Fault Oriented Test Pattern Generation for Sequential Circuits Using Genetic Algorithms

Eero Ivask, Jaan Raik, Raimund Ubar
Department of Computer Engineering, Tallinn Technical University, Estonia

Abstract

Current paper presents a new technique of test pattern generation for sequential digital circuits based on using Genetic Algorithms (GA). Previously published approaches are targeting the entire set of undetected faults simultaneously, as opposed to concentrating on single faults, one-by-one. The method proposed below implements a two-pass algorithm. In the first pass random test sequences that quickly detect a number of easy-to-test faults are generated. During the second pass we switch to a fault oriented strategy where the remaining faults are considered separately by GA. The main motivation for selecting the fault oriented approach is better convergence of the genetic optimization process with the presence of a single target fault.

1: Introduction

Current paper presents a genetic algorithm based approach to test generation that differs from most of the previous works by specifically targeting single faults. Experiments show that the fault-oriented method is more effective in terms of achieved fault coverage than any other ATPG it was compared to. The paper is organized as follows. In Section 2 we explain the basic structure of the two-pass algorithm. Section 3 provides more detailed information about the genetic optimization implemented in the second pass. Section 4 presents experimental results, and finally, conclusions are given.

2: Basic structure of the algorithm

The algorithm performs two passes. In the first one, a set of random input sequences is generated and fault simulated in order to detect the easy-to-test faults. Subsequently, remaining faults are targeted one-by-one by a Genetic Algorithm (GA). The only assumption that is made about the circuit to be tested is the presence of a global reset signal. A rather wide sub-class of synchronous digital circuits falls into the cathegory.

During the first pass, random test sequences are generated as follows. First, n test sequences of l vectors are generated where we keep track of the global reset. At the beginning of each sequence the reset signal is activated and then kept in an inactive state up to the end of the sequence. Subsequently, m additional sequences are generated where the global reset is treated randomly. Our experience showed that combining these two strategies offered significantly higher fault coverage than chosing any one of them. In experiments presented below the values for l, n and m are 100, 100 and 10.

The second pass considers the remaining faults one-by-one. It implements a technique based on genetic optimization. Randomly activated fault effect is propagated onto primary outputs with help of genetically engineered test sequences new test vectors are constructed based on the best vector sequences developed progressively during previous iterations.

3: Fault oriented genetic optimization

In order to solve any problem, the following components are must in genetic algorithm [1]:

- chromosomal representation of solution to the problem,
- way to create an initial population of solutions,
- an evaluation (fitness) function in order to estimate the quality of the solution
- genetic operators that alter the structure of "children" during reproduction
- fine tuned parameters

Representation. In context of test generation for sequential circuits, sequence of test vectors will be an individual. Concurrent sequences form the population.

Initialization. Initially, a random set of test sequences is generated. Such an initial test sequence set is subsequently given to a simulator tool for evaluation.

Following steps of algorithm are carried out repeatedly.

Evaluation of test vectors. We use fault simulation in order to evaluate test sequences. Simulation is carried out only for particular fault under consideration. *Fitness* of the test sequence is calculated as following:

C_{a*} activated+ C_p *propagated,

Where 'activated' is number of clock cycles when the fault effect was activated in the circuit and 'propagated' is the number of clock cycles when fault effect was propagated onto some flip-flop. C_a and C_p are constants, which show how much weight is given to parameters. We selected 0.1 for C_a and 1 for C_p.

Fitness scaling. As a population converges on a solution, the difference between *fitness* values may become very small. Best solutions can not have significant advantage in reproductive selection. We use square values for test sequence' fitness values in order to differentiate good and bad individual.

Selection of candidate sequences. Selection is needed for finding two candidates for crossover. Based on fault simulation results better test sequences are selected. Roulette wheel selection mechanism was used here.

Crossover. Swapping genetic material of the two parents allows useful genes (relevant bits) to be combined in their offspring (new test sequence). Most successful parents reproduce more often. Beneficial properties of two parents combine. Crossover and selection (fitness function) are the keys to genetic algorithm's power. In current work, one-point horizontal and one-point vertical crossover (Fig. 1) were implemented, however at present only experimental results for vertical crossover are available.

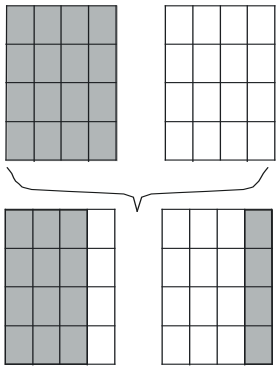

Figure 1. Vertical one point crossover

Mutation. Random mutation provides background variation and occasionally introduces beneficial genetic material. In all of the test vectors, every bit is inverted with a certain probability *p*. However, bit position corresponding to reset input is not altered. Using such knowledge-based technique helps to reduce the search space.

4: Experimental results

In all experiments following parameters for GA were used: population size 16 sequences, sequence length 200 vectors, maximum number of generations 200, crossover type vertical one point, mutation rate 0.01, elitist selection.

Table 1 shows comparison of test generation results of four ATPG tools. These are, GATEST [2] and HITEC [3], and GA (based on the approach described in current paper), respectively. The experiments for the first two tools were run on a 300 MHz SUN UltraSPARC 10 computer. Test for GA were generated on a 366 MHz SUN UltraSPARC 60 which is a slightly faster machine. On the other hand, GATEST and HITEC used internally the parallel fault simulator PROOFS, which is in average about 4 times faster than the serial simulation implemented in GA. Thus, the comparison of run times in Table 1 is not exactly precise. However, it gives an idea of the speed differences between various algorithms. In order to perform straight comparison of the test quality, actual stuck-at fault coverages of the test patterns generated by all the four tools were evaluated by single fault simulator.

Table 1. Test generation results

circuit	HITEC [3]		GATEST [2]		GA	
	cov., %	time, s	cov., %	time, s	cov., %	time, s
gcd	89.3	196	92.2	90	**93.0**	702
mult	63.5	2487	77.3	3027	**80.5**	19886
diffeq	95.1	> 4 h	96.0	4280	**97.9**	53540
huff	12.5	16200	27.6	3553	**52.8**	> 10h

References

[1] J.H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975

[2] E. M. Rudnick, et al., "Sequential circuit test generation in a genetic algorithm framework", *Proc. DAC Conf.*, 1994.

[3] T.M. Niermann, J.H. Patel, "HITEC: A test generation package for sequential circuits", *EDAC*, 1991.

[4] J.Raik, R.Ubar. "Sequential Circuit Test Generation Using Decision Diagram Models", *DATE*, 1999.

[5] Goldberg "Genetic algorithms", Addison-Wesley USA, 1991