Power Aware Testing by Proper Don't Care Filling of Test Patterns

Oindrila Chakraborty, Parna Chakraborty, Priyanka Choudhury and Sambhu Nath Pradhan
Department of ECE, NIT Agartala
Jirania,Agartala
Pin-799055, Tripura, India.

ABSTRACT
With the advancement in automation, the importance of periodic testing of electronic circuits during their lifetime is increasing day by day. Generally, a circuit or system consumes more power in test mode than in normal mode. This extra power consumption can give rise to severe hazards in circuit reliability or, in some cases, can provoke instant circuit damage. Thus it is necessary to reduce the power consumption during test mode. This power is proportional to the number of node switching due to the feeding of successive test patterns. Test patterns generated by ATALANTA with –D option, contains don't cares. Efficient filling of don't cares in the patterns may reduce the number of switching when they are applied in succession. In this paper we have presented an approach based on Genetic Algorithm (GA) for don't care filling of the test patterns generated by Atalanta to reduce switching during circuit testing without compromising fault coverage. The proposed GA based formulation can save upto 58% power compared to existing approach. A trade-off between fault coverage and transitions also has been presented in this paper.

General Terms
Low Power VLSI Testing.

Keywords
Testing, Low Power, Genetic Algorithm, Don't care, Fault coverage, Switching.

1. INTRODUCTION
Modern VLSI devices such as computers and electronics devices has become complex because of the decreased dimensions, referred to as feature size of transistors and interconnecting wires from tens of microns to tens of nanometers. The reduction in feature size has increase the implementation of millions of transistors and also resulted in increased operating frequencies and clock speeds. However the reduction in feature size increases the probability that a manufacturing defect in the IC will result in a faulty chip. A very small defect can easily result in a faulty transistor or interconnecting wire when the feature size is less than 100 nm. Furthermore, one faulty transistor or a single wire can make the entire chip fail to function properly or at the required operating frequency. Yet, defects generated during the manufacturing process are unavoidable, and, thus, some number of ICs is expected to be faulty. Therefore, testing is required to guarantee fault free products, regardless of whether the product is a VLSI device or an electronic system composed of many VLSI devices.

In this work we have targeted single stack-at faults of the device. A stuck-at fault transforms the correct value on the faulty signal line to appear to be stuck at a constant logic value, either logic 0 or logic 1, referred to as *stuck-at-0* or *stuck-at-1* respectively.

Test currently ranks among the most expensive and problematic aspects in a circuit design cycle, revealing the ceaseless need for test-related innovative solutions. As a result, several techniques have been developed [1], both for enhancing the testability of a design through Design-for-Testability (DFT) modifications and for improving the test generation and application process. Traditionally, these techniques are evaluated according to a number of parameters: the area overhead, the fault coverage achieved, the test application time, the test development effort, etc. The recent development of complex, high-performance, low-power devices implemented in deep submicron technologies creates a new class of more sophisticated electronic products, such as laptop computers, cellular telephones, audio and video-based multimedia products, energy efficient desktop computers. This new class of systems makes power management a critical parameter that cannot be ignored during test development as the power and energy of a digital system are considerably higher in test mode than in system mode [2-4]. The reason is that test patterns cause as many nodes switching as possible while a power saving system mode only activates a few modules at the same time. Another reason is that successive functional input vectors applied to a given circuit during system mode have a significant correlation, while the correlation between consecutive test patterns can be very low [5]. To meet specified power limits during test and avoid system destruction, it is really important to reduce power dissipation during testing.

Some approaches have been reported in the literature [5-6] 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. The initial set of patterns is generated by ATPGs like ATALANTA [7] etc.

In [8], the authors proposed a 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, 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.

Example 1: All the test patterns generated by ATALANTA are not required for examining the fault coverage of a circuit. The power dissipation during testing [10] is minimized by reducing the number of transition in the circuit. Usually test vectors are in random and hence it is necessary to efficiently fill don't cares and rearrange the order of occurrence of test vectors so that the switching activity between successive test vectors is minimum. This can be explained with an example.

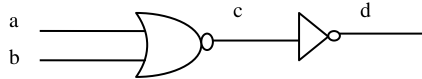

Fig 1: An example circuit

Suppose the test pattern generated for the above circuit are ‘x1’ and ‘1x’, where ‘x’ denotes don’t care. Since there is only one x in each patterns therefore the don’t cares can be filled in two ways. Let, after filling don’t cares the test patterns are ‘01’and ‘11’. If the sequence of test application is ‘1x’ and then ‘x1’, the number of transitions is 4 but the number of transitions would be 5 for the filling of don’t cares such that the patterns are ‘11’ and ‘10’. By this example it is evident that the number of transitions depends on the way of filling don’t cares. For filling two don’t cares in two patters there are four choices. So, for a big circuit having large inputs, number of test patterns and don’t cares are very high. All possible filling and finding the best patterns giving lowest transitions is hard problem. To solve this problem we have formulated a Genetic Algorithm (GA) based heuristic which gives the proper way of filling don’t cares such that dynamic power dissipation during testing is reduced by minimizing the node transitions without compromising the fault coverage.
The rest of the paper is organized as follows. Proposed GA-based technique for don’t care filling is presented in Section 2. Section 3 depicts the experimental results and conclusion is given in Section 4.

2. PROPOSED GA-BASED TECHNIQUE FOR DON’T CARE FILLING

Don’t cares, present in the test patterns are filled using Genetic Algorithm (GA). GA is search algorithm based on the mechanics of the natural selection process (biological evolution). The most basic concept of GA is that the strong tend to adapt and survive while the weak tend to die out. GA’s have the ability to create an initial population of feasible solutions, and then recombine them in a way to guide their search to only the most promising areas of the solution space. Each feasible solution is encoded as a chromosome and each chromosome is given a measure of fitness via a fitness (evaluation or objective) function. The fitness of a chromosome determines its ability to survive and produce offspring. A finite number of chromosomes, having different fitness values are called population. This size of the population is maintained throughout all the generations.
Chromosome structure-The chromosome is a string of ‘0’ and ‘1’ which corresponds to don’t care bit present in the pattern.

Generation of initial population-with an example let us illustrate how initial populations are created.

Example 2: The test patterns obtained after running ATALANTA with –D 1 option for c17.bench circuit are ‘100xx11’, ‘00xxx1x’, ‘xx101xx’, ‘0 x1xx11’, ‘1xx10111’, ‘00x1x10’ and ‘001xx01’. Here, ‘x’ represents don’t care. As the number of don’t cares present in the set of test pattern is 19, the size of the chromosome is 19. Then, each bit of the chromosome is filled by ‘1’ or ‘0’ which are generated randomly. Let the chromosome be “010100011101000000”. The first bit i.e. ‘0’ replaces the first don’t care of the first test pattern. Similarly the next bit of the chromosome replaces the next don’t care bit of the test pattern and this goes on until it reaches the last bit of chromosomes. So, the test patterns generated are 1000111, 0001010 etc.

2.1 Operators of GA

Operators are used to generate the populations for the next generation. The operators in GA are selection, crossover and mutation.
Selection - selection is usually the first operator applied on population. From the population, the chromosomes are selected based on fitness value for to crossover, mutation and to produce offspring.
Crossover - Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to the user definable crossover probability. In this work we have used two point crossovers [5].
Two point crossover operators randomly select two crossover points within chromosomes. Between these two points part of chromosome is exchanged to produce two new offspring.
Consider the following crossover operation as below.

Parent 1: ‘11011|0010011|0110’ and Parent 2: ‘11011|1100001|1110’. Interchanging the parent chromosomes across the crossover points-The offspring produced are: Offspring 1: ‘11011|1100001|0110’ and Offspring 2: ‘11011|0010011|1110’.

Mutation- Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes to the next. Mutation alters one or more gene values in a chromosome from its initial state. This can result in entire new gene values being added to the gene pool. With the new gene, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is intended to prevent the search falling into the local optimum of the state space. For mutation to occurs we have randomly generated two numbers between 1 and N (N is the size of the chromosome). All the digits within these two numbers are flipped.

2.2 Cost Function

In this section we formulate the cost function used to measure the fitness of chromosomes in a population. Fitness function quantifies the optimality of the solution (chromosomes) so that particular chromosomes may be ranked against all the other solutions. The function depicts the closeness of a given ‘solution’ to the desired result. As, we are targeting to maximize fault coverage and minimize transition, the cost function contains weighted sum of both the parameters. Let, *max_fault* and *max_tran* are the maximum fault coverage and maximum transitions among all the chromosomes in the initial population respectively. For *i*-th chromosome, let FC_i be the fault coverage and TR_i be the total number of transitions. Cost of the *i*-th chromosome is then formulated as,

$$Cost = w_1 \times (FC_i / max_fault) + w_2 \times (1 - (TR_i/max_tran))$$

Where, w_1 and w_2 are two empirically determined constants such that $w_1 + w_2 = 1$.

Complete flow of the proposed technique is depicted in Fig. 2.

After filling the bit values of chromosome of example 2 in the test patterns, patterns no. 2, 3 and 4, 6 are same. These repeated test patterns are removed and fault coverage and transitions are calculated. The steps that are followed in this method are as follows.
Step1: Input: test patterns, weight (w_1, w_2).

Step2: Output: fault coverage and transition count.

Step3: Create chromosomes whose size equal to number of don't care in the set of test vectors.

Step4: Random function generates number (1 & 0) for chromosomes generated in step 3.

Step5: Replace the don't cares of test patterns and filter the repetition.

Step6: Calculate fault coverage and transition for the filtered test vectors.

Step7: Calculate the cost function ($Cost = w_1 \times (FC_i / max_fault) + w_2 \times (1 - (TR_i/max_tran))$).

Step8: Sort the cost and the chromosome of don't care in ascending order.

Step9: Undergo direct copy, selection, mutation on all the chromosomes.

Step10: repeat from step5 to step9 on new generated chromosomes until the cost function doesn't change anymore.

Step11: when cost function doesn't change for 20 consecutive generations, terminate the generation.

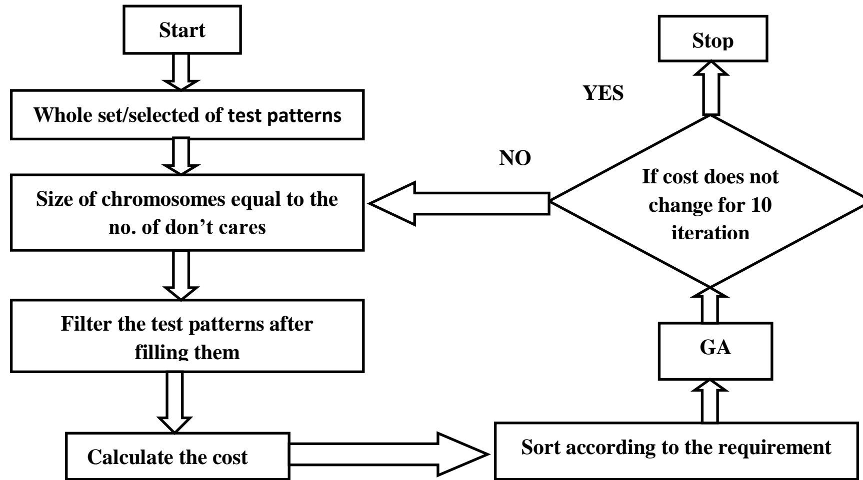

Fig 2: Flow chart of the proposed GA-based don't care filling

Table 1: Fault coverage and transition for different weights

Weight	**w_1=0,w_2=1**		**w_1=0.25,w_2=0.75**		**w_1=0.5,w_2=0.5**		**w_1=0.75,w_2=0.25**		**w_1=1,w_2=0**		**Max Gen**	**H4 [9]**	
Circuit	**FC**	**Trans**	**FC**	**Trans**	**FC**	**Trans**	**FC**	**Trans**	**FC**	**Trans**		**FC**	**Trans**
C17	100	8	100	10	100	16	100	20	100	64	58	100	44
C432	92.9	8427	98.2	8601	99.7	8887	99.8	8765	99.7	23070	299	93.89	22215
C499	98.9	10392	98.9	11034	98.9	10766	98.9	10744	98.9	19777	361	97.83	6756
C880	98.3	9936	99.8	9845	99.3	9919	99.9	10184	99.8	19737	300	63.96	7451
C1355	98.7	10379	98.8	12527	99.3	11025	99.7	11947	99.7	11834	498	61.28	10789
C1908	98.9	28709	99.9	29123	98.9	30714	99.9	30836	99.4	20920	595	83.97	16963
C6288	97.9	626947	98.9	692544	98.8	647547	98.9	632588	97.9	903063	680	97.97	2555792
Avg FC/trans wrt only w_1=	0.986	1.000	0.998	1.093	0.999	1.179	1.0023	1.262	1.000	2.562		0.862	2.177

3. RESULTS

In a circuit dynamic power consumption will be there if there is node switching for a particular technology. So, in our experiments we have considered dynamic power as the number of transition (0 to 1& from 1 to 0). We have performed experiments on iscas85 benchmark circuits. In the experiment, the whole redundant set of test patterns (generated using "– D 1" option of ATALANTA) is taken. For different values of w_1 and w_2, FC and transitions after running our GA-based algorithm are tabulated in table 1.

Table 1 shows fault coverage and transition count at different values of w_1 and w_2. Column, 'FC' and 'Trans' are the fault coverage and number of transitions before termination respectively for different values of weights. Column 'Max Gen' is the maximum number of generation among all the weighted solutions. From table 1 it can be observed that fault coverage obtained for different circuits are approximately high and same. It can be observed also that when w_1=1 and w_2=0, the cost function depends only on fault coverage, not on transition count. If the values of the fault coverage at (w_1=1, w_2=0) is taken as unity then the faults coverage at different values of w_1 and w_2 with respect to this is shown in the last row of the above table 1. Similarly cost function is dependent on transition count only when w_1=0 and w_2=1. If the values of the transition is at w_1=0, w_2=1 is taken as unity then the transition count at different values of $w_1$and w_2 with respect to the transition at w_1=0 and w_2=1 is shown in the above table. It can be observed that optimized fault coverage and transition count obtained at w_1=w_2=0.5. The last row of table 1 shows the average of fault coverage and transition for different values of weight with respect to fault coverage (at w_1=1, w_2=0) and transition (w_1=0, w_2=1) respectively. For different weighted combinations fault coverage almost remain same but the number of transitions reduces as the weight associated with transition increases. Last two columns are the fault coverage and transition count after filling don't care with H4 heuristic [9]. This fault coverage is 14% lower and transition is 117% higher. This shows the effectiveness of our approach in terms of high fault coverage and reduced power (transition).

4. CONCLUSION

In this paper we have presented a GA based approach to fill don't cares present in test patterns such that test power is reduced by reducing transition without compromising fault coverage. Fault coverage is more than 99% and reduction in test power is more than 58% compared to previous work H4 [9].

5. ACKNOWLEDGMENTS

This work was supported by RPS project (Ref. No.: 8023/RID/RPS-24/(NER)2011-12) sponsored by All India Council of Technical Education, New Delhi – 110 001.

6. REFERENCES

[1] Patrick Girard "Survey of low power testing of VLSI circuits", IEEE design and test of computers, may-june2002

[2] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," IEEE VLSI Test Symp., pp. 4-9, April 1993.

[3] W.H. Debany, "Quiescent Scan Design for Testing Digital Logic Circuits," Dual-Use Tech. & App., pp. 142-I 5 I , 1994.

[4] J. Rajski and J. Tyszer, "Arithmetic Built-In Self-Test for Embedded Systems," Prentice Hall PTR, 1998.

[5] S. Wang and S.K. Gupta, "DS-LFSR : A New BIST TPG for Low Heat Dissipation," IEEE Int. Test Conf., pp. 848-857, October 1997.

[6] F. Corno et al., "A Test Pattern Generation Methodology for Low Power Consumption," Proc. 16th VLSI Test Symp. (VTS 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp 453-459.

[7] 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.

[8] S. Chattopadhyay, N. Choudhary "Genetic Algorithm based Approach for Low Power Combinational Circuit Testing," Proceedings of the 16th International Conference on VLSI Design (VLSI'03).

[9] 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.

[10] M.Abromovici, M.A.Breuer and A.D. Friedman,"Digital System Testing and Testable Design" New York, Computer science press, 1990.