An Efficient Software Tool based on SCOAP for Testability Analysis of Combinational Circuits

Sreeja Rajendran [1], Mary Lourde R [2]

Department of Electrical and Electronics Engineering
Birla Institute of Technology and Science Pilani, Dubai Campus, UAE.

[1]sreejamanojnair@gmail.com, [2] marylr@dubai.bits-pilani.ac.in

***Abstract* - Identification of regions of a logic circuit which are poorly testable is referred to as Testability Analysis. It is generally computed using either Sandia Controllability Observability Analysis Program (SCOAP) or Probability based Analysis. Logic circuits usually consist of thousands of gates. Software tools which performs the testability analysis of digital circuits will ease the computation process and at the same time provide high degree of accuracy. This paper discusses a software tool developed to compute the testability parameters namely the controllability and observability of each node in the circuit using MATLAB. The developed testability analysis software tool takes Hardware Description Language (HDL) file as input and generates the controllability and observability values of each node described by the HDL. The possibility of utilizing testability analysis for detection of Hardware Trojans (HT) is also discussed in this paper. The effect of HT insertion on a system is evaluated using c-17 benchmark circuit and the results are presented.**

Keywords - Testability Analysis, SCOAP, Trojan Detection.

I. INTRODUCTION

The entire process of converting a design into a product passes through several phases. The IC design house focusses on developing a circuit design which is efficient both in terms of timing and performance. This is made possible through partitioning, floor planning, placement, routing etc. To implement the design, the design house requires modules (intellectual property) from third party vendors as well as a fabrication facility to construct the design on chip. As attention is devoted to improved performance of the design, determining the testability of the developed design is seldom done at the designer's end. The relevance of testability analysis of a circuit arises with secure IC design. It is very important in the event of Hardware Trojan (HT) attacks. A HT is any unwanted alteration to the design resulting in a deviation from the desired functionality at a later point in time. Any of the entities involved in the VLSI design cycle could be responsible for the malicious insertions. And therefore it is extremely important to prepare the system to foil such attempts by adversaries. For this purpose the designer should clearly know the regions of the circuit which are likely to be targeted by intruders. This information can be obtained by testability analysis of the logic circuit.

Testability analysis is a two-step process which is performed on gate level circuit description. Controllability and Observability are the two parameters that decide the testability of every node in the circuit. Controllability is a measure of how effectively we can set a particular signal to logic level 1or 0. On the other hand Observability measures the difficulty in propagating the logic value at a node to the output of the circuit. The first step in the process is to compute the controllability values of all the nodes in the circuit starting from the primary inputs and proceeding to the outputs. The computation of observability values commences from the output node travelling all the way up to the inputs.

Testability analysis tools based on SCOAP are not readily available or provided by any major CAD tool developers in the market. The newly developed simulation tool provides the following advantages:

a. The tool works on the Verilog netlists of the circuit which is the universal standard and therefore does not require any additional formatting.
b. Can compute the testability parameters of combinational circuits with any level of complexity.
c. The tool works on MATLAB platform.
d. The computed parameters can be further used in classification of circuit nodes in to safe and vulnerable zones and thereby eases the job of designer engineers and provides them with vital information regarding the security of the designed circuit.
e. The tool can also be applied to compute testability parameters of reverse engineered circuit netlists which can be an effective mechanism for Hardware Trojan detection.

The rest of the paper is organized as follows: Section 2 gives a brief description of the various categories of hardware attacks followed by the state of the art hardware detection techniques in section 3. Motivation behind the work, Flowchart for Implementation of Algorithm for Automatic Computation of Testability Parameters, Existing Methods for Testability Analysis, Algorithm description,

Results and Application of Testability Analysis for HT detection is given in the sections that follow.

II. HARDWARE ATTACKS

Any attacks that results in a deviation in the functionality of a chip from the expected output can be broadly termed as hardware attacks. These attacks can result in leakage of information through an implanted backdoor, damage the chip or even result in Denial of Service (DoS). Hardware attacks can be launched at various levels of abstraction like circuit, gate and RTL. Attacks at the circuit level are initiated through changes in manufacturing process parameters like doping, channel width etc. At gate level, logic gates are added or deleted to launch attacks. Hardware attacks at RTL level can be introduced by the addition of states to the finite state machine.

Hardware attacks on ICs can be classified into three on the basis of their intended actions which is shown in table1. The first category can be called data attacks which aims at retrieval of secret data on an IC (eg. smart cards). Design attacks forms the second class of attacks. The target of these attacks is to acquire design information of the chip for the purpose of counterfeiting (eg. Reverse engineering). The last class of attacks, referred to as functionality attacks attempts to alter the functionality of the IC resulting either in faulty operation or rendering it non-functional.

The motive behind hardware attacks can be economic reasons or even for intelligence purposes. Integrated circuits are the backbone of all safety critical systems. Launching hardware attacks on such sensitive systems can result in catastrophic effects. To protect digital systems from these hardware attacks, designers should have a vast knowledge about security vulnerabilities and the measures to overcome them. To build a secure system, the designers should be aware of the possible attacks to watch out for and about how to protect the system from such malign attacks. Analyzing the susceptibility of a design to malicious intrusions is the key to developing a secure system. Thorough analysis of a circuit design is essential to determine its susceptibility to Hardware Trojan insertion. Nodes with poor testability have been identified as the potential sites for HT insertion [1, 2, 3]And therefore testability analysis is an integral step to assess the vulnerability of circuit designs.

TABLE 1. CLASSIFICATION OF HARDWARE ATTACKS ON IC BASED ON INTENDED ACTIONS

Data Attacks		**Design Attacks**	**Functionality Attacks**
Invasive	Non-invasive	Reverse engineering	Hardware Trojans (addition/ deletion of gates)
Reverse engineering	Side channel attacks		
Micro-probing	Hardware Trojans		

III. STATE OF THE ART IN HARDWARE TROJAN DETECTION

HT detection can be classified into static and dynamic approaches. Static approach relies on formal verification methods for the identification of unused/redundant logic in the genuine circuit. This approach can be utilized either in design stage or post fabrication. In [4] , the approach is used post fabrication where in unused filler cells are replaced with functional standard cells which are connected together. These cells form a combinational circuit which performs an arbitrary function. The signature obtained from the Output Response Analyzer will differ if any of these standard cells have been replaced. Unused Circuit Identification (UCI) presented in [5] is a hybrid hardware-software approach to detect the presence of HTs in design phase. Any unused circuitry in the design is flagged as suspicious and detached from the genuine circuit if it does not get activated during any of the design verification tests. Logic is inserted during runtime hardware check to ascertain if the circuit flagged as suspicious is legitimate and thereby raising an exception. FANCI [6] works along similar lines where in Boolean functional analysis is used to flag suspicious nets in the circuit. A functional truth table is constructed for every gate in the design in order to determine the control each input exerts over the output of the gate. An input which weakly affects the output of a logic gate is termed nearly unused logic, which could be used by an adversary. The authors in [7] compare the current signatures of the IC at two different time frames using a Temporal Self Referencing approach. This method improves HT detection sensitivity by eliminating the effect of process variations and capacitive coupling.

Dynamic approach on the other hand, requires activation of HTs in order to be able to detect them. This is made possible by increasing the transition probabilities of rare nodes (nodes which rarely get activated) in the circuit. This helps to activate the Trojans and thereby detect their presence on the chip. There are various mechanisms for Trojan activation. In [8] and [9] the authors have developed a gate level Trojan activation mechanism using dummy scan flip-flops and 2:1 MUX respectively. In Trojan activation using dummy scan flip-flop, probability analysis is used to determine the set of rare nodes whose transition probability falls below a preset threshold value. Dummy scan flip-flops are inserted to improve the transition probability of such nodes. A 3 input gate has been used to improve the transition probability of signals in [10]. Statistical approach [11] [12] [13] is yet another approach for HT activation which focuses on developing input test vectors which facilitate multiple excitations of the rare nodes thereby triggering the Trojans and resulting in their detection. In

other words, statistical approach improves the coverage of test vectors in detecting HT.

Statistical learning algorithms have been used to develop a framework to detect HT insertion during fabrication [14]. Here a linear system is formulated for every chip using a sparse gate profiling technique. Next the process variations for each chip is calibrated using Bayesian inference. Finally Simultaneous Orthogonal Matching Pursuit algorithm is used to solve the linear systems for a batch of chips. HT are detected by analyzing the solutions. The authors in [15] extracted Trojan features from Trojan inserted benchmark circuits. These features like low switching logic gates, Multiplexer Select lines etc. were used to create a score based threshold to segregate HT inserted netlists from genuine netlists. Using Random Forest Classifier, the best set of Trojan features which can effectively detect Trojan nets is obtained in [16]. Features are extracted from IC images obtained through Reverse Engineering in [17]. Here SVM is used to classify the ICs as Trojan-free and Trojan-inserted. Features are determined by comparing the physical layouts in RE images with that of golden IC. In this method images of each layer of a set of N chips are divided into non overlapping grids. For each grid in a layer features are extracted. The trained classifier obtains a decision boundary for each layer. Based on the decision, grids in a layer are classified as genuine or malicious.

IV. MOTIVATION BEHIND WORK

In recent years significant work related to the detection and analysis of Hardware Trojans has been carried out. However, very little emphasis has been given to the susceptibility assessment of circuit designs. The authors in [18] have presented the vulnerability analysis of a circuit at various levels of abstraction. This helps to quantify the difficulty in detecting the presence of an HT in the circuit, if inserted. HT attacks implemented at gate level involves addition or deletion of gates in the circuit. The size of an HT is negligible as compared to the size of an IC which is comprised of millions of gates. And therefore they do not have an evident impact on the power and timing profiles of the circuit. Hence they can be wrongly accounted as noise arising due to process variations or environmental factors. This is the fundamental challenge in HT detection.

In order to prevent HT insertion completely, primary importance needs to be devoted to prepare the system to thwart malicious insertions. Since testability analysis is performed at gate level, we restrict the circuit design analysis to gate level of abstraction. The gate level description of a circuit is a collection of Boolean functions implemented using logic gates. A HT inserted at gate level can be a single gate, a collection of gates or a combination of gates and flip-flops, which produce an undesired effect at some point during the operation of the device. In order to understand the working mechanisms of gate level HTs, it is important to know the components of a HT. There are two basic components for a HT namely Trigger and Payload. In simple words Trigger is the activating mechanism and Payloadinitiates the undesired action. The Trojan classification [19] shown in figure 1 includes the various digital triggers and payloads.

Combinational and sequential are the two primary classes of digitally triggered Trojans. A combinational Trojan is a collection of gates where the trigger section taps signals from multiple nodes in the genuine circuit and activates the payload. The signals which are likely to be chosen would be those signals whose logic levels change very rarely. A sequential Trojan uses memory elements in addition to logic gates from the genuine circuit. This makes detection of the presence of HTs more complex. And the test vectors applied during the testing phase might not activate the Trojan circuitry. Undetected HTs can have adverse effects on the reliability and expected lifetime of the circuit [20].The only approach to prepare the digital systems against malicious intrusions of this kind is through secure design. This process is twofold. Firstly we need to identify the nodes in the original circuit which could fall prey to malicious intrusions. This information can be obtained only through the process of Testability Analysis. This reinforces the necessity to analyze the testability of any digital system to protect it from unwanted interference. Testability also ensures quality and reliability of mission critical systems [21].

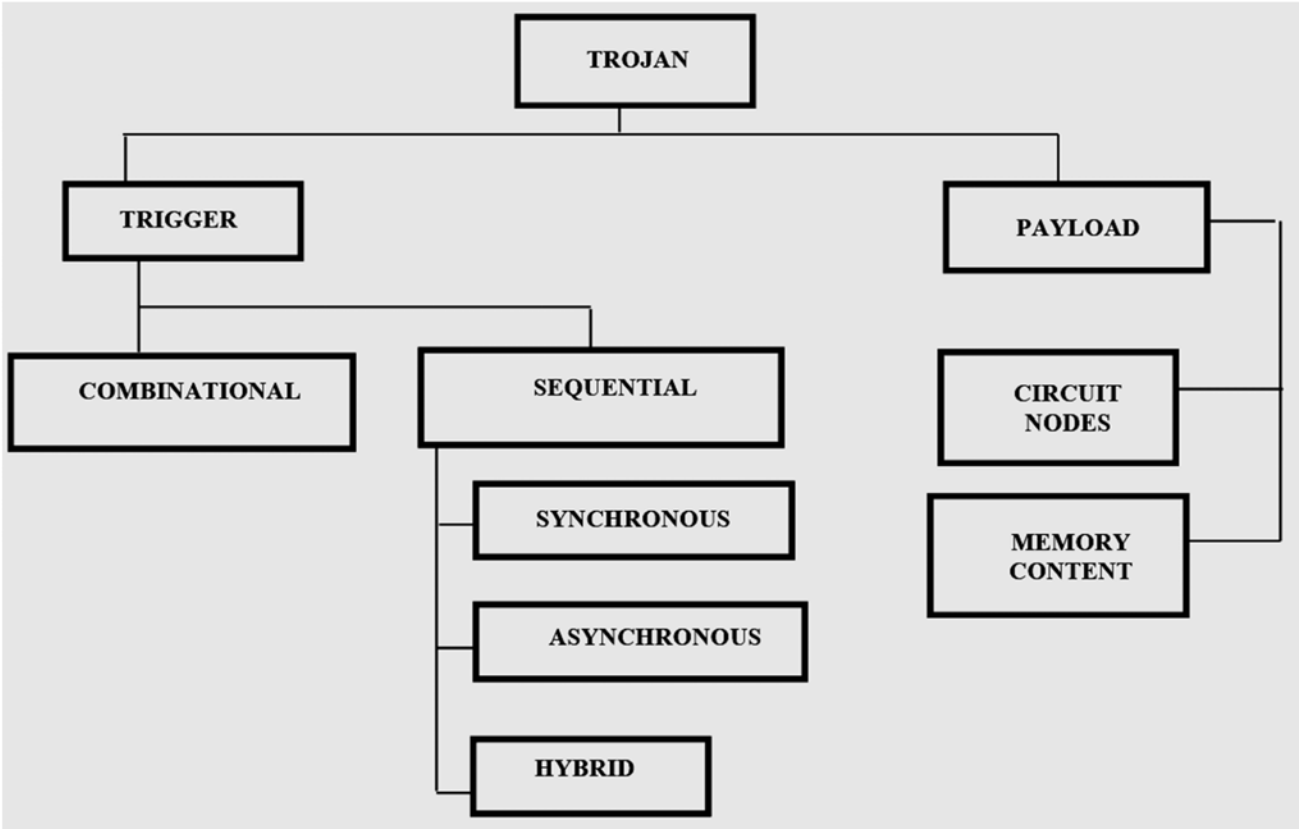
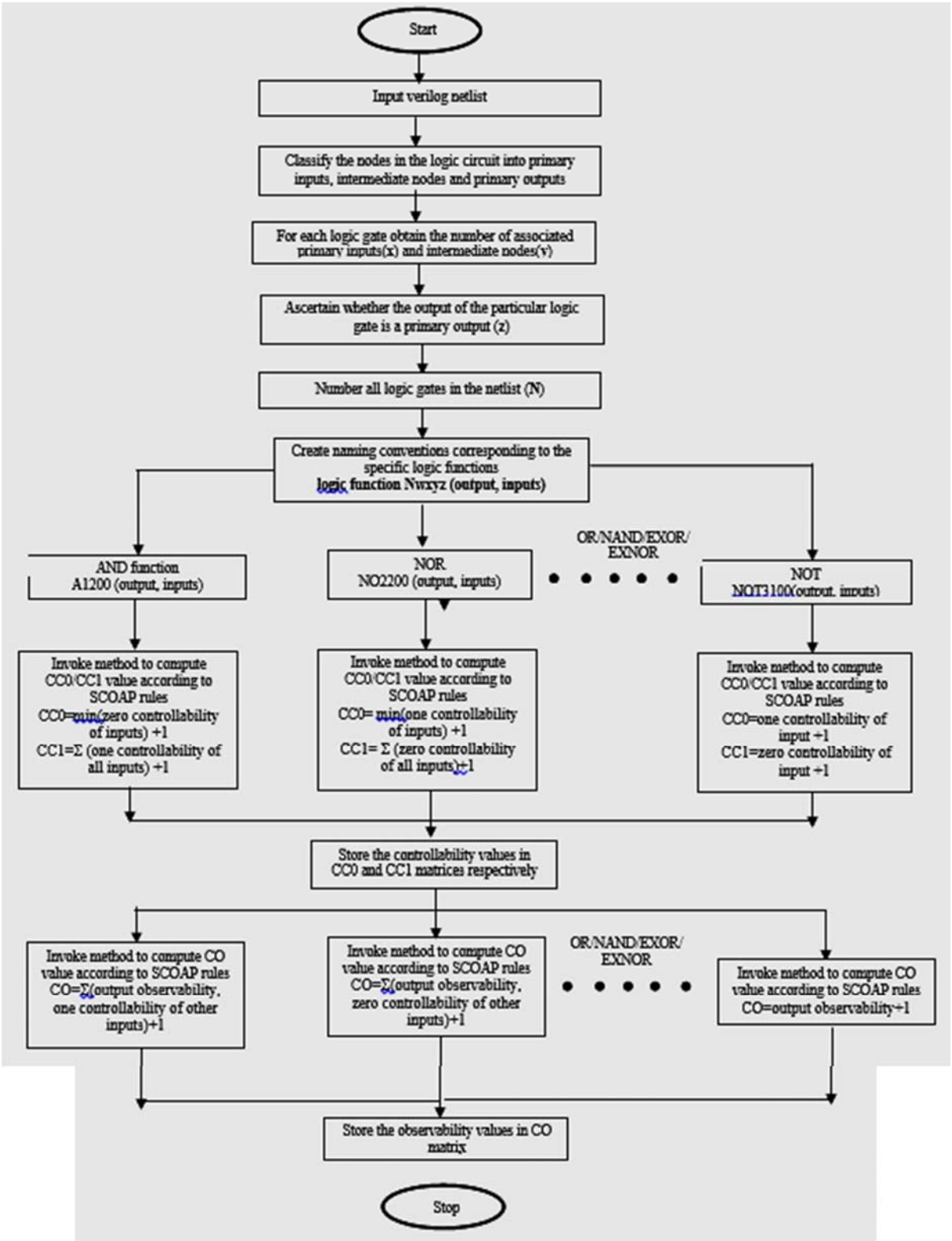


Fig. 1. Classification of Gate Level Trojans

TABLE II. SCOAP RULES FOR VARIOUS LOGIC FUNCTIONS[22]

Logic gate	Combinational Zero controllability	Combinational One controllability	Combinational Observability
AND	min(zero controllability of inputs) +1	Σ (one controllability of all inputs) +1	Σ(output observability, one controllability of other inputs)+1
OR	Σ (zero controllability of all inputs) +1	min(one controllability of inputs)+1	Σ(output observability, zero controllability of other inputs)+1
NOT	one controllability of input +1	zero controllability of input +1	output observability+1
NAND	Σ(one controllability of all inputs)+1	min(zero controllability of all inputs)+1	Σ(output observability, one controllability of other inputs)+1
NOR	min(one controllability of inputs)+1	Σ (zero controllability of all inputs) +1	Σ(output observability, zero controllability of other inputs)+1
EXOR	min[CC1(input1)+CC1(input2), CC0(input1)+CC0(input2)]+1	min[CC1(input1)+CC0(input2), CC0(input1)+CC1(input2)]+1	Input1- Σ(output observability, min(CC0(input2),CC1(input2))+1 Input2- Σ(output observability, min(CC0(input1),CC1(input1))+1
EXNOR	min[CC1(input1)+CC0(input2), CC0(input1)+CC1(input2)]+1	min[CC1(input1)+CC1(input2), CC0(input1)+CC0(input2)]+1	Input1- Σ(output observability, min(CC0(input2),CC1(input2))+1 Input2- Σ(output observability, min(CC0(input1),CC1(input1))+1

V. FLOWCHART FOR IMPLEMENTATION OF ALGORITHM FOR AUTOMATIC COMPUTATION OF TESTABILITY PARAMETERS



VI. EXISTING METHODS OF TESTABILITY ANALYSIS

Testing is the classical approach to detect defects that arise in chips during manufacturing or packaging. Considering the fact that malicious intrusions are capable of evading detection and also because testing methodology has been primarily designed to detect defective chips, there is a fairly high chance that security problems go undetected. The process of assessing the testability of digital systems is defined as testability analysis. In simple words, testability is a numerical measure of the impact of a signal value at a particular node on the output of the circuit. It is captured using the parameters Controllability and Observability. In case the signal at a particular node is corrupted due to malicious interference, these parameters help to understand the chances that the interference will reflect in the value at the output signal. In other words, it is indicative of the probability of discrepancies being captured during testing phase. There are various approaches for testability analysis like SCOAP [22] [23], PREDICT [23], COP [24] etc.

Testability analysis can be topology based or simulation based. Topology based methods include SCOAP and Probability based analysis at gate level. Topology based testability analysis is a static approach. Goldstein developed SCOAP testability analysis. The major advantage of SCOAP algorithm is that it has a linear computational complexity (O(n)). It therefore provides a speedy estimate of the circuit's testability. The computed testability values range from 1 to infinity. The developed simulation tool is based on SCOAP.

Probability based analysis works on the signal probabilities as the name suggests. The probability values range between 0 and 1. Smaller the value, greater the difficulty to control or observe the signal. Though it provides higher level of accuracy of computed values, the computational complexity is exponentially large.

Simulation based analysis on the other hand involves the application of test vectors for logic simulation as well as fault simulation of digital circuits. In logic simulation, test vectors are applied and responses are collected like occurrence of 0's, 1's, transition from 0 to 1/ 1 to 0 etc. These data are analyzed to locate areas of poor testability. All the above mentioned methods can be employed for testability analysis at gate level of abstraction.

At Register Transfer Level (RTL), behavioral synthesis has been used for testability analysis. This has generated testability measures for finite state machines. This is commonly referred to as RTL level testability analysis. There are various approaches to perform testability at RTL level. In[25],a structured graph is used to represent the RTL circuit where the registers are represented by vertices and the edges between them are functional blocks. The maximum level in the structured graph indicates the difficulty in testing the circuit. In order to include high level design information about functional block another approach is suggested in [26]. Here a Directed Acyclic Graph (DAG) is used to represent the data flow within the functional blocks. The nodes of the DAG denotes arithmetic, logical or data transfer operations while the edges represent the signals involved in the operation. The primary advantage of RTL is that it improves data path testability at reduced area overhead.

Following are the highlights of SCOAP method of testability analysis

Provides low computational complexity

Can be used to predict the length of test vectors.

Gives a clear picture about the number of signals that need to be manipulated in order to control or observe a particular signal. (all values are positive integers)

When addressing hardware attacks, the overestimated testability metrics generated by SCOAP can be used as a precautionary measure to protect the system against malicious intrusions.

In order to clearly understand the algorithm for testability analysis, it is important to know the basics of SCOAP on which the tool works. For a combinational circuit the numerical values computed for a signal at a node are

CC0 - combinational zero controllability

CC1 - combinational one controllability and

CO - combinational observability

As a rule, CC0 and CC1 values of primary inputs to a circuit are set to 1. At the same time, CO of primary outputs are set as 0. The computation of the parameters for various logic gates are given in table 2.

VII. ALGORITHM FOR TESTABILITY ANALYSIS

The input for the tool is a Verilog netlist. For computation purpose, Verilog netlists of ISCAS benchmark circuits have been used. The Verilog file is taken as input by the MATLAB program. It is then converted into a character array. The next step is to categorize the netlist into arrays of primary inputs, intermediate nodes and primary outputs which are named primary, internode and output respectively. Once the various nodes are classified into the aforementioned three groups, the next step in the algorithm is to use a specific naming nomenclature and recreate the netlist in the new format. As it is evident from table 2, controllability/observability values of the signal at a node depends on the controllability values of inputs and also observability values of outputs. Hence the nomenclature includes the number and category of inputs and outputs. The nomenclature used in the development of the tool is as follows

orNwxyz(output, inputs)

N- name of gate. Eg. OR gate is named O.

w- number of the gate. Eg. if it is the third OR gate in the circuit, w will be 3.

x- number of primary inputs.eg. if one of the inputs to the OR gate is a primary input, x will be 1.

y-number of intermediate inputs. eg. if 2 inputs fall in this category, y will be 2.

z- primary output or not. eg. if the output of the OR gate is a primary output, z will be 1 else it will be 0.

For instance considering a line of code in the netlist of c17 benchmark circuit

nand NAND2_1 (N10, N1, N3);

NAND2_1 is the name of NAND gate. N10 is the output of the gate while N1 and N3 are the inputs. The algorithm denotes all NAND gates as NA and numbers the gates in the order of their occurrence in the netlist. NAND2_1 is the first NAND gate in the netlist of c17. Therefore it will be denoted as NA1.The algorithm also determines whether the output of the particular gate is a primary output or not. Similarly it also determines whether the inputs belong to primary or intermediate nodes category. N10 is an intermediate node while both the inputs are primary inputs as listed in the input statement. So the same line will be converted to the following format

nandNA1200 (N10, N1, N3);

c17 benchmark circuit Verilog netlist in original format is given below

```
module c17 (N1,N2,N3,N6,N7,N22,N23);
input N1,N2,N3,N6,N7;
output N22,N23;
wire N10,N11,N16,N19;
nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);
endmodule
```

The converted netlist after the nomenclature mapping is applied is as shown below

```
module c17 (N1,N2,N3,N6,N7,N22,N23);
input N1,N2,N3,N6,N7;
output N22,N23;
wire N10,N11,N16,N19;

nandNA1200 (N10, N1, N3);
nandNA2200 (N11, N3, N6);
nandNA3110 (N16, N2, N11);
nandNA4110 (N19, N11, N7);
nandNA5021 (N22, N10, N16);
nandNA6021 (N23, N16, N19);

endmodule
```

Methods are defined as per SCOAP rules, for all logic functions. For a particular logic function, the corresponding method is invoked to compute the testability parameters. The zero and one controllability values of all nodes are stored in a matrices named CC0 and CC1 respectively. The first value in the matrix CC0 i.e. CC0 (1) is the zero controllability value for the output node of the gate numbered 1 in the netlist i.e. N10. Similar is the case with CC1 matrix. The CC0 and CC1 matrices computed by the tool for the module c17 are given below

CC0 matrix

3 3 4 4 5 5

CC1 matrix

2 2 2 2 4 5

N10 has a CC0 value of 3 and a CC1 value of 2.

After computation of controllability values, observability values for every node is computed. The tool is devised with the purpose of focussing on the internal nodes of the circuit. So the observability parameters for the primary inputs are not listed out. Primary inputs are easy to control and cannot be tampered with. Now consider the NAND gate shown below

nandNA6021 (N23, N16, N19);

N23 is a primary output and therefore has a combinational observability value (CO) of 0. N16 is the output of the third NAND gate labelled NA3110 in the circuit (as can be seen in the converted netlist). Hence the observability value for N16 will be saved in CO matrix at CO (3). For N19 (NA4110), the computed value is stored at CO (4).

CO matrix

3 5 3 3

CO of N16 = 3 and CO of N19 = 3

VIII. RESULTS AND DISCUSSIONS

The testability parameters have been computed for various benchmark circuits and are listed in table 3. The functionality of the benchmark circuits and the total number of logic gates in the circuit are given in the table. It gives a picture of the complexity of the logic circuit. The maximum and minimum values for the controllability parameters CC0 and CC1 as well as the values for observability is tabulated. The maximum and minimum value of every parameter is indicated in the table so that the reader gains an understanding of how large the variation is within a particular logic circuit and also its level of complexity.

Ideally it is good if the CC/CO values are closer to theoretical minimum which is 1/0. Though SCOAP defines the method to determine testability parameters of the circuit, it does not provide a range of values to classify the nodes in a logic circuit as safe and susceptible ones. Classification of the nodes on the basis of testability parameters will help designers to identify critical nodes/regions in the logic

circuit which need to be attended to for safeguarding against malicious attacks

TABLE III. TESTABILITY PARAMETERS OF VARIOUS BENCHMARK CIRCUITS.

Benchmark Circuit	No of logic gates	Logic function	CC0 min	CC0max	CC1min	CC1max	CO min	CO max
c432	160	27 channel interrupt controller	2	613	2	222	1	1634
c499	202	32 bit single error correcting circuit	2	109	3	243	2	299
c1355	546	32 bit single error correcting circuit	2	4130	2	20687	1	20714
c2670	1269	12 bit ALU and controller	2	292	2	291	1	581
c880	383	8 bit ALU	2	264	2	265	1	260

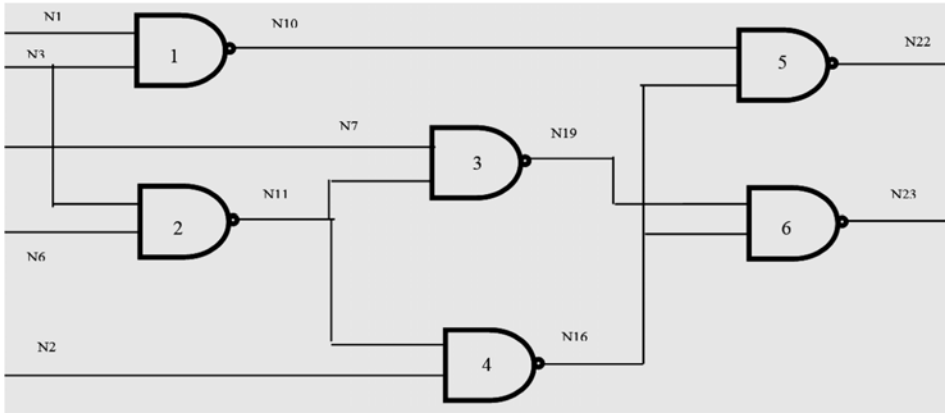


Fig 2. C-17 benchmark circuit

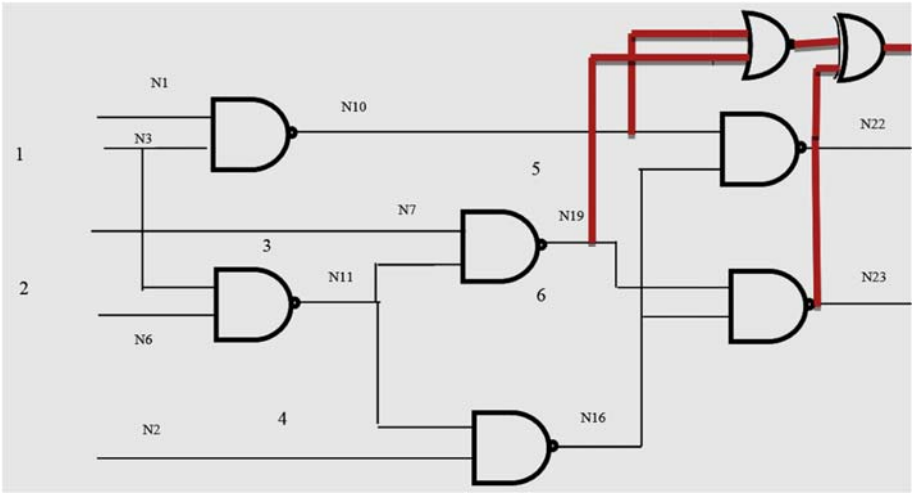


Fig 3. c-17 benchmark circuit inserted with Combinational Trojan 1.

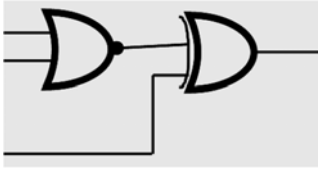


Fig 4. Hardware Trojan circuit.

TABLE IV. COMPARISON OF TESTABILITY PARAMETERS OF C-17 BENCHMARK CIRCUIT WITH AND WITHOUT TROJAN.

SIGNALS	COMBINATIONAL TESTABILITY			
	Trojan Free	Trojan-1	Trojan -2	Trojan- 3
N10	3/2/3	3/2/3	3/2/3	3/2/3
N11	3/2/5	3/2/5	3/2/5	3/2/5
N16	4/2/3	4/2/3	4/2/3	4/2/3
N19	4/2/3	4/2/7	4/2/7	4/2/7
N22	5/4/0	5/4/0	5/4/0	5/4/0
N23	5/5/0	5/5/4	5/5/4	5/5/4

TABLE V. TRIGGERING CONDITION AND ACTIVATION PROBABILITY OF INSERTED TROJANS [20]

Trojan	Triggering Condition	Trojan Activation Probability
Trojan-1	N10=0, N19=0	2/32
Trojan-2	N10=0, N11=0	4/32
Trojan-3	N10=0, N16=0	2/32

X. APPLICATION OF TESTABILITY ANALYSIS FOR HT DETECTION

The effect of combinational Trojans on testability parameters of the c17 circuit in ISCAS-85 benchmark suite is computed and tabulated in table 4. c-17 circuit shown in figure 2 has been chosen for ease of explanation. The combinational Trojan inserted is a 2 input NOR gate followed by a 2 input EXOR gate is shown in figure 3 [20]. Three different cases where the Trojan taps signals from different points in the genuine circuit and its impact on the testability metrics is shown in table 4. The Trojan circuit inverts the signal N23. All the three cases result in inversion of N23 for different triggering conditions. Logic 0 at both inputs of the NOR gate results in conditional complementing of signal N23 through the EXOR gate. c-17 circuit inserted with Trojan-1 is shown in figure 3.

From the testability analysis of these circuits, it is observed that in all three cases only the observability values for certain signals namely N19 and N23 have been altered due to the insertion of Trojans. This is because N19 is one of the inputs of gate 6 generating N23 signal. So the change in observability of N23 will result in a change in observability of N19. The other input of gate 6 is N16. N16 also drives gate 5. The observability value for the signal at a node with a number of fan out stems will be the minimum of the observability values of the fan outs. And therefore the observability values for N16 is taken from testability parameters of N22 and N10 which is lower than that of N23 and N19. Another point to be noted from the evaluation is that the controllability values of all signals remain unchanged. This is because the Trojan payload is not stitched to any intermediate nets. In such a scenario, changes will be reflected in the controllability values of those nets as well.

Activation of HT circuit is the key factor that ensures their detection by the application of test vectors. Table 5 shows the triggering conditions and activation probability for the various Trojans inserted in c-17 benchmark. It can be seen that the activation probabilities are as low as 6.25%. It should be noted that such low values of activation probability are obtained when a 2 gate Trojan circuit is introduced into a genuine circuit comprising of a total of 6 gates. This analysis is clearly indicative of extremely poor transition probability for HTs inserted into standard circuits comprising thousands of logic gates.

However using testability metrics to detect HT insertion eliminates the necessity to activate HT. Insertion of HTs will result in a change in testability metrics of a few nodes in the genuine circuit. Using the process of reverse engineering and an efficient path retrace algorithm, the presence of malicious gates can be determined in any circuit.

XI. CONCLUSION

An efficient general purpose software tool based on MATLAB is developed to compute the testability parameters of combinational digital circuits of any level of complexity. The computation of testability parameters of various benchmarked digital circuits has been carried out using the MATLAB based testability analysis tool that has been developed. The tool is developed based on Sandia Controllability Observability Analysis Program. It falls under the category of topology based testability analysis procedure. The tool eases the process of computation of controllability and observability values of various signals in the circuit. The testability parameters computed using the tool can be used to develop methods to classify the nodes of a logic circuit into safe and susceptible categories. This in turn will lead to secure digital system design for highly

sensitive applications. The tool can also be used for HT detection through the process of reverse engineering. The application of testability analysis for HT detection has been elaborated. It is found that introduction of HTs result in a change in testability parameters of certain nodes in a circuit. Subsequently suitable path retrace algorithm can be used to identify the location of malicious gates in the particular circuit.

REFERENCES

[1] Y. Shiyanovski, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer and W. Clay, "Process Reliability based Trojans through NBTI and HCI effects," in NASA/ESA Conference on Adaptive Hardware and Systems, 2010.

[2] X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar and B. S., "Sequential hardware Trojan: Side-channel aware design and placement," in Proc. of the IEEE 29th International Conference on Computer Design, 2011.

[3] G. T. Becker, A. Lakshminarasimhan, S. S. L. Lang, V. Suresh and W. Burelson, "Implementing hardware Trojans: experiences from a hardware Trojan challenge," in Proc. of the IEEE 29th International Conference on Computer Design, 2011.

[4] X. Kan, D. Forte and M. Tehranipoor, "A Novel Built-in Authentication Technique to prevent inserting Hardware Trojans," IEEE Transactions on Integrated Circuits and Systems, vol. 33, no. 12, pp. 1778-1791, 2014.

[5] M. Hicks, M. Finnicum, S. T. King, M. M. Martin and J. M. Smith., "Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically.," in IEEE Symposium on Security and Privacy, 2010.

[6] A. Waksman, M. Suozz and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis," in ACM SIGSAC conference on Computer & communications security, 2013.

[7] T. Hoque, S. Narasimhan, X. Wang, S. Mal-Sarkar and S. Bhunia, "Golden-free hardware Trojan detection with high sensitivity under process noise," Journal of Electronic Testing, vol. 33, no. 1, pp. 107-124, 2017.

[8] H. Salmani, M. Tehranipoor and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 1, pp. 112-125, 2012.

[9] B. Zhou, W. Zhang, S. Thambipillai, J. T. K. Jin, V. Chaturvedi and T. Luo, "Cost-efficient acceleration of hardware trojan detection through fan-out cone analysis and weighted random pattern technique," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 5, pp. 792-805, 2016.

[10] S. Dupuis, M.-L. Flottes, G. D. Natale and B. Rouzeyre, "Protection Against Hardware Trojans With Logic Testing: Proposed Solutions and Challenges Ahead," IEEE Design & Test 35, vol. 35, no. 2, pp. 73-90, 2018.

[11] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in Cryptographic Hardware and Embedded Systems-CHES 2009, Switzerland, 2009.

[12] S. Saha, R. S. Chakraborty, S. S. Nuthakki and D. & Mukhopadhyay, "Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability," in Saha, S., Chakraborty, R. S., Nuthakki, S. S., & Mukhopadhyay, D. (2015, September). Improved test pattern generation for hardware trojan detection using gen Cryptographic Hardware and Embedded Systems, France, 2015.

[13] Q. Liu and H. Li, "Signal Word-Level Statistical Properties-based Activation Approach for Hardware Trojan Detection in DSP Circuits," IET Computers & Digital Techniques, 2018.

[14] X. Chen, L. Wang, Y. Wang, Y. Liu and H. Yang., "A general framework for hardware trojan detection in digital circuits by statistical learning algorithms," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 10, pp. 1633-1646, 2017.

[15] K. Hasegawa, S. Youhua, M. Yanagisawa and N. Togawa, "A Score-Based Classification Method for Identifying Hardware-Trojans at Gate-Level Netlists," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015.

[16] K. Hasegawa, M. Yanagisawa and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in IEEE International Symposium on Circuits and Systems (ISCAS), 2017.

[17] C. Bao, D. Forte and A. Srivastava, "On application of one-class SVM to reverse engineering-based hardware Trojan detection," in 15th International Symposium on Quality Electronic Design (ISQED), 2014.

[18] M. Tehranipoor, H. Salami and X. Zhang, "Design Vulnerability Analysis," in Integrated Circuit Authentication, Springer International Publishing, 2014, pp. 125-145.

[19] R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in IEEE International Conference on High Level Design Validation and Test Workshop, 2009.

[20] D.Mukhopadhyay and R.S.Chakraborty, "Overview of Hardware Trojans," in Hardware Security- Design, Threats and Safeguards, Taylor and Francis Group, 2015, pp. 382-396.

[21] G. K. Contreras, A. Nahiyan, S. Bhunia, D. Forte and M. Tehranipoor, "Security vulnerability analysis of design-for-test exploits for asset protection in SoCs," in 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Tokyo, 2017.

[22] L-T.Wan, C-W.Wu and X.Wen, VLSI Test Principles and Architectures Design For Testability, Elsevier, 2006.

[23] V.D.Agarwal and M.L.Bushnell, "Testability Measures," in Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI circuits, New York, kluwer Publishers, 2002, pp. 129-150.

[24] F. Brglez, P. Pownall and R. Hum, "Applications of testability analysis: From ATPG to critical delay path tracing," in nternational test conference on The three faces of test: design, characterization, production, 1984.

[25] T. Lee, W. Wolf, N. Jha and J. Acken, "Behavioral synthesis for easy testability in data path allocation.," in International Conference on Computer Design: VLSI in Computers and Processors, 1992.

[26] S. Boubezari, E. Cerny, B. Kaminska and B. Nadeau-Dostie, "Testability analysis and test-point insertion in RTL VHDL specifications for scan-based BIST," IEEE transactions on computer-aided design of integrated circuits and systems, vol. 18, no. 9, pp. 1327-1340, 1999.