Genetic Algorithm based Approach for Low Power Combinational Circuit Testing *

Santanu Chattopadhyay Naveen Choudhary

Dept. of Computer Science & Engg.
Indian Institute of Technology
Guwahati
India – 781 039
Email: santanu@iitg.ernet.in

Abstract

With the advancement in automation, periodic testing of electronic circuits during their lifetime is becoming more and more important. For such a circuit, it is thus very much necessary to reduce the power requirement during the testing phase also. This paper presents a Genetic Algorithm based formulation to solve the problem of generating a test pattern set such that it has high fault coverage and low power consumption. Exhaustive experimentation done on ISCAS85 combinational benchmark suite has shown that the this tool results in upto 78% reduction in transition activity over the original test set generated by ATPGs like ATALANTA [1].

1 Introduction

With the increasing use of portable computing and wireless communication, energy dissipation has become a major concern in today's VLSI design. For example, the cost constraints of consumer electronic product typically require plastic package, which imposes a strong limit on the energy dissipation. The market acceptance of mobile application and laptop computers also depends on the operation per battery pack. Therefore, the reduction of the electrical energy consumption has become one of the most challenging areas of research in this domain. CMOS technologies have very low power consumption [2] when signals are not switching. The majority of the energy dissipation in the current day technology is due to charging and discharging of the load capacitance of logic gates, which occurs when logic gates undergo signal transitions. Hence in order to reduce power consumption, it is desirable to reduce the switched capacitance during the test mode. For such application, pseudorandom testing [3] does not prove to be effective since they often require too many clock cycles. **Automated Test Pattern Generators** (ATPGs) appear to be the better alternative, since they give a minimum set of test patterns required to test the circuit under a particular fault model.

Some approaches have been reported in the literature with the intent of generating a test pattern set which is able to minimize power dissipation during the test application in addition to the classical ATPG parameters. In [4], a scheme is presented that uses a genetic algorithm based approach for reducing the hamming distances between the consecutive patterns in the test set. As the hamming distance reduces, the switching activity in the circuit is also expected to reduce. In [5] a scheme has been proposed based on reordering the initial test pattern set and exploiting *don't cares* present in the initial set. The initial set of patterns is generated by ATPGs like ATALANTA [1], MTP etc. This technique uses a recursive *TSP (Travelling Salesman Problem)* formulation and tries to decrease the switching activity in the circuit at each recursive call. To exploit the don't cares present in the initial set, it suggests various heuristics. For sequential circuits, a low power test pattern generation methodology has been proposed in [6]. It first generates a large set of test patterns, and then selects a subset from it so that the fault coverage is high, as well as, the power consumption is reduced. However, for combinational circuits the approach will produce suboptimal results. This is because in the case of combinational circuits, for most of the fault models, fault coverage is independent of the order in which test patterns are applied to the circuit, whereas in sequential

*This work is partially supported by the reserch project sponsored by DST, Govt. of India, order no. III.5(93)/2001-SERC(Engg.)

circuit testing, test patterns need to be applied in the specific sequence only.

In this work, we have proposed a novel strategy to generate a set of test patterns that minimizes power dissipation during testing. Apart from judiciously selecting test patterns from a large set generated by ATALANTA [1], it also optimizes the order in which the selected patterns are to be applied to minimize switching of individual circuit gates under a zero gate delay model. The proposed strategy reduces switching activity in ISCAS85 benchmark circuits by more than 70%, on an average, over the compressed test pattern set generated by ATALANTA. It may be noted that the method proposed in [5] reduces switching by only 24% on an average. The approach presented in this paper can be divided into two distinct phases. Phase 1 is a *Preprocessing Phase* in which we generate a redundant set of test patterns and some associated information. Phase 2 is the *Selection Phase* that does the pattern selection and ordering using a genetic algorithmic approach utilizing the information generated in Phase 1. Section 2 details the Preprocessing Phase. Selection Phase has been presented in Section 3. Section 4 presents the experimental results.

2 Phase I: Preprocessing Phase

It consists of the following subtasks.

1. First, the tool ATALANTA [1] is used to generate a redundant set of test patterns for the circuit under test. This set is the union of the following two subsets.
   (a) The compressed test pattern set generated by ATALANTA in its default mode. Here a minimal set of test patterns are generated that ensures very high fault coverage.
   (b) A redundant set of test patterns (generated using **"– D 1"** option of ATALANTA), in which for each fault in the circuit a pattern is generated. No compression is applied by ATALANTA in this case. The patterns also contain *don't cares*.
2. Next, the don't cares from the entire set are removed using the heuristics *H4* proposed in [5]. According to this heuristic, for each bit position, the don't care bits are set to the bit that occurs more often. This attempts to reduce the switching of primary input lines when the circuit is in test mode.
3. For each of the patterns in the set, a fault simulation is performed using the efficient **ppsfp** algorithm [7] to obtain the set of faults detectable by each of the patterns. For this we assume zero gate delays.
4. For each pair of test patterns in the set, we measure the number of switching in the gates of the circuit by successive application of the patterns.

These information are used in the next phase to select a subset of test patterns that ensures high fault coverage and minimized transitions in the circuit gates.

3 Phase II: Selection Phase

The Selection Phase consists of a genetic algorithm to select a set of patterns from all the patterns generated in the Preprocessing Phase and reordering them to reduce transitions of circuit gates. The selected reordered test pattern set provides a high fault coverage and a good reduction in gate switchings. An efficient approximate *Travelling Salesman Problem (TSP)* formulation has been used to reorder the pattern set. In the following, we provide the genetic algorithmic formulation of the problem – the solution representation, genetic operators, and the cost estimation strategy.

3.1 Solution Representation

Each chromosome is represented by an array of varying size with the maximum size equal to the number of test patterns (say, *maxnumofpatns*) generated in Phase I. Each entry in this array is an integer between 1 and *maxnumofpatns*. It corresponds to the index of the pattern in the whole set of test patterns. Chromosomes are not allowed to have any duplicate test patterns.

3.2 Crossover Operator

In this genetic algorithm formulation, the selection of chromosomes participating in crossover are not uniformly random. Rather, it has been biased towards selecting chromosomes with better fitness values. For this purpose, the whole population is sorted according to their fitness values. A certain percentage of population with better fitness value is defined to be the **"Best Class"**. To select a chromosome participating in crossover, first a uniform random number between 0 and 1 is generated. If the number is greater than 0.6, a chromosome from the **Best Class** is selected randomly. Otherwise a chromosome from the entire population is selected. This approach of selecting more fit chromosomes to participate in crossover leads to the generation of better off-springs as compared to the truly random one.

After selecting two candidate chromosomes (say g_1 and g_2 with sizes n_1 and n_2 respectively) to participate in crossover, we randomly select two points (say p_1

and p_2) – one on each of the chromosomes. The two new chromosomes G_1 and G_2 are created through the following procedure.

```
for i = 1 to p1 do G1[i] = g1[i]
k = p2 + 1
while (k <= n2) do
  if g2[k] is not already existing in G1
    G1[i] = g2[k]
    i = i + 1
  endif
  k = k + 1
end while
for i = 1 to p2 do G2[i] = g2[i]
k = p1 + 1
while (k <= n1) do
  if g1[k] is not already existing in G2
    G2[i] = g1[k]
    i = i + 1
  endif
  k = k + 1
end while
```

3.3 Mutation Operator

Mutation is a very important operator as far as bringing variety into the population is concerned. As the population size is finite, the crossover operator alone cannot bring enough variation to the population. The mutation operator has been given special care in this genetic algorithm. The following operations in equal proportion are used to mutate the population.

Delete: A chromosome is randomly picked from the population. A random number of patterns are dropped from the chromosome. The patterns to be dropped are also selected randomly.

Insert: A chromosome is randomly picked from the population. An integer *num-patt-ins*, is randomly generated. This corresponds to the number of new patterns to be inserted into the chromosome. For this purpose, *num-patt-ins* number of patterns are selected randomly from the set of all test patterns generated in Phase I and are appended into the chromosome. Care is taken to ensure that the patterns already existing in the chromosome are not inserted again.

Exchange: A chromosome is picked randomly from the population. A random number of patterns from the chromosome are then exchanged with the patterns belonging to the set of all patterns generated in Phase I. The patterns to be replaced and the replacing patterns are always selected randomly ensuring that duplicate patterns do not appear in the chromosome.

3.4 Other Operators

Apart from the crossover and mutation operators, two new operators **merge** and **split** are also introduced in this genetic formulation. The objective behind these operators is to explore the possibility of creating good chromosomes from unhealthy (chromosomes with lesser fitness) part of the population. The following operations in equal proportion are used for this purpose.

Merge: This operator selects two chromosomes and then merges them together to form a new chromosome. First, two chromosomes are selected from the population such that the smaller chromosomes (having lesser number of patterns) have very high probability of being selected. Selected chromosomes are then merged together with the expectation of producing a better chromosome. Duplicates, if any, are then removed. This operation may be profitable because the smaller chromosomes may not be having enough number of test patterns to provide a high fault coverage. Hence, their merger may produce a better solution.

Split: The objective of this operator is to select a large chromosome and then split it from the middle to form two new chromosomes. A chromosome is first randomly selected from the population such that larger chromosomes have very high probability of getting selected. The chromosome is then split into two new chromosomes. This operation may result in good chromosomes because a large chromosome may be having patterns which are unnecessary, in the sense that, the faults covered by these are already covered by some other patterns in the set. Splitting the large chromosome results in smaller ones that require lesser switching in the circuit gates.

3.5 Measure of Fitness

In this section, we present the strategy to measure the fitness of a chromosome. Since the problem has the dual objective of getting high fault coverage while minimizing the gate transitions, the fitness measure has essentially got two components:

- Fault coverage
- Gate transitions.

While for a given set of test patterns, fault coverage remains the same, the number of internal gate transitions varies with the order in which patterns are applied to the circuit under test. Hence, to get minimized transitions in a circuit, we need to get a good ordering of the input patterns. In the following we provide a *Travelling Salesman Problem (TSP)* formulation of this reordering.

3.5.1 TSP Formulation

First, we construct a completely connected weighted graph (V, E), where, the set of vertices V corresponds to the set of patterns in the chromosome. Weight of the edge e_{ij} between the vertices v_i and v_j is equal to the number of transitions in the internal gates resulting from the application of pattern p_j just after the pattern p_i. Since, generally the weights of the edges satisfy *triangle inequalities (that is, it is always cheapest to go directly from vertex u to vertex w; going by way of intermediate vertex v cannot be less expensive)*, we have used the following efficient $O(n^2)$ approximate *Travelling Salesman* algorithm [8].

APPROX-TSP(G, c)

`Select a vertex` $r \in V[G]$ `to be a` *root* `vertex.`
`Grow a minimum spanning tree` T `for` G `from root` r `using` **Prim's algorithm [8].**
`Let` L `be the list of vertices visited in a preorder tree walk of` T.
`Return the Hamiltonian cycle` H `that visits the vertices in the order` L.

3.5.2 Cost Function

In this section we formulate the cost function used to measure the fitness of chromosomes in a population. Let,

X_1 = maximum fault coverage among all the chromosomes in the population, and
X_2 = maximum of total number of transitions resulting from the application of test patterns belonging to a chromosome, among all the chromosomes in the population.

Moreover, for the i-th chromosome, let fc_i be the fault coverage and pc_i be the total number of transitions. Cost of the chromosome i is then formulated as,

$$C = C_1 \times (fc_i/X_1) + C_2 \times (1 - (pc_i/X_2))$$

where, C_1 and C_2 are two empirically determined constants. Through exhaustive experimentation, the optimum values of C_1 and C_2 have been found to be 0.6 and 0.4 respectively. Fitness of a chromosome is regarded as high if the cost computed, C, is close to 1.0.

It may be noted that, the best 20% of chromosomes at any generation are directly transferred to the next generation, so that the solution does not degrade between the generations.

4 Experimental Results

In this section, we present the results of our experimentation with ISCAS85 combinational benchmark suites. The results have been tabulated in Table 1. For each of the circuits, we have first run the ATPG ATALANTA [1] in it's default mode. In this mode, ATALANTA generates a compressed completely specified set of test pattern to achieve a high degree of fault coverage. The fault coverage attained has also been noted in the Table. We have also measured the number of transitions occurring in internal circuit gates by applying this unordered test pattern set to the circuit inputs. This set has then been reordered using the TSP algorithm to generate a good reordered sequence. It has been found that the reordering results in 19.02% reduction in the gate transition on an average, as compared to the unordered test set generated by ATALANTA.

Next, we have used the genetic algorithm based approach detailed in this paper, to select a set of test patterns from a large number of patterns generated by ATALANTA (using "–D 1" option) and reorder them. The resulting fault coverage and number of transitions are noted in the Table. It can be observed that with a nominal sacrifice in the fault coverage (about 1.14% on an average), this approach achieves very high reduction in circuit gate transitions. With respect to the original unordered test set produced by ATALANTA, the reduction is 70.26% on an average, while with respect to the ordered set, the reduction is 63.18%. Such a high degree of reduction in transition activity has been possible due to the judicious selection of patterns from a very big redundant set of patterns. The column "CPU time" notes the actual CPU time required to run the GA program for different benchmarks on a Pentium III Linux machine with 64 MB RAM, 550 MHz clock. The various GA parameter utilized in the solution are as follows.

Number of generation	:	1000
Population size	:	1000
Best Class for Crossover	:	20% of population
Mutation	:	30%
Crossover	:	30%
Split	:	10%
Merge	:	10%

Table 2 presents a comparison of our approach with the best results of [5]. Results are reported for both the tools as the percentage reduction in gate transitions achieved over ATALANTA. It may be observed that the GA based approach reduces the transitions further by 24.06%, on an average, as compared to [5].

5 Conclusion

Low power BIST is important for battery operated systems. In this paper, we have used a genetic algorithm based approach to reduce the switching activity and hence the power consumption in a combinational circuit. The genetic algorithm presented makes effective use of redundancy present in the test pattern set and also exploits the reordering capability of the test patterns to a good extent. The experimental results demonstrate the feasibility and the effectiveness of the proposed approach. The results on ISCAS85 benchmark combinational circuits show that reduction in transitions are significant and ranges from 55% to 78% over the original unordered test set generated by ATALANTA. At present, we are working on reducing the time required to run the programs, and also on more accurate measure of the power optimization achieved by the scheme.

References

[1] H. Lee and D. Ha, "On the Generation of Test Patterns for Combinational Circuits," Tech. Rep. 12-93, Dept. of Electrical Engg., Virginia Polytechnic Institute and Statr University, 1993.

[2] K. Roy and S. Prasad, *Low Power CMOS VLSI Circuit Design*. Wiley Inc., 2000.

[3] P. K. Lala, *Digital Circuit Testing and Testability*. Academic Press, 1997.

[4] S. Chattopadhyay, "Reordering Test Pattern with Don't Cares for Minimizing Power Dissipation During Combinational Circuit Testing," in *Fourth International Conference on Innformation Technology*, pp. 260–264, December 2001.

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

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

[7] P. Banerjee, *Parallel Algorithms for VLSI Computer Aided Design*. Prentice Hall International, 1994.

[8] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Prentice Hall International, 1990.

[9] D. Johnson and L. McGeoch, *The Travelling Salesman Problem : A Case Study in Local Optimization – in Local Search in Combinational Optimization*. E.H.L Arts and J.K. Lenstra (eds), John Wiley and Sons, 1996.

[10] S. Hurst, *VLSI Custom Micro Electronics*. Marcel Dekker AG, 1999.

[11] T. Mitchell, *Machine Learning*. McGraw Hill Inc., 1997.

[12] X. Z. amd K. Roy and S. Bhawmik, "A Tool for Energy Conscious Weighted Random Pattern Testing," in *12th International Conference on VLSI Design*, pp. 416–422, January 1999.

Table 1: Benchmark results

Circuit	Atalanta				Genetic algorithm based method					
	fault cov.	No. of transitions			fault cov.	No. of transition	%red. in fault cov.	% red. in transitions over Atalanta		CPU time (in min.)
		Unordered	Ordered with TSP	% red. in TSP				unordered	ordered	
c5315	98.9	114707	103058	10.15	96.89	36519	2.01	68.16	64.56	6.85
c1355	99.5	17688	15169	14.2	96.57	5152	2.93	70.87	66.03	5.91
c1908	94.9	42018	32918	21.6	95.52	9129	-0.62	78.27	72.26	5.91
c2670	95.7	54368	43374	20.2	93.77	15357	1.93	71.75	64.59	6.09
c3540	78.2	88171	63600	27.8	81.71	38642	-3.51	56.17	39.24	5.84
c432	99.3	3894	2911	25.2	97.33	1130	1.97	70.9	61.18	4.81
c499	98.9	5218	4264	18.3	95.51	1386	3.39	73.4	67.49	5.26
c6288	99.6	30190	28288	6.3	99.41	8651	0.39	71.34	69.41	6.07
c880	99.8	9297	7657	17.6	98.51	2835	1.29	69.51	62.97	4.90
c7552	98.3	304143	234536	22.9	96.66	84293	1.64	72.28	63.18	11.11
			Average	19.02			1.14	70.26	63.18	
Population size = 1000, Number of generation = 1000										

Table 2: Comparison with earlier work

Circuit	% reduction using [5] over Atalanta	% reduction using our approach over Atalanta	Difference
c5315	51.5	68.16	16.66
c1355	68.6	70.87	2.27
c1908	43.4	78.27	34.87
c2670	45.6	71.75	26.15
c3540	20.1	56.17	36.07
c432	51.3	70.9	19.6
c499	7.8	73.4	65.6
c6288	56.3	71.34	15.04
c880	49.6	69.51	19.91
c7552	67.8	72.28	4.48
Average	46.2	70.26	24.06