# SAT BASED AUTOMATIC TEST PATTERN GENERATION

P Partha Koundinya, Y Sai Krishna Reddy , K Rutwesh , V Mani Deepak

Department of Electronics and Communication Engineering, SRM University-AP

## ABSTRACT

The detection of a stuck-at fault in the digital circuit requires specific set of inputs, commonly known as test inputs. The generation of such test input patterns automatically is called automatic test pattern generation (ATPG). The fault detection problem is NP-complete. Therefore, with an increase in the size of the circuit and consequently the possibility of the number of faults, the generation of test inputs becomes computationally challenging. We have used the Boolean satisfiability (SAT) algorithm to deal with such complexity. Along with the inputs and outputs, the faults are assumed as additional variables, and the SAT equation was generated. The result of this SAT problem was obtained with *Minisat 2.2* algorithm. The results obtained were analyzed to determine the test pattern. The SAT-based approach has much less complexity than $2^n$ because of heuristic nature of the SAT solvers. The results are provided for combinational circuits with stuck-at faults. The method is extendible to the faults in sequential circuits in the future.

**Keywords**: Boolean Satisfiability, Conjunctive Normal Form, NP-Complete, Automatic test pattern generation (ATPG)

## INTRODUCTION

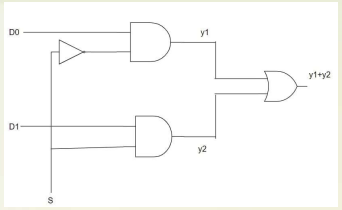
Different fault models exist in digital circuits, like the bridging fault model, transistor faults, stuck-at-fault model, and open fault model. Among these, the stuck-at-faults are more likely to occur, and thus their detection becomes essential [1]. In the current work, we confined ourselves only to the detection of stuck-at faults. A connection is shorted with either supply voltage or ground in a stuck-at fault, making it permanent 'logic 1' or 'logic 0'. Due to these faults, the behavior of the circuit may change. For detecting such faults, the inputs are required such that the effect of faults on the output is observable. Such inputs are called test inputs, and the sequence is called test pattern [1].

Automatic Test Pattern Generation (ATPG) is a method that is used to find the test pattern or test sequence. There are several methods available for the ATPG in combinational and sequential circuits [1]. Since the fault detection problem is NP-complete [2], Boolean satisfiability (or SAT) algorithms are one of the best-suited methods to find appropriate test sets [3]. The SAT algorithms determine if there exists at least one assignment such that the conjunctive normal form (CNF) expression holds TRUE. Some of the interesting SAT-based ATPG methods were given in [4-7]. Our work extends the SAT idea to generate ATPG to detect multiple simultaneous stuck-at faults with *Minisat* 2.2 [8] when the Boolean circuit description is known.

## METHODOLOGY

Methodology involves the following steps.

**1. Derivation of SAT expression and DIMACS file:** The SAT expression is usually represented in CNF or product of sum form. The SAT expression can be derived from the Boolean equivalence expression given by $p \leftrightarrow q \equiv (\bar{p} + q)(p + \bar{q})$. This equation always holds true for any binary value of $p$ and $q$. DIMACS file is a standard format which describes the Boolean circuit that can be given as an input to the SAT solver. For example, the SAT expression and DIMACS for 2 X 1 MUX is shown in Fig. 1.




| S | $D_0$ | $D_1$ | Y |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

```
c This is a DIMAC FILE
p cnf 4 5
-4  2  3  0
3  -4  -1  0
1  -4  2  0
1  4  -2  0
-1  4  -3  0
```

Fig. 1: Example to represent 2 X 1 fault-free MUX in a DIMACS format.

From the equivalence relation, we get, $Y \leftrightarrow Y_1 + Y_2$

$\rightarrow (\overline{Y} + (Y_1 + Y_2)).(Y + (\overline{Y_1 + Y_2})) \equiv 1$

$\rightarrow (\overline{Y}+D_0+D_1).(D_1+\overline{Y}+\overline{S}).(S+\overline{Y}+D_0).(S+Y+\overline{D}_0).(S+Y+\overline{D}_1) \equiv 1$

This expression is converted to DIMACS file as shown in Fig. 1

**2. Consideration of fault as a variable :** Assume that there is fault in a wire that connects $x$ with $y$. The fault has three states, viz. no-fault, stuck-at 0 (sa-0), and stuck-at 1 (sa-1). Therefore, it requires minimum two variables to represent these states, let us say $p$ and $f$, where $p$ indicates presence (existence) of the fault and $f$ indicates the type of the fault (sa-0 or sa-1). Therefore, we get output expression as $y = x\bar{p} + pf$. A SAT expression and DIMACS file can be derived along with these new variables as explained earlier. This type of modeling is also followed in [6].

**3. Algorithm to generate the test inputs :** After step 1 and step 2 the following algorithm is then implemented to obtain test inputs for different fault conditions.

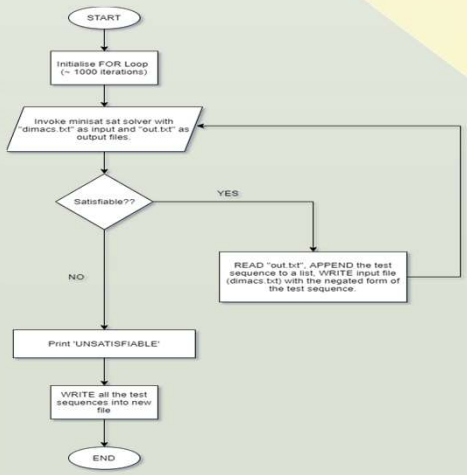
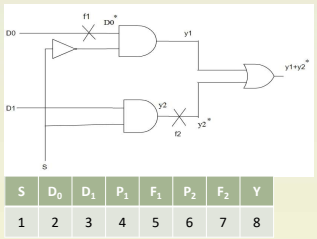



Fig. 2: Flowchart of the complete process to obtain the test pattern. The satisfiability is tested with *MiniSAT2.2* algorithm.

## MAIN RESULTS

The method explained above was implemented with python programming language. For the demonstration of our algorithm, we assumed two stuck-at faults, $f_1$ and $f_2$ in a 2:1 mux circuit and obtained the desired test patterns.




| S | $D_0$ | $D_1$ | $P_1$ | $F_1$ | $P_2$ | $F_2$ | Y |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

```
c This is a DIMACS FILE
p cnf 8 22
-8 6 3 2 5 0
-8 6 3 -4 5 0
-8 6 3 4 2 0
-8 6 1 2 5 0
-8 6 1 -4 5 0
-8 6 1 4 2 0
-8 7 3 2 5 0
-8 7 3 -4 5 0
-8 7 3 4 2 0
-8 7 1 2 5 0
-8 7 1 -4 5 0
-8 7 1 4 2 0
-8 7 -6 2 5 0
-8 7 -6 -4 5 0
-8 7 -6 4 2 0
-8 -1 6 3 0
-6 -1 7 3 0
-8 -1 7 -6 0
1 8 -2 4 0
1 8 -5 -4 0
-3 -1 6 8 0
8 -6 -7 0
```

Fig. 3: Example of 2 X 1 MUX with two faults and corresponding DIMACS format.

The algorithm described in Fig 2 was implemented on Fig. 3, and after analyzing the results, the following test patterns were obtained.

| Type of Faults | | Test Pattern |
|---|---|---|
| **Fault f1** | **Fault f2** | **(S, $D_0$, $D_1$ respectively)** |
| Sa-0 | Absent | 010,011 |
| Sa-1 | Absent | 001,000 |
| Absent | Sa-0 | 101,111 |
| Absent | Sa-1 | 000,001,100,110, |
| Sa-0 | Sa-0 | 010,011,101,111 |
| Sa-1 | Sa-0 | 000,001,101,111 |
| Sa-0 | Sa-1 | 000,001,100,110 |
| Sa-1 | Sa-1 | 000,001,100,110 |

## CONCLUSIONS

Satisfiability algorithms reduce the complexity of finding test patterns much below $2^n$. The SAT-based algorithm is capable of generating multiple test patterns simultaneously. It also addresses the critical issue of generating a test pattern for multiple simultaneous stuck-at faults. However, the problem of separating equivalent faults still persists. The method is implemented on combinational circuits, which we will try to extend for sequential circuits in our future work.

## KEY REFERENCES

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital systems testing and testable design," Design for Testability, 1990.
[2] E. Fornasini and M. E. Valcher, "Fault detection analysis of Boolean control networks," IEEE Transactions on Automatic Control, vol. 60, no. 10, pp. 2734-2739, 2015.
[3] S. A. Cook, "The complexity of theorem-proving procedures," in Proceedings of the third annual ACM symposium on Theory of computing. ACM, 1971, pp. 151-158.
[4] S J. Marques-Silva and L. Guerra e Silva, "Solving satisfiability in combinational circuits," in IEEE Design & Test of Computers, vol. 20, no. 4, pp. 16-21, July-Aug. 2003.
[5] P. Lin and S. P. Khatri, "Efficient cancer therapy using Boolean networks and Max-SAT-based ATPG," in IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS).
IEEE, 2011, pp. 87-90.
[6] Deshpande, Anuj and Layek, Ritwik, "Fault Detection and Therapeutic Intervention in Gene Regulatory Networks using SAT Solvers", Biosystems, Elsevier, Volume 179, Pages 55-62, May 2019.
[7] S. Eggersglub and R. Drechsler, "A Highly Fault-Efficient SAT-Based ATPG Flow," in IEEE Design & Test of Computers, vol. 29, no. 4, pp. 63-70, Aug. 2012.
[8] Eén N. and Sörensson N., "An Extensible SAT-solver", in Theory and Applications of Satisfiability Testing. SAT 2003.*LNCS*, vol 2919. Springer, Berlin, Heidelberg.

## ACKNOWLEDGEMENTS

We thank our advisor, Dr. Anuj Deshpande, for his invaluable guidance and support during this project.