Multi-Objective Genetic Algorithm for Testing MCNC FPGA Benchmark Circuits

Vinay Chopra[1], Dr. Amardeep Singh[2]

[1]Assistant Professor, Department of Computer Science & Engineering, DAVIET Jalandhar (Punjab), India

[2] Professor, Department of Computer Science & Engineering, Punjabi University Patiala (Punjab). India

[1]vinaychopra222@yahoo.co.in, [2]amardeep_dhiman@yahoo.com

Abstract— This work has shown that FPGA testing in the form of Boolean SAT, which is a NP complete problem, can be competently solved using the Multi-Objective Genetic Algorithm by first converting it into the Boolean SAT and representing it as a constraint satisfaction problem. This simulation work reads the Xilinx MCNC Benchmark circuits as BLIF word files and dynamically circuit graphs are created for above used circuits. Then these graphs are used for FPGA circuit testing using GA procedure and results have shown that as the problem size are increased by increasing number of variables and clauses, the fault coverage ratio increases. Also as the numbers of generations are increased, fault coverage ratio and CPU time to test the circuit for each FPGA circuit has improved.

Keywords— FPGA, CNF, MOGA, BLIF.

I. INTRODUCTION

FPGA testing or ATPG [1] is a major research topic which is used in semiconductor electrical testing in which the vectors or input patterns are required to check a device for faults that are automatically generated by a program. The Berkley Logic Interchange Format (BLIF) is used to represents FPGA Microelectronics Center of North Carolina (MCNC) circuits [26] in textual form in which all the circuits are inputted as a text file to the designed simulator. All experiments have been performed using MATLAB 7.0 and its Genetic Algorithm standard library for all of our practical work. In all cases the population size is fixed, the standard 1-point and 2-point crossover operator has been applied at a 50% rate, the mutation is 0.1% and selection is performed using the roulette wheel selection algorithm [25]. The algorithm scans the genes in a random order each gene is flipped, and the flipped is accepted if the fitness score is maximum. When all the genes have been tested the next generation is formed from the previous generation. There are some interesting issues concerning convergence to solution for formulating FPGA testing problem, as SAT instances as an optimization problem using the Multiobjective Genetic Algorithm. First of all, whenever a candidate solution evaluates to 1, we know that the solution has been found and search can be terminated. Conversely, there is strong motivation to continue the search until a solution is found. The difficulty, of course is that on any particular run there is no guarantee that a solution will be found in a reasonable amount of time due to the increasing homogeneity of the population as the search proceeds.

In order to obtain statistically significant results, several runs are required for different parameters on the SAT instances. The quality achieved is measured by the fault detection ratio which represents the ratio of number of clauses satisfied divided by total number of clauses. Second parameter, which is used to measure the computational cost of the Genetic Algorithm, is the time to evaluate the output. This measure does not depend upon the machine used and on the actual implementation.

II. THE DESIGNED GENETIC ALGORITHM FOR FPGA TESTING

The ATPG algorithm performs in two stages. In the first stage, the initial population of test vectors and sequences are generated by pseudo random process. In the second stage, the test vectors are evolved based on fitness function [74] as follows:

A. *Stage I of the Designed Algorithm*

In this stage the initial sequences consist 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 in the circuit then that fault is detached from the fault list and the subsequent sequence is added into the solution set otherwise the last sequence generated in the subsequent cycle is added to the set until max-iter is countered. The initial population of GA composed of the sequences generated in Stage I. Then two individuals (parents) are selected and crossed to create two entirely individuals (child) and each child is mutated with some small mutation probability to generate new sequences with some small mutation probability. The earlier population is discarded. The selection operator is rank based selection in which the solutions are sorted according to their fitness from the worst (rank 1) to the best (rank N). After that the proportionate selection operator is applied with the ranked fitness value and improved solutions are chosen. A crossover probability of 1 and mutation probability of 0.01 is used in all circuits. The maximum generation is varied from 1, 10, 20 and 30 to reduce the execution time. During test generation population, size of 1, 10, 20 and 30 is used. The fitness function used is:

```
Fitness = NFi
Where NFi is the number of faults detected.
{
FL= {total number of faults}
initial pop=phase I (FL);
if (FL =NULL)
break;
phase II (initial pop, FL);
}
```

Fig 1:Pseudo-code of overall GA based test pattern generation used.

B. *Stage II of the Designed Algorithm*

The initial population of GA [27] is composed of the sequences generated in Stage I. The entire process is repeated as given in Stage I for all the FPGA circuits' i.e.

a) Generate a new population from the sequences generated in Stage I using crossover operation.
b) Each child is mutated with some small mutation probability.
c) The solutions are sorted according to their fitness using rank based selection from the worst to the best.
d) Then the proportionate selection operator is applied with the ranked fitness value and better solutions are chosen.
e) 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 discarded.
f) A crossover probability of 1 and mutation probability of 0.01 is used in all circuits. The no_gen is varied to be 1, 10, 20 and 30 and test generation pop_size is also varied from 1, 10, 20 and 30 as in Stage I.

```
Function Phase II
{
Initial pop from phase1;
for (l=0;l<no_gen;l++)
{for (k=0; k<popsize;k++)
{select two individuals from
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;}}}
```

Fig 2: Pseudo-code of Phase II

III. FPGA TESTING RESULTS

Table 1, 2 & 3 give the complete testing solution with varying number of generations from 10, 20 and 30 respectively. The number of variables and clauses are sufficient to explain the order of magnitude of the FPGA testing problem. All simulation is being made by varying the number of iterations. Numbers of variables are fixed for each circuit as input. Two parameters are observed for different number of Generations firstly is Fault Coverage ratio and secondly is CPU time. The columns in the table record the following data: The numbers of Boolean variables correspond to the Testing function, number of clauses in the Testing function, Fault Coverage of the algorithm.

A. *Results by changing the No. Of Generations and keeping other parameters Same*

Table 1: Performance of FPGA Testing using Multi-objective GA with 1 Generation

circuits	variables	clauses	Clauses satisfied	Time	Fault coverage
asymmt	2604	32450	25700	0.12	79.2
alu2	3882	94088	79504	11.90	84.5
apex7	1893	15358	11441	0.12	74.5
c880	4623	72021	64242	1096	89.2
example2	3603	41023	34172	0.38	83.3
K2	13176	445254	410524	25.4	92.2
Too-large	3972	60432	51608	136.2	85.4
Vda	7436	168604	151574	44.5	89.9

In order to find out the any relation between these parameters experiment is being carried out with varying number of generations. As shown in [see table 1] the different circuits are being tested for fixed generation i.e. 1 in this case. For asymmt circuit we have total 2604 variables and 32450 numbers of clauses. Fault coverage ratio is calculated for 100 numbers of runs. It shows the total numbers of clauses satisfied are 25700 for asymmt out of total of 32450 clauses. So the fault coverage calculated is 79.2 %. Similarly for other circuits, fault coverage is calculated.

B. *Results by changing the No. Of Generations to 10 and keeping other parameters Same*

Table 2: Performance of FPGA Testing using Multi-objective GA with 10 Generations

circuits	variables	clauses	Clauses satisfied	Time	Fault coverage
asymmt	2604	32450	25750	0.10	79.4
alu2	3882	94088	79604	11.45	84.6
apex7	1893	15358	11641	0.11	75.8
c880	4623	72021	64942	1075	90.1
example2	3603	41023	34572	0.35	84.27
K2	13176	445254	411534	25.1	92.4
Too-large	3972	60432	52765	135.6	87.31
Vda	7436	168604	151883	43.5	90.08

Similarly the different circuits are being tested for fixed generation i.e. 10. For each circuit and the same number of variables the numbers of runs are fixed as 100. In this case fault coverage ratio improves as compared to the previous case [see table 2] for each circuit. Numbers of satisfied clauses for asymmt circuit are increased from 25700 to 25750 and similarly the results are improved for other circuits also. The time required for this calculation is 0.10 ms.

C. ***Results by changing the No. Of Generations to 20 and keeping other parameters Same***

Now generations are increased with gap of 10 to get the relation between fault coverage and generation's .Other parameters like number of runs, number of variables are kept same. Every time number of generations are increased fault coverage ratio improves. Numbers of satisfied clauses for asymmt circuit are increased from 25750 to 26550.The time required for this calculation is 0.08 ms.

Table 3: Performance of FPGA Testing using Multi-objective GA with 20 Generations

circuits	variables	clauses	Clauses satisfied	Time	Fault coverage
asymmt	2604	32450	26550	0.08	81.8
alu2	3882	94088	80503	10.25	85.6
apex7	1893	15358	12143	0.09	79.1
c880	4623	72021	67972	986	94.37
example2	3603	41023	35576	0.28	86.72
K2	13176	445254	414584	24.2	93.1
Too-large	3972	60432	53968	126.6	89.3
Vda	7436	168604	152783	41.5	90.61

D. ***Results by changing the No. Of Generations to 30 and keeping other parameters Same***

circuits	variables	clauses	Clauses satisfied	Time	Fault coverage
asymmt	2604	32450	28650	0.75	88.28
alu2	3882	94088	83413	9.45	88.65
apex7	1893	15358	13152	0.85	85.6
c880	4623	72021	68873	942	95.6
example2	3603	41023	38277	0.25	93.3
K2	13176	445254	425587	21.2	95.54
Too-large	3972	60432	55869	119.6	92.44
Vda	7436	168604	159789	38.5	94.77

Now generations are increased with gap of 10 to get the relation between fault coverage and generation's .Other parameters like number of runs, number of variables are kept same. Every time number of generations are increased fault coverage ratio improves. Numbers of satisfied clauses for asymmt circuit are increased from 26550 to 28650.The time required for this calculation is 0.75 ms.

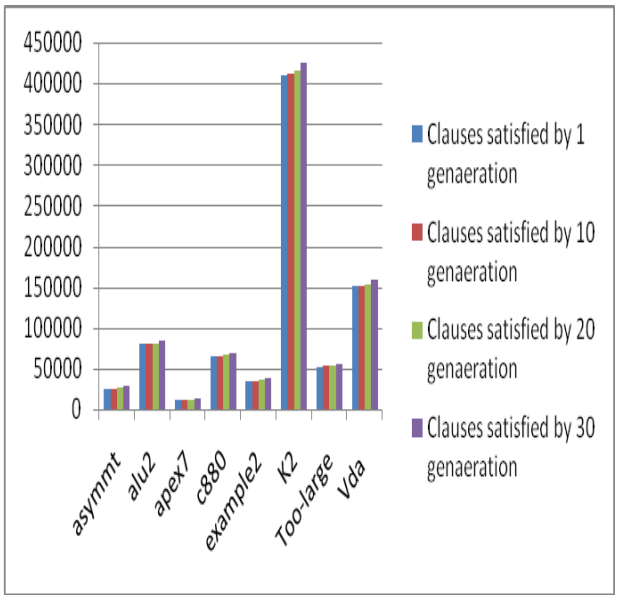

Fig 3: Comparison of satisfied clauses with variation in generations for various circuits

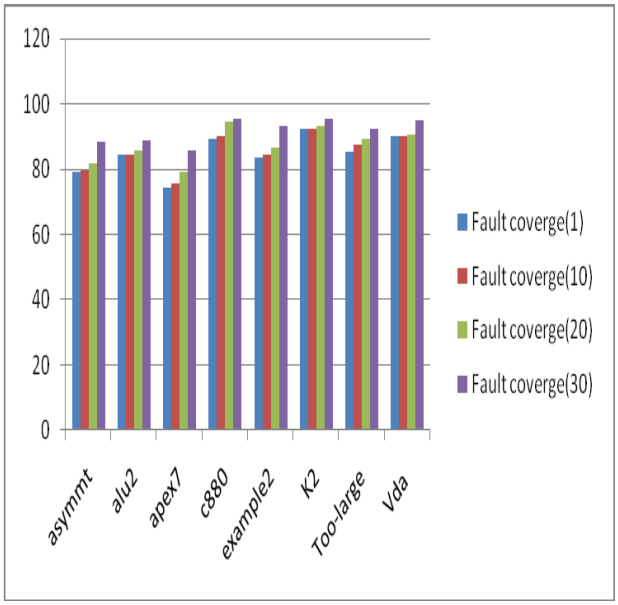

Fig 4: Comparison of fault coverage with variation in generations for various circuits

Graph shows the effect of increasing the number of generations with fault coverage ratio. Each generation takes a different value as input so every generation produces different value for fault coverage ratio. Fault coverage ratio increases linearly with number of generations.

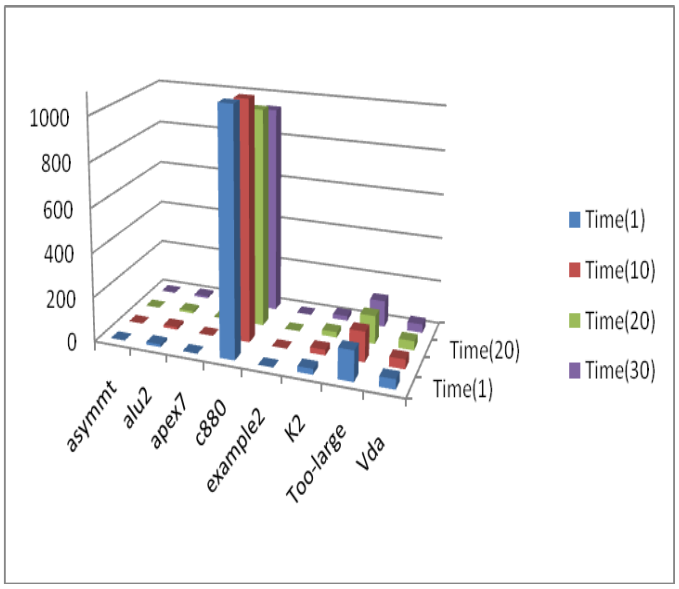

Fig 5: Comparison of time taken with variation in generations for various circuits

IV. CONCLUSION

In this paper Multi-Objective Genetic Algorithm has been designed for solving FPGA testing by first converting it into the Boolean SAT and representing it as a constraint satisfaction problem. This simulation work reads the BLIF files representing the Xilinx MCNC Benchmark circuits as word file and dynamically circuit graphs are created for above used circuits. Then these graphs are used for FPGA circuit testing using GA procedure and results have shown that as the problem size are increased by increasing number of variables and clauses, the fault coverage ratio increases. Also as the numbers of generations are increased, fault coverage ratio and CPU time to test the circuit for each FPGA circuit has improved.

V. REFERENCES

1. Görschwin Fey, Junhao Shi and Rolf Drechsler "Efficiency of Multi-Valued Encoding in SAT-based ATPG" 36th International Symposium on Multiple-Valued Logic Singapore. ISBN: 0-7695-2532-6, May 17-May 2006.
2. Jie Qin "A Brief Introduction to Application-Dependent FPGA Testing" Dept. of Electrical and Computer Engineering 200 Broun Hall, Auburn University, AL 36849-5201.
3. Junhao Shi, Görschwin Fey ,Rolf Drechsler, Andreas Glowatz , Jürgen Schlöffel and Friedrich Hapke "Experimental Studies on SAT-Based Test Pattern Generation for Industrial Circuits" 6th International Conference ASICON Vol.2, pp:970-972 ,24 Oct 2005.
4. Paolo Prinetto, Maurizio Rebaudengo, and Matteo Soriza "GATTO: A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits" IEEE Transactions on Computer Aided Design Of Integrated Circuits And Systems, Vol. 15, Issue No. 8, August, 1996.

5. Gregor Papa, Tomasz Garbolino ,Franc Novak and Andrzej H lawiczka " Deterministic Test Pattern Generator Design With Genetic Algorithm Approach" Journal of Electrical Engineering, Vol. 58, Issue No. 3, pp:121-127, 2007.
6. Jin-Kao Hao, Frédéric Lardeux and Frédéric Saubion "A Hybrid Genetic Algorithm for the Satisfiability Problem" LERIA, Bd Lavoisier, F-49045 Angers Cedex 01, 1998.
7. Tracy Larrabee, member IEEE "Test Pattern Generation using Boolean Satisfiability" IEEE Transactions on Computer-Aided Design, Vol. 11, Issue No. 1, January 1992.
8. J. Shi G. Fey R. Drechsler A. Glowatz F. Hapke J. Schlöffel "PASSAT: Efficient SAT-based Test Pattern Generation for Industrial Circuits" Proceedings of IEEE Computer Society Annual Symposium on VLSI, pp: 212-217, 11-12 May 2005.
9. V.D. Aggrawal "Test Generation for MOS Circuits Using D-Algorithm" 20th IEEE Conference on Design Automation pp: 64-70, 27-29 June 1983.
10. P. Goel, "PODEM-X: An Automatic Test Generation System for VLSI Logic Structures" 18th IEEE Conference on Design Automation pp: 260-268, 29 June 1981.
11. Fujiwara, H. "On the Acceleration of Test Generation Algorithms" IEEE Transactions on Computers Vol. C - 32, Issue No. 12, pp: 1137-1144, Dec. 1983.
12. Carlos A Coello. "A Comparative survey of Evolutionary based Multiobjective Optimization" International Journal of Knowledge and information Systems, Vol. 1, pp: 269-308, December 1999.
13. Carlos M. Fonsecay , Peter J. Flemingz, "An Overview of Evolutionary Algorithms in Multiobjective Optimization" International Journal of Evolutionary Computation, Vol. 3, Issue No.1, pp 1-16,April 7,1995.
14. Kalyanmoy Deb, Associate Member IEEE, Amrit Pratap, Sameer Agarwal, and T. Meyarivan "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II" IEEE Transaction on Evolutionary Computation, Vol.6, pp: 182-197, April 2002.
15. Haiming Lu, Gary G.Yen Member IEEE "Multiobjective Optimization Design using Genetic Algorithm" Proceedings of the IEEE International Conference on Control Applications pp:1190-1195 ,September 5-7 ,2001.
16. Ujjwal Maulik, Sanghamitra Bandyopadhyay, Anirban Mukhopadhyay "Multiobjective Genetic Algorithms for Clustering" book published by Springer Heidelberg Dordrecht London New York, ISBN 978-3-642-16614-3,e-ISBN 978-3-642-16615-0,2011.
17. Jin-Kao Hao, Frédéric Lardeux and Frédéric Saubion "A Hybrid Genetic Algorithm for the Satisfiability Problem" LERIA, Bd Lavoisier, F-49045 Angers Cedex 01, 1998.
18. Gi-Joon Nam and Karem A. Sakallah. "Detailed Routing of Complex FPGAs via Search-Based Boolean SAT" Symposium on Field Programmable Gate Arrays, Monterey, CA, pp: 167-175, December 2004.
19. L. Andrew C. "Field Programmable Gate Array Logic Synthesis using Boolean Satisfiability", M. Tech. Thesis submitted to Graduate Department of Electrical and Computer Engineering Department, University of Toronto 2005.
20. Fadi A. Aloul and Assim Sagahyroon "SAT-Based Techniques in Test Vectors Generation" International Journal of Advances in Information Technology, Vol. 1, Issue No. 4, November 2010.
21. Fahiem Bacchus, Toby Walash (Eds.) "Theory and Applications of Satisfiability Testing" 8th Springer International Conference, SAT June 19-23, 2005.
22. Y.L. Wu, S. Tsukiyama, and M. Marek-Sadowska, "Graph Based Analysis of 2-D FPGA Routing" IEEE Transactions on Computer-Aided Design, pp: 33-44, Jan. 1996.
23. Ujjwal Maulik, Sanghamitra Bandyopadhyay, Anirban Mukhopadhyay "Multiobjective Genetic Algorithms for Clustering" book published by Springer Heidelberg Dordrecht London New York, ISBN 978-3-642-16614-3,e-ISBN 978-3-642-16615-0,2011.
24. Haiming Lu, Gary G.Yen Member IEEE "Multiobjective Optimization Design using Genetic Algorithm" Proceedings of the IEEE International Conference on Control Applications pp:1190-1195 ,September 5-7 ,2001.
25. Rolf Drechsler, Stephan Eggersglub, Gorschwin Fey and Daniel Tille "SAT-based Automatic Test Pattern Generation" Dagstuhl Seminar Proceedings 08351, Evolutionary test Generation 2009.
26. Ian Kuon, Russell Tessier and Jonathan Rose "FPGA Architecture: Survey and Challenges" Foundations and Trends in Electronic Design Automation Vol. 2, No. 2 pp: 135–253, 2008.
27. S.Jayanthy, M.C.Bhuvaneswari "Simulation Based ATPG for Crosstalk Delay Faults in VLSI Circuits using Genetic Algorithm" ICGST- AIML journal, ISSN: 1687-4846, Vol. 9, Issue No.2, December 2009.