*

[7] K. Roy and S. Prasad. SYCLOP: Synthesis of CMOS logic for low power application. *Proc. Int. Conf. Computer Design.Oct. 1993, pp. 464-467*

[8] C.-Y. Tsui, M. Pedram and A. Despain. Technology decomposition and mapping targeting low power dissipation. *Proc. ACM/IEEE 30th Design Automation conference, 1993, pp. 68-73*

[9] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. *Proc. ACM/IEEE 30th Design Automation Conference, 1993, pp. 74-79*

[10] H. Vaishnav and M. Pedram. Pcube: A performance driven placement algorithm for low power designs. *Proc. EURO-DAC, Sept. 1993, pp. 72-77*

[11] H. Weste and K. Eshraghian. Principles of CMOS VLSI Design: A Systems Perspective, 2nd ed. *Reading, MA: Addison-Wesley, 1992*

[12] B. A. Julstrom and G. R. Raidl. Weight-Biased Edge-Crossover in Evolutionary Algorithms for Two Graph Problems. *Proceedings of the 2001 ACM symposium on Applied computing, 2001, pp.321-326*

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. Proceedings of the Parallel Problem Solving from Nature VI Conference. *Springer. Lecture Notes in Computer Science No. 1917: 849-858*, 2000.

[14] D. S. Ha, ATALANTA: An ATPG Tool, Bradley Department of Electrical Engineering, Virginia Polytechnic and State University, Blacksburg, VA, 1994.

[15] Latypov P.K. Energy Saving Testing of Circuits. *Automation and Remote Control, April 2001, vol.62, no.4, pp. 653-655(3)*

[16] V. Dabholkar. Optimization problems in low-power and stress testing. *Ph.D. dissertation, State University of New York at Buffalo, 1996*

[17] G. Reinelt. The traveling Salesman: Computational Solutions for TSP Applications. *Springer, Berlin, 1994, pp. 211-213*

[18] H. K. Lee and D. S. Ha. HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, pp. 1048-1058, September 1996*

[19] E. J. Anderson, C. A. Glass, and C. N. Potts. Machine Scheduling. In E. Aarts and J. K. Lenstra, ed., *Local Search in Combinatorial Optimization, Chapter 11.* John Wiley & Sons Ltd., 1997

[20] T. W. Williams and S. Sunter. How Should Fault Coverage Be Defined? *Proceedings of 18th IEEE VLSI Test Symposium (VTS'00), Montreal, Canada, 2000*

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

[22] T. H. Corman, C. E. Leiserson, AND R. L. Rivest. Introduction to Algorithms, *McGraw-Hill, 1990.*

[23] D. Whitley. A Genetic Algorithm Tutorial, *Statistics and Computing (4):65-85*, 1994.

[24] R. Sarker and H. Abbass. Differential Evolution for Solving Multi-Objective Optimization Problems. *Asia-Pacific Journal of Operational Research, vol. 21, no.2, World Scientific, pp.225-240., 2004*

[25] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy, *Evolutionary Computation 8 (2), 149-172, 2000.*

[26] E. Zitzler and L. Thiele. Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation, 3(4), pp. 257-271, 1999.*