Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 2 (2025)

Greedy Test Pattern Generation for Detecting Faults Using SAT in Locked circuits

K. Yaswini1, Dr. A. M. Prasad2, T.V. Ramani3

M. Tech, VLSI&ES, JNTU Kakinada, Andhra Pradesh, India.

Professor, Dept. Of ECE, JNTU Kakinada, Andhra Pradesh.

Assistant Professor, Dept. Of ECE, JNTU Kakinada, Andhra Pradesh.

Abstract:- Achieving high fault coverage is also crucial for reducing manufacturing defects. This greedy test pattern generation generates test pattern with minimum set of test pattern count that is capable of detecting all the faults with reduced testing time. In this the existing methodology is logic locking where we insert a logic gate at various locations of the logic circuit with the keys attached to them. By enabling correct keys only, the operation operates correctly. Here by applying SAT (Boolean Satisfiability) attack we can detect the fault in the circuit. SAT based ATPG is used to identify faults in digital circuits. SAT solver finds test pattern effectively to detect the fault. The existing method consists of two approaches the first approach is to generate one test pattern per fault and the second approach is to generate test pattern for a group of faults. We mainly targeted on Stuck – at faults. We propose greedy test pattern generation, So it is used in Automatic test pattern generation it is particularly used for digital circuits testing and fault detection capability. This greedy test pattern generation focuses on generating a minimal set of test vectors that can detect maximum number of faults. This greedy test pattern generation algorithm consists of the several steps, the first step is fault list initialization, the second step focuses on generating a pool of patterns, the third step is fault simulation, and the fourth step is greedy selection. Here we demonstrate that our method can achieve 100% fault coverage and the efficacy of this methodology can be shown on ITC' 99 benchmarks circuits. These ITC 99 (International Test Conference 1999) benchmark circuits are set of standardized digital circuits that are used for evaluating test pattern generation. Additionally, we observe significant test generation time savings in the proposed in the second approach, as analysing multiple faults together helps reduce computational conflicts.

Keywords- Fault coverage, Greedy test pattern generation, SAT, Fault detection, Logic

1. Introduction

Very Large-Scale Integration integrates millions or billions of transistors in to a single silicon chip to create a complex electronic circuit. Where as to integrate these chips, we required testing the circuit weather there is a fault or not. A Fault is an error or defect that is not caused intentionally which does not work properly or because of some disturbances. These faults are occurred due to errors in digital circuits. Whereas these faults need to be tested. So testing is a process of checking whether there is a manufactured IC meets the required specifications

or not. For example, if a wire in a circuit is unintentionally broken, it may prevent a signal from propagating, leading to a malfunction. There are several faults in digital circuits we choose Stuck-at Fault. Whereas Stuck-at faults are of two types one Stuck-at-one (s-a-1) and the other one is Stuck-at-zero (s-a-0). Now stuck-at –one is defined as the particular wire or node is constantly at logic 1 is called s-a-1. Stuck –at zero is defined as a particular wire or node is constantly at logic 0 is called s-a-0. For example, if a logic gate output is expected to be logic 1 but the output is at logic 0, so it is due to the manufacturing defect. Another type is the Bridging Fault, which occurs when two nodes in a circuit are unintentionally shorted together, causing them to share the same logic value. Such faults may arise from lithography errors, dust particles, or defects in metal interconnections of ICs. For instance, if two adjacent wires on a PCB accidentally connect, it can lead to unintended behaviour like incorrect logic states or excessive power dissipation. However, the tools used for fault detection are Xilinx Vivado, Icarus Verilog and PyCharm.

2. Existing Method

Here the existing method [1] consists of Logic Locking. Whereas logic locking is defined as inserting an extra logic in the digital circuit with some secret key attached to the digital circuit. Logic locking is performed to counter the IP privacy where circuit fault detection is happened by using secret key. Here is an overview how logic locking is performed in a digital circuit. for the below Fig. 1. consists of original circuit and XOR based logic locking where XOR gates are induced that there is a presence of fault site in the circuit. Where there is a fault in the circuit and that fault is replaced with the Xor gate. Here the secret key (K) is stored in to tamper proof memory as shown in Fig. 1.

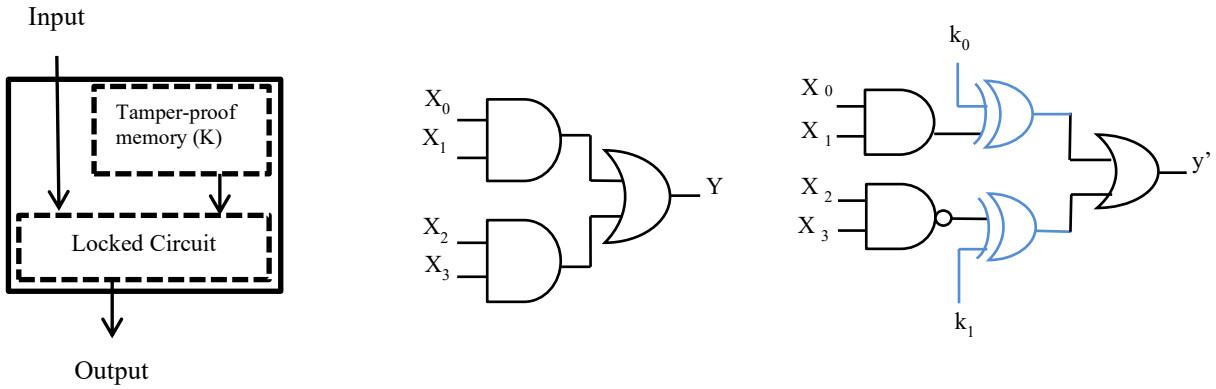


Fig.1. Logic Locking. (a) Logic Locking Overview. (b) Original Circuit. (c) XOR - based Logic Locking {k0, k1} = {01}.

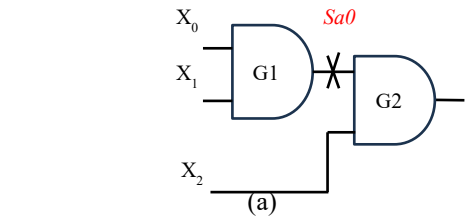
In the above Fig. 1. (b) It is a circuit with Boolean function $Y = X_0X_1+X_2X_3$ for the original circuit. Fig.1. (c) shows the modification of the original circuit with some logic insertion which is called logic locking here in the Fig.1. (c) XOR – based logic locking is used the Boolean formula for the circuit

$$Y'=(X_0X_1 \oplus K_0) + (X_2X_3 \oplus K_1)$$

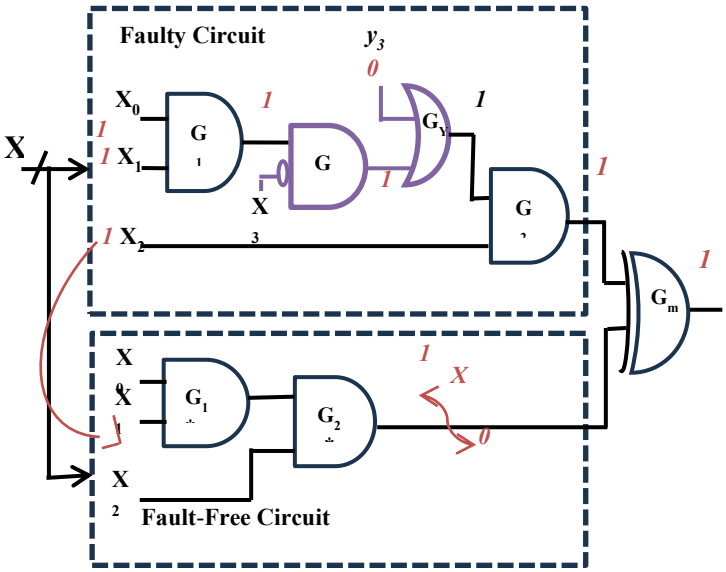
Where $\{k_0\ k_1\}$ are secret keys. The secret key {0, 1} that compares y' to y for all the input combinations.

A. How Sat Attack Is Performed on Logic Locking

Here the miter construction is performed for applying the SAT attack on the locked circuits as shown in [2].



(a)



(b)

Fig.2.Conflict in solving miter circuit (a) A simple circuit with *sa0* fault. (b) Conflict during SAT assignment

A miter circuit construction constructs a circuit that compares the outputs of faulty circuit and fault free circuit is called a miter circuit construction. In the Fig. 2. (a) Shows a circuit a simple circuit with *sa0* Fig. 2. (b) Shows a circuit with miter construction and also faces conflict with SAT assignment. Here the conflict arises due to the input combination of {1, 1, 1} if the result of miter circuit is 1 that means the fault is detected if we consider {1,1,1} as input to faulty one then the output is 0, for the fault free circuit the output is also 0 that means the fault is not detected now the SAT solver should perform backtrack and it resolves the conflict with logic 0 therefore the output of the fault free circuit is logic 1 then the fault is detected.

The steps how SAT performs on logic locking

1. The inputs are locked circuit with K value and unlocked circuit and output K_c that unlocks the circuit.

2. Initialize i to 1 for the iteration count purpose and make the construction of miter construction that means the comparison of faulty and fault free.

3. Use the SAT solver to find the Distinguishing input pattern and the correct keys only when both the outputs of faulty and fault free are not equal.

4. Repeat the loop until the correct DIP is found for each iteration make the increment of the counter and exit the loop after finding the DIP.

Now we will see how logic locking is performed on the logic gates

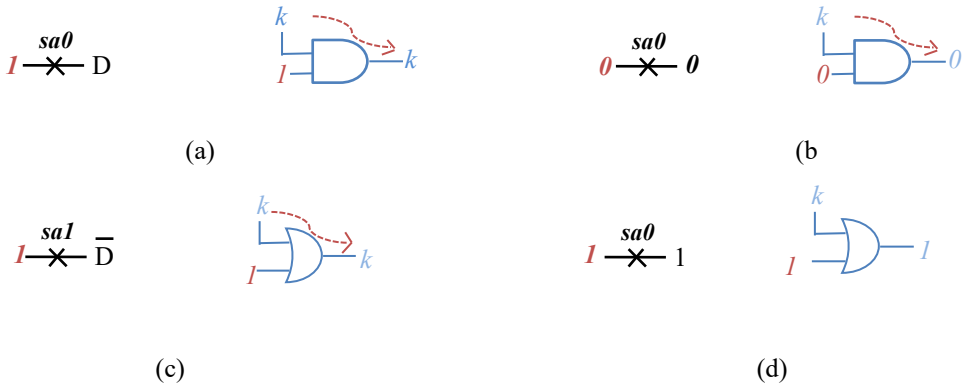
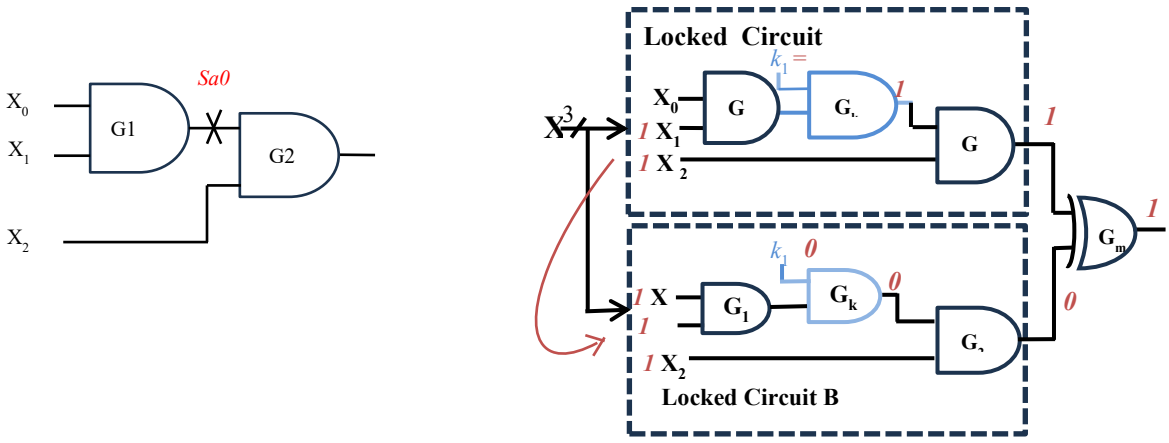
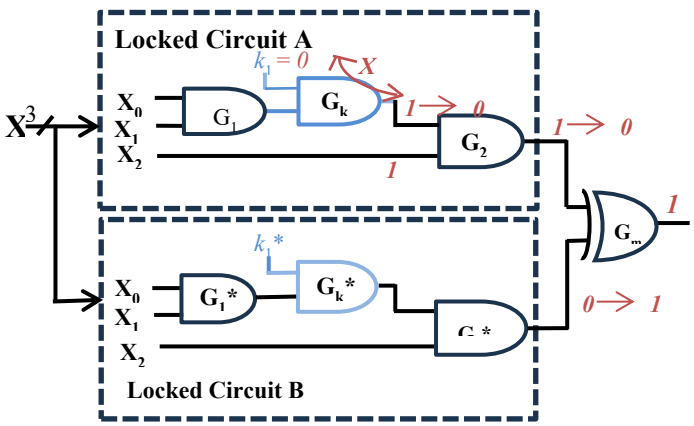


Fig.3. Logic locking modeling with saf with AND or OR key gate (a) successful propagation of key k with logic1. (b) Failed propagation of k with logic 0, a sa1to OR key gate. (c) Successful propagation of k with logic 0, and (d) failed propagation of k with logic 1

There are two approaches in fault detection first one approach is generating one test pattern per fault and the second approach is generating a test pattern for a group of faults.

1) Approach-1: Generating one test pattern per fault Approach1 focuses on SAT-based method to find one input pattern that is capable of detecting a specific fault by DIP between correct and incorrect circuit behaviour (via a 1-bit key). The sat solver is to find an input where faulty and fault free circuit produces different outputs and this pair is saved as constraint for fault detection.





(c)

Fig.3. The test pattern generation with the SAT attack miter. (a) A simple circuit with a sa0 fault. (b) No backtrack in deriving the satisfiable assignment with k_1=1. (c) Backtrack at an earlier stage with k_1 = 0.

Here in the Fig. 3(a) represents the circuit with simple sa0 fault and the Fig. 3(b) shows the circuit with no back tracking is applied for the circuit Fig.3(c) represents the circuit with back tracking and makes the circuit to find satisfiable DIP to detect the fault.

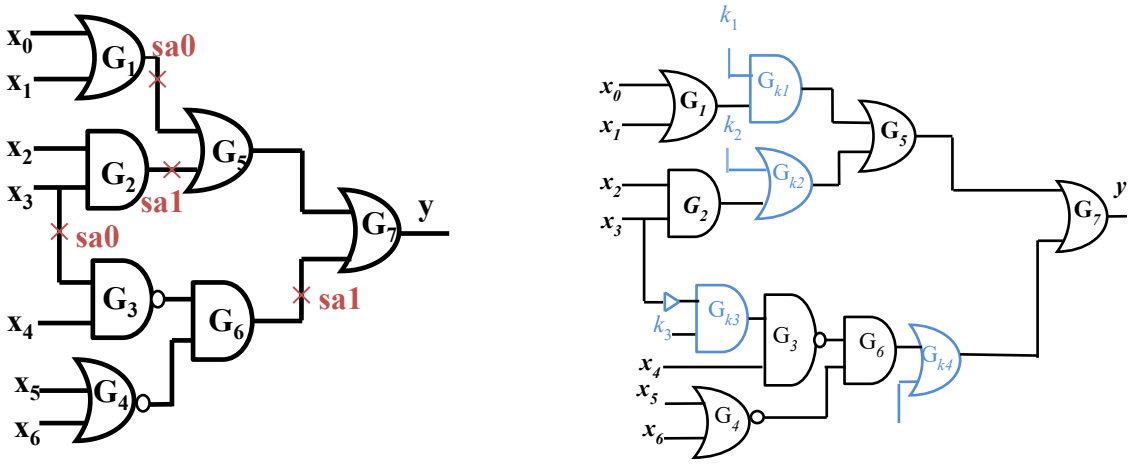
The steps how to detect a fault with approach1

1) First initialization that initialization consists of circuit description and a reference circuit and also the fault list that fault list consists three different items they are detected faults, undetected faults, and patterns used.

2) A SAT attack is performed on the faulty circuit versus the reference circuit to find the distinguishing pattern and the key.

3) If no pattern is found, that fault may be undetected and add it to redundant list and the pattern and key is found then added to detected fault list and the pattern is saved to the patterns list.

2) Approach-2: Generating a test pattern for a group of faults focuses on multiple faults in a circuit. While the first approach1 deals with a single pattern for a single fault, instead of adding single key circuit in approach1 we will add multiple key gates where the number of keys is same as faults to be analyzed. From the SAT point of view, each distinguishing pattern and in general, removes the multiple incorrect keys from the search space. Here the locked gates are replaced with AND, OR gates if the fault is SA0 then SA0 is replaced with AND gate and then SA1 is replaced with OR gate.



(a) (b)

Fig .4. Test pattern generation with group of faults (a) circuit with multiple faults (b) the saf equivalence with logic locking

The following steps are useful to detect the fault using approach2 [1]

1)First initialization that initialization consists of circuit description and a reference circuit and also the fault list that fault list consists three different items they are detected faults, undetected faults, and patterns used.

2) Now apply logic locking and convert in to testable format and also perform SAT attack to obtain multiple DIPS.

3) Use the fault simulation to check whether the patterns detect the faults by the generated DIPS.

Proposed Method

Here we propose greedy test pattern generator for fault simulation [10]. It is a strategy of Automatic test pattern generator used in digital circuits mainly the greedy test pattern duty is to generate a minimal set of test pattern that can detect maximum number of faults. For proving this we need to use greedy algorithm. So we are applying this greedy test pattern generation to the bench mark circuits to reduce the test pattern count.

Here are the steps how the greedy test pattern generator works

Step1: Initialization

1) Now inject the faults to the bench mark circuits here we were testing the stuck – at faults.

2) Have an empty test pattern set.

Step2: Pattern Evaluation

For each of the generated pattern check how many faults the pattern can detect

Step 3: Select Best pattern

Select the pattern that can detect maximum number of undetected faults which were present.

Step 4: Fault Elimination Remove all the faults that are detected by the chosen pattern from the list of undetected faults.

Step 5:

Repeat all the steps from step 2-4 until all the faults are detected and stop repeating until all the faults detected.

Now we will see this proposed approach in the bench mark circuits in the results column.

4. Results

The Existing method results were shown here the Fault Detection on ITC 99 benchmark circuits with using Logic Locking In the ITC-99 Benchmark circuits, without using the Logic Locking approach, some faults remain undetected. However, by applying the Logic Locking concept, all stuck-at faults in the circuits are successfully. The proposed greedy test pattern generation generated minimal number of test patterns that detects all the faults and reduction is increased compared to the existing method reduction in TABLE1 as the proposed approach can detect 100 percent fault coverage.

TABLE1: Stuck-at Fault Summary for ITC'99 Benchmark

Benchmark	Total Faults	With applying Logic Locking				With proposed greedy approach			
		Number of test patterns	Timing report	Detected faults	FC (%)	Number of test patterns	Timing report	Detected faults	FC (%)
b04_opt_C	3554	3	0.07	3549	100	1	0.02	3549	100
b04_C	4144	5	0.11	4094	100	2	0.6	4094	100
b05_opt_C	3272	1	0.08	3265	100	1	0.03	3265	100
b05_C	5850	30	558.7	4747	100	16	530.2	4747	100
b07_opt_C	2456	11	0.05	2455	100	5	0.02	2455	100
b07_C	2470	5	0.08	2464	100	2	0.02	2464	100
b11_opt_C	3318	7	0.07	3316	100	3	0.03	3316	100
b11_C	4378	2	0.14	4212	100	1	0.7	4212	100
b12_opt_C	6048	17	0.10	6047	100	7	0.3	6047	100
b13_C	1928	8	0.05	1848	100	3	0.02	1848	100
b14_opt_C	58584	3	11.6	58265	100	1	8.3	58265	100
b14_C	35844	5	30.9	35806	100	1	20.5	35806	100
b15_opt_C	48220	7	170.4	46922	100	2	150.2321	46922	100
b15_C	53470	19	10.9	51952	100	10	5.8	51952	100
b17_opt_C	157418	12	4780.1	154248	100	5	4540.2	154248	100
b17_C	192174	1	133.7	187897	100	1	120.2	187897	100
b20_opt_C	79748	3	22.3	79644	100	1	15.3	79644	100
b20_C	118298	25	135.8	117614	100	10	110.6	117614	100
b21_opt_C	80504	39	22.8	80398	100	17	14.6	80398	100
b21_C	120436	3	247.7	119742	100	1	220.6	119742	100
b22_opt_C	1145656	52	31.0	114387	100	25	18.9	114387	100
b22_C	175510	26	584.6	174653	100	10	4679	174653	100
b18_opt_C	469602	7	1471.1	469044	100	3	1321.5	469044	100
b18_C	672242	1	3740.1	668478	100	1	2740.5	668478	100
b19_C	1355584	8	4675.1	1347025	100	3	2428.5	1347025	100

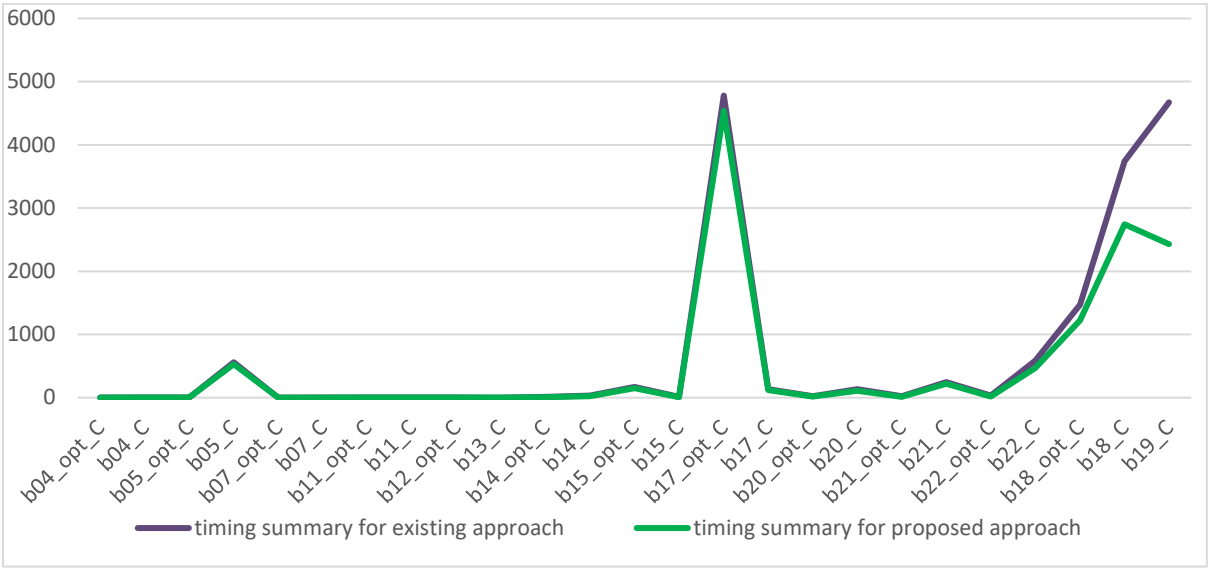


Fig.5 Timing report Analysis for Bench mark circuits

TABLE2: Comparison between Approach1, Approach2 and Proposed Approach on Number of Test Patterns

Benchmark	Approach 1	Approach 2	Reduction for existing approach	Proposed Approach	Proposed Reduction
b05_opt_C	4	1	75.00%	1	75.00%
b11_C	5	2	60.00%	1	80.00%
b15_opt_C	11	7	36.36%	2	81.81%
b17_opt_C	26	12	53.85%	5	80.7%
b17_C	1	1	0.00%	1	0.00%
b20_C	53	25	52.83%	10	81/1%
b21_C	8	3	62.50%	1	87.5 %
b22_C	61	26	57.38%	10	83.6%
b18_opt_C	14	7	50.00%	3	78.57%
b18_C	4	1	75.00%	1	75.00%
b19_C	99	8	91.92%	3	96.96%

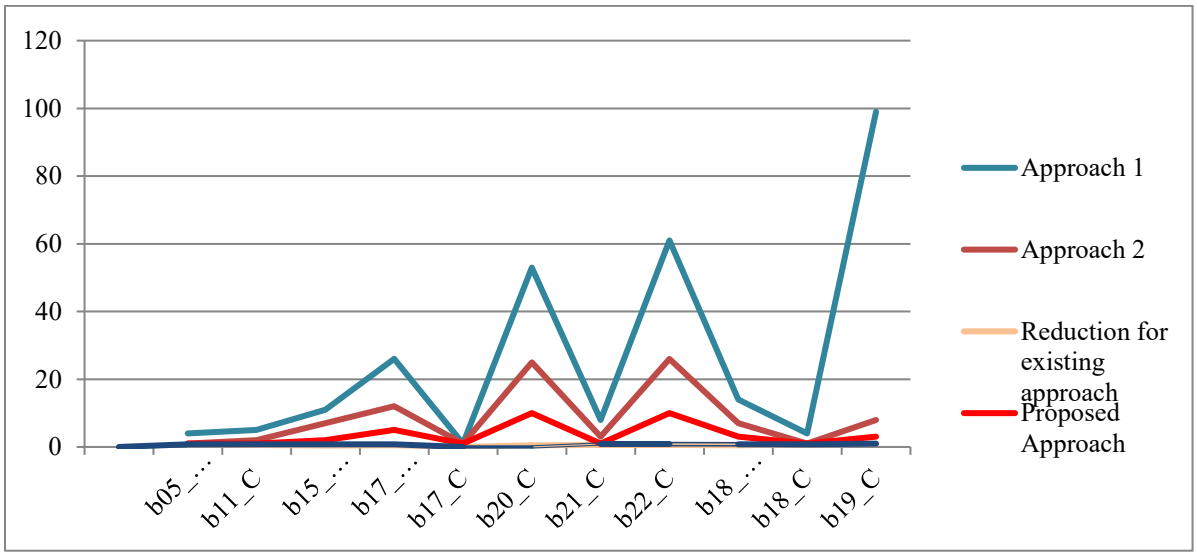


Fig.6 Test pattern Analysis for Bench mark Circuit

5.Conclusion

In this proposed method we have implemented and presented greedy test pattern generation and it achieves high fault coverage with minimal number of test patterns in digital circuits that detects all the faults with highest fault coverage. We have applied greedy test pattern generation to the ITC benchmark circuits with less test pattern and also reduction percentage is increased. Therefore, having low number of test patterns become one of the major objectives in VLSI testing with less time and achieves high fault coverage.

6. References

[1] Yadi Zhong, Ujjwal Guin, Comprehensive Test Pattern Generation Approach Exploiting the SAT Attack for logic locking.

[2] Y. Zhong and U. Guin, "Complexity analysis of the SAT attack on logic locking," IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst., pp. 1– 14, 2023. [3] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT based attack on compound logic locking techniques," in Proc.Des. Automat. Test Europe Conf. Exhib., 2021, pp.

[3] N. Limaye, S. Patnaik, and O. Sinanoglu, "FaSAT: Fault-aided SAT based attack on compound logic locking techniques," in Proc.Des. Automat. Test Europe Conf. Exhib., 2021, pp.

[4] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in Proc. Great Lakes Symp. VLSI, 2017, pp. 179–184.

[5] A. Jain, M. T. Rahman, and U. Guin, "ATPG-Guided fault injection attacks on logic locking," in Proc. Int. Conf. Phys. Assurance Inspection Electron., 2020, pp.

[6] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in Proc Int Symp Hardware Oriented Secur Trust, 2015, pp 137–143.

[7] J. Rajendran et al., "Fault analysis-based logic encryption," IEEE Trans. Comput, vol 64, no2, pp 410–424, Feb 2015.

[8] T. Larrabee, "Test pattern generation using Boolean satisfiability," IEEE Trans. Comput. Aided Design Integr. Circuits Syst., vol. 11, no. 1, pp. 4–15, Jan. 1992.

[9] S. Eggersglus, R. Wille, and R. Drechsler, "Improved SAT-based ATPG, more constraints, better compaction," in Proc. Int. Conf. [5] A. Jain, M. T. Rahman, and U. Guin, "ATPG-Guided fault injection attacks on logic locking," in Proc. Int. Conf. Phys. Assurance Inspection Electron., 2020, pp.

[10] J. Raik, A. A. Jutman, R. Ubar Tallinn Technical Univ., Raja 15, 12618 Tallinn, Estonia, "Fast Static Compaction of Test Sequences using Implications and Greedy Search".