International Journal of Computer and Communication Technology

Volume 3 | Issue 4 Article 7

October 2012

ATPG for Faults Analysis in VLSI Circuits Using Immune Genetic Algorithm

P. K. Chakrabarty
Professor, Department of CSE., IT,BHU, India, parthak@yahoo.com

S. N. Patnaik
Asst.Professor, ECE Department, DRIEMS, Cuttack, India, patnaiksn@gmail.com

ATPG for Faults Analysis in VLSI Circuits Using Immune Genetic Algorithm

P.K.Chakrabarty[1], S.N.Patnaik [2]
1Professor, Department of CSE., IT,BHU, India
2Asst.Professor, ECE Department, DRIEMS, Cuttack, India
[1]parthak@yahoo.com [2]patnaiksn@gmail.com

Abstract : As design trends move toward nanometer technology, new Automatic Test Pattern Generation (ATPG)problems are merging. During design validation, the effect of crosstalk on reliability and performance cannot be ignored. So new ATPG Techniques has to be developed for testing crosstalk faults which affect the timing behaviour of circuits. In this paper, we present a Genetic Algorithm (GA) based test generation for crosstalk induced delay faults in VLSI circuits. The GA produces reduced test set which contains as few as possible test vector pairs, which detect as many as possible crosstalk delay faults. It uses a crosstalk delay fault simulator which computes the fitness of each test sequence. Tests are generated for ISCAS'85 and scan version of ISCAS'89 benchmark circuits. Experimental results demonstrate that GA gives higher fault coverage and compact test vectors for most of the benchmark circuits.

1. Introduction

As a consequence of technological advances which have resulted in an increase of VLSI chip density, increased number of interconnect layers and in an improvement of timing performances, the test for static stuck-at faults only has turned out to be insufficient, and it is now also required to deal with physical defects which affect the timing behavior of a given circuit. Various noise sources such as crosstalk and power supply noise has a significant impact on the timing performance of deep submicron design (DSM) designs. The increasing number of transistors in the chip leads to more devices switching simultaneously resulting in power supply noise which reduces device voltage levels and increases signal delay. Interconnection lines which were assumed to manufacturing testing. In this paper, we present a GA based test generation algorithm for crosstalk delay faults. The GA generates candidate test vectors and the crosstalk delay fault simulator computes the fitness of candidate test vectors. The remainder of the paper is organized as follows. Section 2 discusses prior work. Section 3 describes the algorithm for finding a reducing list of target faults, Section 4 gives a brief description of crosstalk delay fault simulator, Section 5 describes the features of GA, Section 6 describes test generation in GA framework and Section 7 presents the experimental results for ISCAS'85 and scan version of ISCAS' 89 benchmark circuits. The paper is concluded in Section 8.

2. Static timing analysis

The number of crosstalk faults between all possible combinations of A-lines and V-lines are very large and impractical to detect for large complex circuits. Some of the faults cannot be tested or need not be be electrically isolated can now interfere with each other leading to functional problems. One such interaction caused by parasitic coupling between wires is known as crosstalk. These noise effects can cause completely validated chip to malfunction and lead to performance degradation of deep submicron design. There are two main types of cross talk effects: cross talk induced pulses and cross talk induced delay. The type of cross talk effect dealt in this paper is cross talk induced delay. Cross talk delay is induced when two lines, an aggressor line (A-line) and victim line (V-line) have simultaneous or near simultaneous transitions, which may cause undesirable effects including glitches, increase or decrease in the signal delay [1]. If both the lines transit in the same direction, the effective delay is reduced leading to crosstalk speedup. If aggressor and victim transit in the opposite direction then there will be an increase in delay leading to crosstalk slowdown. The designer has two options to eliminate errors caused by crosstalk either by resizing drivers, rerouting signals, shielding interconnect lines and other such redesign techniques or to develop techniques to generate tests for crosstalk.

The latter option is often taken by designers as redesign may be very expensive. Moreover test generation for crosstalk also enables more aggressive design and enables more comprehensive post-

tested in a circuit. Hence reduced set of crosstalk delay faults are derived by static timing analysis of the circuit. The number of critical paths and the lines that lie on the critical path are calculated using the topological and timing information. Then the lists of target faults are obtained which are smaller than the set of all possible combinations of faults. The algorithm for calculating the reduced target fault list we calculated :

1. The latest transition time and the earliest transition time for each line are calculated.

   2. From the maximum value of the latest transition time the longest path (critical path) is found. The lines in the longest path are found. These form the set of victim lines.

2. The timing window of the selected victim line is ompared with aggressor line and if the windows overlap than that selected V- and A- line pair is added to the target fault list. The input fault list to the simulator is the reduced set of target faults.

3. Crosstalk delay fault simulator

   9. Read the next vector and repeat steps 2 to 8.

   Given a test vector sequence as inputs, the objective of the fault simulator is to determine which of these faults are detected. Faults Detected are those that cause a logic error that is, a glitch at the POs. The delay effect at the primary output is also noted. The basic simulator is time wheel based event driven simulator. It uses three valued logic.The fault simulator is capable of detecting n-aggressor/single victim faults.

   Fault simulation based ATPG Algorithm.
   1. Simulate the good circuit for random vectors.
   2. Read the fault from the fault list.
   3. Simulate the good and bad circuits for the fault for each vector.
   4.The victim and aggressors should have opposite transition. Then fault is activated.
   5. Inject the fault by delaying the victim alone by a time step equal to one unit delay. Other events are scheduled as usual as for a good circuit. Continue simulation until the end of timeframe.
   6. Compare the values at all POs and sequential inputs
   with the good circuit.
   7. If the good and bad circuits differ, then fault is detected and the victim and activated aggressors pairs are removed from the fault list.
   8. Repeat from step2 for all the faults in the fault list.

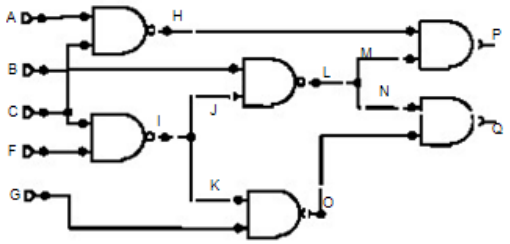

Figure 1. Example circuit

In Figure1, it was found from the static timing analysis that the example circuit consist of six longest paths namely:[C,I,L,P],[F,I,L,P], [C,I,L,Q],[C,I,O,Q],[F,I,L,Q] and [F,I,O,Q]. The victim set is [C, F, I, L, O, P, Q].The total number of target crosstalk faults for the circuit is 42. Two patterns (00001, 01000) are applied at the primary inputs of the logic circuit. The set of aggressors /victim taken for consideration are [H, I, O, P, Q/L]. Aggressors O, P are activated. Hence the victim L is given a one unit delay in the logic cone. A glitch is observed on Q and delay effect is observed on P. The activated aggressors are removed from the fault list. The other aggressors are considered for the next test pattern. The other faults are read from the fault list and fault simulation is done. The whole process is repeated for all the test vectors in the test set.

4. Overview of Genetic Algorithms

5. In the area of VLSI test, GA has been successfully used in stuck-at fault test and gate delay fault test. The simplicity, robustness, efficiency and effectiveness of GA make them a promising tool for complex applications. GA maintains a population pool of candidate solutions called strings or chromosomes. Each string is associated with a fitness value determined by a user defined fitness function. Figure 2 shows the flowchart of GA. GA starts with an initial population typically generated randomly and the evolutionary process of reproduction, crossover and mutation are used to generate an entirely new population from the existing population. The new population and the existing population compete for membership in the generation's membership pool. Selection of the chromosomes for new population is governed by the replacement strategy. The old population is discarded. The

sequence of selection, crossover mutation completes one-generation cycle. GA progresses through generations until the goal is reached such as fixed number of generations.

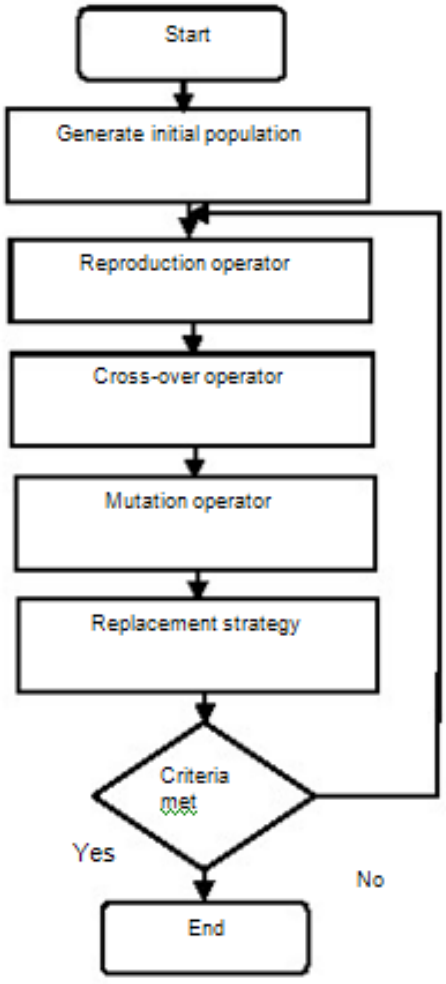

Figure 2. Flow chart of genetic algorithm

6. GA based test generation algorithm.

The goal of crosstalk delay fault test generation is generating a reduced test set, which contains as few as possible test vectors as possible. This is essentially a process exploring the space of test vector pair. So GA can be utilized to optimize the process of exploring. Every test vector/sequence can be treated as an individual or string. Thus we can view the number of test vector/sequence pairs as a population. To assess every test vector/sequence pair in a population in any generation of evolution, the crosstalk delay fault simulator is suitable. The pseudo code for GA based ATPG algorithm presented in this paper is shown in Figure 3.The ATPG algorithm performs in two phases. In the first phase the initial population of test vectors and sequences are generated by pseudo random process. In the second phase the GA phase, the test vectors are evolved based on fitness function. The fitness function used is:

Fitness = NFi

Where NFi is the number of faults detected.

```
FL= {reduced set of crosstalk delay faults}
{
initial pop=phase I (FL);
  if (FL =NULL)
  break;
phase II (initial pop, FL);
\}
```

Figure3. Pseudo code of overall GA based ATPG algorithm

Phase1

In this phase the initial sequences composed of M vectors are generated based on pseudo random process. The generated sequences are fault simulated for the faults in the fault list. If the sequence detects fault that fault is removed from the fault list and the corresponding sequence is added into the solution set. If no faults are detected by the sequence, then the last sequence generated in the corresponding cycle is added to the set. This process is repeated for max_iter. The pseudo code of phase I is shown in Figure 4.

```
Function Phase I
initial pop (FL)
For (i=0; i<max_iter, i++)
{
initial pop=phase I(FL);
randomly generate sequences of length L;
for (each sequence)
 {if sequence detects faults in the fault list
  {
      add sequence to the test set;
      drop the faults detected by that sequence;
  }
 }
return (initial population);
}
```

Figure 4 Pseudo code of phase I of GA based ATPG algorithm

Phase II

The initial population of GA is composed of the

sequences generated in phase I. To generate a new population from the existing one, two individuals (parents) are selected and crossed to create two entirely individuals (child) and each child is mutated with some small mutation probability. The selection operator is rank based selection. In rank based selection, the solutions are sorted according to their fitness from the worst (rank1) to the best (rankN).Each member in the sorted list is assigned a fitness equal to the rank of the solution in the list. Thereafter the proportionate selection operator is applied with the ranked fitness value and better solutions are chosen. The two parents are crossed to create two entirely new individuals (i.e.) child and each child is mutated with some small mutation probability. The two new individuals are than placed in the new population and the process continues until the generation is entirely filled. The previous population is iscarded. Crossover used is one point crossover. A crossover probability of 1 and mutation probability of 0.01 is used in all circuits. The no_gen is assumed to be 8, to reduce the execution
time. During test generation pop_size of 16 is used. The pseudo code for phase II is shown in Figure 5 Function Phase II

```
{
      Initial pop from phase1;
for (l=0;l<no_gen;l++)
    {for (k=0; k<popsize;k++)
            {selecttwoindividualsfrom
      population;
            apply crossover with probability 1;
            apply mutation with probability 0.01;}
            compute fitness of the individuals;
for (each sequence)
      if (sequence detects the faults in the fault list
         { add sequence to the solution set;
            drop the faults detected by the
      sequence;
         }
    }
}
```

Figure 5 Pseudo code of phase II of GA based ATPG algorithm

7. Experimental results

The crosstalk delay fault simulator and genetic algorithm based test generator is implemented in MATLAB under the LINUX environment. The random and GA based ATPG is applied to ISCAS'85 combinational circuits and several scan version of ISCAS'89 sequential circuits. Table 1 gives the characteristics of ISCAS'85 combinational circuits and the scan version of ISCAS'89 sequential circuits. After the circuit name, the number of primary inputs (PIs), number of primary outputs (POs), number of gates, number of paths and the number of critical paths are given. The entire circuit paths are analyzed using the tree data structure. The total number of paths in the circuit is calculated using depth first search algorithm which employs recursive search procedure. Critical paths are paths whose delay is longer than a given percentage of the longest propagation delay in the circuit. The selection is done using static timing analysis and given gate delay. The gate delay for static timing analysis is assumed to be one time unit. Table2 and Table 3 shows the coupling fault coverage obtained for a few benchmark circuits. Fault effect at PO gives the number of faults for which delay effect due to the coupling fault was noticed at the primary output. Faults detected are those that cause a logic error at the primary output. As it not practical to test all faults for large circuits reduced set of target faults are calculated by selecting victims which lie in the critical path and for each fault the aggressor-victim timing window should overlap. The input fault list for each circuit consists of every combination of single aggressor/single victim pairs. Gates are assumed to have a unit delay and the crosstalk delay value to be injected is also assumed to be one time unit. The bold number in Table2 and Table 3 represents the maximum number of faults detected for each circuit for GA compared to random vectors. In Table2 for the combinational circuit's c449 and c880 95% of the coupling faults produced a delay at the primary output. For 13 of the 15 scan version of the sequential circuits 98%of the coupling faults produced a delay effect at the primary output. The sequential circuit s1196, s1238 and combinational circuit c432 with critical paths of 9, 30 and 2199 respectively had produced a delay effect for only 86%, 78.2%and 84.8% of the coupling faults respectively. This may be due to the fact that the basis for finding the critical paths and target fault reduction is static timing analysis which does not consider the false paths.

In Table3 the number faults detected using GA based PG were much greater than random vectors for 17 of the 19 benchmark circuits tested. The numbers of faults detected were about 70% for the combinational circuit's
c432 and c449. For c880 fault coverage of 48%was reported. The numbers of faults detected were 24% to 66% for 10 of scan version of the sequential benchmark circuits. However for some specific scan

version of sequential circuit with coupling faults from 4000 to 10000 the GA based test generator reported lesser fault coverage's of 12% to 18%. The low fault coverage is not
unusual because due to conflicting Boolean conditions there are many coupling faults that are impossible to sensitize or to propagate to the primary output. Moreover the unit delay injected might be insufficient to cause a logic error at the primary output. For most of the
benchmark circuits compact test vectors were obtained. The CPU execution time shown in Table2 and Table3 is the execution time for GA based test generation. It was slightly higher for larger circuits. The graph shown in Figure 6 gives the comparison chart showing the efficiency of genetic algorithm over random vectors for different benchmark circuits with respect to the number
of target faults detected.

Circuit	No. of PIs	No. of POs	No. of Gates	No. of total paths 11	No. of critical paths 6
c17	5	2	6		
c432	36	7	160	83926	2199
c499	41	32	202	9440	14
c880	60	26	383	8642	92
s27	7	4	10	28	6
s208	19	10	96	145	2
s208.1	18	9	104	142	1
s298	17	20	119	231	1
s344	24	26	160	355	1
s349	24	26	161	365	1
s386	13	13	159	207	10
s420.1	34	17	218	474	1
s510	25	13	211	369	1
s526	24	27	193	410	1
s820	23	24	289	492	11
s1196	32	32	529	3097	9
s1238	32	32	508	3558	30
s1488	14	25	653	962	1
s1494	14	25	647	976	1

Table 1: The ISCAS'85 AND ISCAS'89 Circuit characteristics

Table2: Fault Effect at Primary Output

Circuit	No. of victims	No. of target faults	Fault effect at PO		CPU Time (secs)
			Random 12	GA	
c17	7	42		42	0.20
c432	103	9327	6629	7918	125.35
c449	207	21879	17937	20806	22.7
c880	70	9279	5722	8922	24.68
s27	10	74	70	72	0.03
s208	18	743	559	726	1.00
s208.1	12	558	371	512	1.26
s298	10	537	532	533	2.31
s344	21	1190	1185	1189	7.53
s349	21	1197	1192	1196	7.56
s386	49	4195	4152	4189	2.45
s420.1	14	1276	927	1257	4.05
s510	13	1098	1091	1098	4.76
s526	10	891	729	886	4.21
s820	43	7738	7362	7728	11.61
s1196	55	10630	5791	9150	20.03
s1238	45	5822	2764	4556	20.36
s1488	18	4305	4302	4304	29.76
S1494	18	4283	4280	4282	45.03

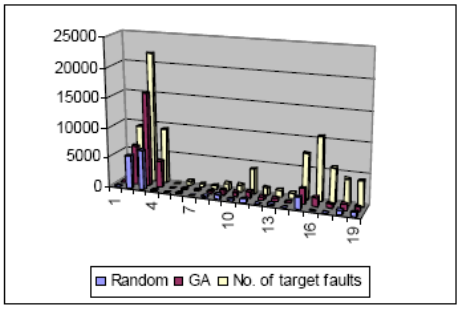

Figure 6 Comparison chart

7. Conclusion

The increased design density in deep submicron designs leads to more significant interference between the signals because of capacitive coupling or crosstalk which can produce oolean errors and delay faults. To guarantee design performance, ATPG techniques must consider how crosstalk affects

propagation delays. In this paper crosstalk delay fault simulator in a GA framework is developed. Redundant crosstalk faults which never affect the performance of the circuit are filtered out. The crosstalk delay fault simulator is capable of detecting n-aggressor/single victim faults. Results are presented for coupling faults which produces a delay effect at the primary output as well as those which produces a logic error at the primary output. The GA based ATPG tests both combinational and scan version of sequential circuits. The combinational and scan version of the sequential circuits produced delay effect for 72% to 99% of the coupling faults.

For coupling faults that produce a logic error at the primary output the method achieved 48% to 72% fault coverage on four combinational circuits and 12% to 66% fault coverage on 15 scan version of sequential circuits. Genetic algorithms are particularly suitable to parallel implementations, so better CPU execution time can be expected by a parallel immune GA based test generator.

8. References

[1] W.Y.Chen, S.K.Gupta, and M.A.Breuer, "Test Generation in VLSI Circuits for Crosstalk Noise", Proceedings of IEEE International Testconference, pages 641-650, October 1998.

[2] Rubio, N. Itazaki, X. Xu and K. Kinoshita, "An approach to the analysis and detection of crosstalk faults in digital VLSI circuits," IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, Vol.13, No.3, pp.387-394, March 1994.

[3] W.Y Chen, S. K. Gupta, and M. A. Breuer, "Analytic Models for Crosstalk Delay and Pulse Analysis under Non-Ideal Inputs", Proc. of the Int'l.Test Conf., pp. 809–818, Nov. 1997.

[4] W.Y.Chen, S.K.Gupta, and M.A.Breuer, "Test Generation for Crosstalk-Induced Delay in Integrated Circuits", Proceedings of IEEE International Test Conference, pages 191-200, October 1999.

[5] W.Y.Chen, S.K.Gupta, and M.A.Breuer, "Test Generation for Crosstalk Induced Faults: Framework and computational results" Journal of Electronic Testing: Theory and Applications, vol 18 pp17-28, Feb, 2002.

[6] Krstic, J.-J. Liou, Y.-M. Jiang and K.-T. Cheng, "Delay Testing Considering Crosstalk-Induced Effects", Proceedings of International Test Conference, 2001.

[7] X. Bai, S. Dey and A. Krstic, "HyAC: A Hybrid Structural SAT Based ATPG for Crosstalk," International Test Conference, 2003, pp. 112-121

[8] Aniket and R. Arunachalam, "A Novel Algorithm for Testing Crosstalk Induced Delay Faults in VLSI Circuits," Proceedings of International Conference on VLSI Design, pp. 479-484, 2005.

[9] H. Li and X. Li, "Selection of Crosstalk-induced Faults in Enhanced Delay Test", Journal of Electronic Testing: Theory and Applications, Vol. 21, No. 2, pp. 181-195, April 2005.

[10] Palit, A.K., Duganapalli, K.K., Anheier, W., "Test Pattern Generation for Crosstalk Fault in DSM chips using Modified PODEM,"TuZ, pp.41-45, 2008.

[11] Sunghoon Chun, Yongjoon Kim, Taejin Kim, Myung-Hoon Yang, Sungho Kang, "XPDFATPG: An Efficient Test Pattern Generation for Crosstalk-Induced Faults" 17th Asian Test Symposium, pp.83-88, 2008.

[12] Kunal P. Ganeshpure, Sandip Kundu "On ATPG for Multiple Aggressor Crosstalk Faults in Presence of Gate Delays" Proceedings of IEEE