



Towards the Detection of Hardware Trojans with Cost Effective Test Vectors using Genetic Algorithm

Sandip Chakraborty^{1,2} · Archisman Ghosh¹ · Anindan Mondal^{1,3} · Bibhash Sen¹

Received: 9 January 2024 / Accepted: 3 June 2024 / Published online: 25 June 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract

Hardware Trojans (HT) are tiny circuits designed to exploit electronic devices, posing risks such as device malfunction or leakage of sensitive information. The adversary aims to implant these HTs specifically targeting nets with minimal signal transition (rare gates) within a circuit, evading detection during functional tests. Some Trojan variants are activated by adversaries under specific periodic conditions. Logic testing, a well-established method for test generation in HT detection, faces challenges due to the impractical scale of the search space, whereas Genetic Algorithms (GA) excel in efficiently navigating extensive solution spaces. This paper presents a GA-based technique that integrates information on effective inputs, along with an adequate fitness function defined based on combinational controllability and structural features, for detecting conditionally triggered ultrasmall HTs. Upon assessing the ITC 99 and ISCAS 85 and 89 benchmarks, we note significant enhancements in trigger coverage and reduced run-time requirements in comparison to state-of-the-art methods like MERO and TRIAGE.

Keywords Hardware trojan · Transition probability · SCOAP measurements · Genetic algorithm

1 Introduction

IC design houses often outsource the manufacturing process to reduce cost and speed up development time [26]. However, these offshore facilities introduce risks such as Counterfeit parts, Intellectual property theft, and hardware Trojans (HTs) [7]. While piracy is one of the biggest

challenges from a financial point of view, the presence of an HT circuit inside a design remains the biggest security threat. HTs pose a significant threat to device security due to their stealthy nature and destructive capabilities [4]. Based on the activation criteria, HTs can be broadly classified into two types, always on and conditionally activated [12]. While the former type is mainly used for information leakage, the latter one remains inactive most of the time and can be used for various malicious activities. These HT circuits are a huge security challenge due to the extreme difficulty of activating and detecting them during testing.

The difficulty with conditionally triggered HTs lies in their ability to elude detection during functional tests. To build such circuits, the adversary targets specific nets inside the netlist that do not exhibit many activities during regular operation. This ensures the HT avoids accidental activation during the test period. Traditional test methods have been found to be insufficient against such stealthy and extremely tiny HTs. As a result, several alternative countermeasures have been proposed in the literature [15]. Among them, logic testing has gained significant attraction due to its effectiveness against small HTs. This approach involves probing the input–output behaviour of a circuit and identifying inconsistencies between its expected and

Responsible Editor: U. Guin

✉ Bibhash Sen
bibhash.sen@cse.nitdgp.ac.in

Sandip Chakraborty
sandipch240@gmail.com

Archisman Ghosh
archiribhu@gmail.com

Anindan Mondal
anindanmondal14@gmail.com

¹ Department of Computer Science and Engineering, National Institute of Technology, Durgapur, India

² Department of Information Technology, Dr. B. C. Roy Engineering College, Durgapur, India

³ Department of Computer Science and Engineering, Asansol Engineering College, Asansol, India

observed behavior. By exploiting these inconsistencies, logic testing can effectively detect HTs that remain dormant most of the time [18].

The Logic testing method uses a set of test vectors to evaluate a circuit, with differences between expected and observed outputs as indicators of potential hard-to-reach faults or HTs. Despite its potential, a vast search space can make logic testing impractical. However, given the difficulty of the problem, these approaches often rely on estimating individual net activity and setting arbitrary threshold values to identify rare HTs. Estimation of net activity is achieved through either simulation of random vectors or probabilistic measurements including metrics such as transition probability and SCOAP (Sandia Controllability/Observability Analysis Programme) measurement. While applying an entire test suite may be infeasible due to many input combinations, these methods can potentially improve the efficacy of logic testing, even for large circuits.

SCOAP measurements are used to estimate the activity of individual nets and identify hard-to-detect faults in a circuit. Analysis of the controllability values of individual nets can help identify rare nets that may be used as triggers for the HT circuit. However, applying arbitrary test patterns to detect such HTs. In this regard, genetic algorithms (GA) can produce a collection of test vectors that increases the possibility of HT detection. In the GA-based technique, the input space is searched employing a population of potential test vectors. The capability to activate hard-to-trigger HTs evaluates each candidate test vector. The most successful candidates are selected to produce a new generation of test vectors, and the process is repeated. This technique is continued until a collection of test vectors that successfully detect the Trojan has been identified. The proposed select input utilizes a subset of critical inputs and accelerates GA convergence while targeting rare net triggers. Although only combinational trigger samples are considered in our experiments, they are covered multiple times, increasing the likelihood of activating a complex Hardware Trojan (HT) circuit even if constructed by an adversary. A conference version is also available at [3], which discusses these observations. However, the current study provides the following improvements -

- a more suitable fitness function for GA and consideration for level-wise SCOAP value distribution.
- detailed analysis on the impact of select input regarding GA performance.
- a much larger comparison against hard to trigger Trojan circuits of varying size.

In light of the above discussion, we have adopted the following strategies to generate a set of effective test vectors for HT detection:

- A genetic algorithm (GA) based approach is proposed that operates only on selected inputs (based on their impact on rare gates).
- A suitable fitness function is defined based on combinational controllability as well as structural characteristics.
- Tested on benchmarks like ITC 99 and ISCAS 85 and 89 while providing evidence on the trigger and rare gate coverage.
- The results are compared with state-of-the-art methods such as MERO [3] and TRIAGE [19].

The paper is structured as follows: In Sect. 2, we introduce preliminary concepts essential to our work. Section 3 discusses the motivation behind our presented work and reviews related works. The detail of the proposed work is illustrated in Sect. 4, while Sect. 5 presents the experimental results. Finally, Sect. 6 concludes the paper.

2 Preliminaries

In this Section, we describe the basic terminologies used throughout in our work.

2.1 Hardware Trojan

It is widely considered that there are two primary components of an HT circuit - Trigger and Payload. The Trigger initiates the HT circuit, while the Payload causes the circuit to malfunction when activated [17]. To evade detection during traditional manufacturing tests, adversaries attempt to implant the HT so that it remains dormant for most of the time. As a result, Triggers are commonly linked to nets (rare gates) that do not change their values during regular operation. Figure 1 presents an illustration of an HT as an example.

2.2 Combinational Controllability

The measurement of Combinational Controllability (CC) serves as a cost function that quantifies the difficulty level involved in setting a signal from the primary input. This algorithm has a linear time complexity and is an integral component of the widely recognized SCOAP measurements [6]. CC measurement assesses the amount of effort needed to set a specific signal to either logic 0 or 1. Table 1 displays the SCOAP Combinational Controllability calculation guidelines for basic gates.

SCOAP calculation for a full subtractor circuit is shown in Fig. 2.

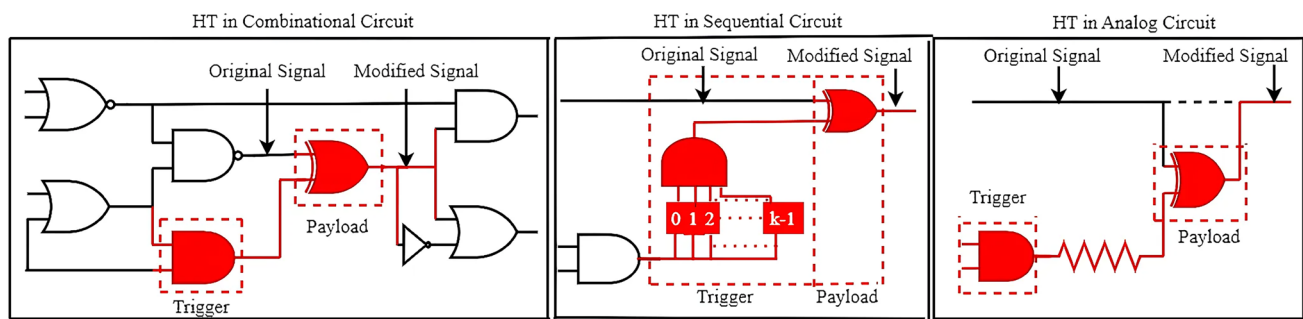


Fig. 1 Demonstration of Hardware Trojan Insertion in Circuit Netlists: (i) Combinational Circuit Design, (ii) Sequential Circuit Design, and (iii) Analog Circuit Design

Table 1 Formulas for the Controllability of different Logic Gates [8]

Logic Gate Category	0 Controllability	1 Controllability
NOT	input $CC1 + 1$	input $CC0 + 1$
OR	$\sum \{\text{input } CC0\} + 1$	$\min \{\text{input } CC1\} + 1$
AND	$\min \{\text{input } CC0\} + 1$	$\sum \{\text{input } CC1\} + 1$

2.3 Genetic Algorithm

The Genetic Algorithm is an evolutionary algorithm inspired by biology and natural selection theory, commonly called "survival of the fittest." It attempts to optimize search problems by creating an initial population representing the solution domain's genetic composition. Various operators like selection, crossover, and mutation develop a new set of adaptive solutions by assessing the initial population's fitness. Over successive generations, the solution domain is steadily reduced until an optimum solution for this problem statement is reached.

Because of the Genetic Algorithm's inbuilt parallelism, it can effectively traverse the search space, making it a great candidate for test generation challenges. Figure 3 visually represents the basic working principle of the Genetic Algorithm.

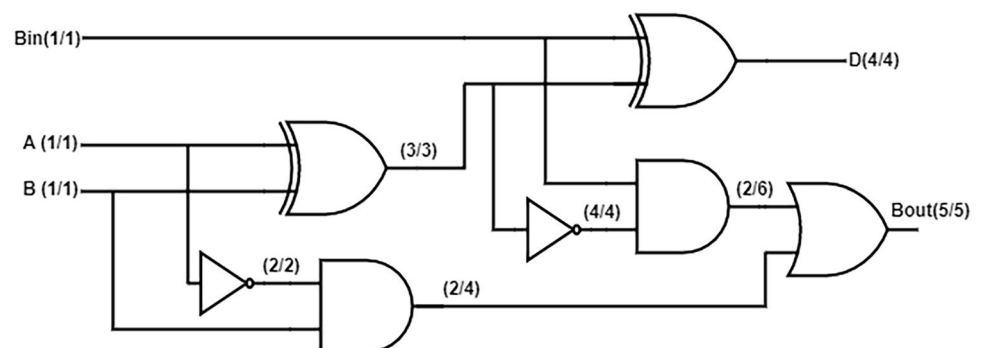
3 Background

The escalating threat of Hardware Trojan (HT) in the domain of hardware security has prompted the development of various detection techniques as reported in the literature. This section is dedicated to a comprehensive review of prior works closely aligned with our methodology. The subsequent segment of this section illuminates the motivation behind our research efforts.

3.1 Related Work

Conventional testing strategies are insufficient for combating hardware Trojans (HT). Karmian et al. [10] proposed a hardware Trojan detection method using Ring Oscillator Network (RON) measurements and a genetic algorithm. Despite high accuracy in classifying trojan-inserted circuits, the method faces a drawback of increased processing time compared to PCA-based approaches without SVM training data. Bao et al. introduced a reverse engineering-based hardware Trojan detection technique [1] utilizing k-means clustering on feature vectors extracted from grid images. The method achieves high accuracy in detecting trojan insertions and enables fine-grained chip classification. However, its sensitivity to the choice of initial centroids and vulnerability to advanced hardware Trojans are notable limitations.

Fig. 2 Controllability Analysis of Gates within a Full Subtractor



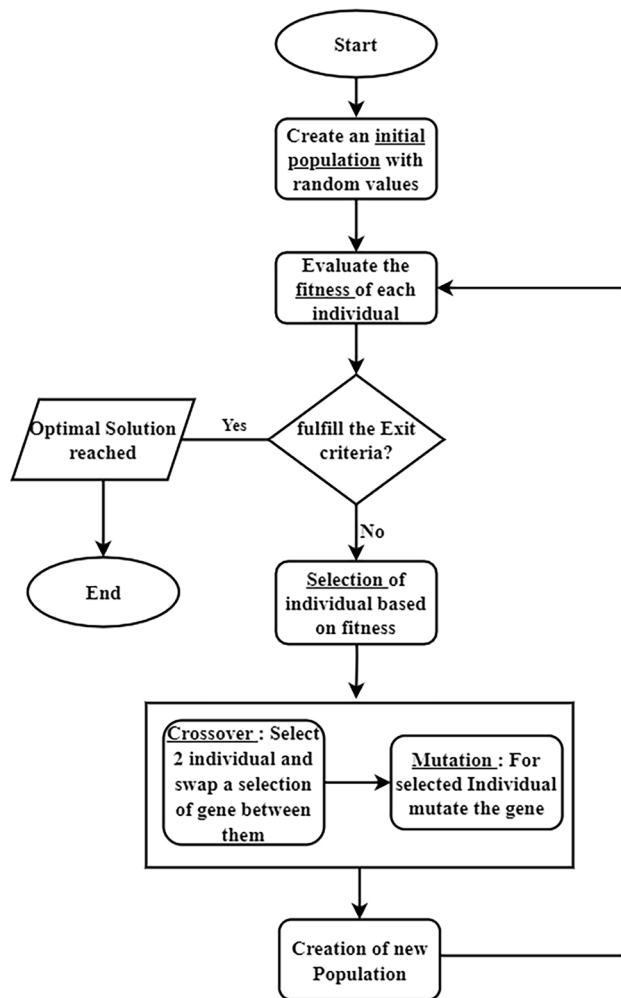


Fig. 3 A flowchart depicting the sequential stages of a Genetic Algorithm. Further elaboration on this process can be found in Section IV

Masaru Oya proposed a hardware Trojan identification method [20] based on Trojan net scoring in gate-level netlists. By detecting max score nets and assigning three Trojan factors, this method successfully identifies hardware Trojans in various gate-level benchmark circuits. Bazzazi et al. proposed one of the methods for hardware detection based on logic testing [2], which extracts the nodes with unique attributes and selects the nodes with the most significant similarity in terms of logical values as targets. The potential trojans are detected from logical values that differ for Trojan-free and Trojan-inserted nodes. This method suffers from significant overhead and power consumption. Vidya Govindan presents another logic testing-based HT detection technique [3], which uses the MERO algorithm approach through which a high trigger coverage is achieved in both combinational and sequential Trojans and has a high

probability. The limitation of this method is its poor coverage on hard-to-detect Trojans and considers only activation of triggering conditions. Sayandeep Saha suggests utilizing a genetic algorithm and Boolean satisfiability to enhance test pattern generation. Saha et al. [22] to detect hardware Trojans in the circuit, where they use a genetic algorithm to detect the maximum number of triggers and store them in a hash table. SAT tool is used to generate test combinations that are not covered by the genetic algorithm and find feasible Trojan sets. This method has more coverage than previously proposed ATPG heuristics, but even after using the SAT tool, some trigger combinations remained unsolved. These drawbacks were resolved in a method by M.A.Nourian that detects HT using genetic algorithm-based logic testing [19] where they use the TRIAGE algorithm to generate optimal test vectors and identify the rare nodes based on transition probability. Cruz et al. [5] employ ATPG and model checking for HT detection but may not effectively uncover less generic Trojans. Mondal et al. proposed a novel test vector generation method for hardware Trojan detection [17] where the final test vectors are optimized based on the toggle effect and bit toggle probability values. Vectors from the proposed method achieve a higher number of nets compared to random vectors.

MaxSense, a hardware Trojan detection method by Lyu et al. [13], utilizes an SMT solver and genetic algorithm to enhance side-channel sensitivity and time complexity. However, its applicability is influenced by hardware characteristics, scalability challenges with large circuits, sensitivity to Trojan variability, potential resource requirements, and robustness issues in the presence of noise or environmental variations. However, this algorithm aims to generate tests to measure hardware switching activity, making it unsuitable for a direct comparison. Shi et al. [24] also proposed a hardware Trojan detection method based on correlation analysis between circuit inputs and inactive gates. However, similar to [13], this method also focuses on test set re-ordering for side channel measurement, rather than Trojan coverage. Pan et al. [21] recently introduced reinforcement learning for test generation. However, reinforcement learning requires substantial environmental interactions. Moreover, it is computation heavy [14], which is not ideal for scalability. The considerable computational overhead inherent in these methodologies persists as a formidable impediment to their broad scale adoption.

3.2 Motivation

In the literature, we have presented various approaches for detecting hardware Trojans, each with its own set of advantages and drawbacks. Many of these methods face

disadvantages, such as increased processing time, sensitivity to Trojan variability, scalability issues, and limitations in trigger efficiency. The key motivation of our proposed methodology is to overcome these limitations and increase the effectiveness of hardware Trojan detection using the following strategies: Advanced Genetic Algorithm, Selective Input Processing, and Fitness Function Incorporating Structural Features, Benchmark Testing. The focus on efficiency, selectivity, and benchmark validation ensures the practical applicability and effectiveness of the proposed approach in real-world scenarios, providing a valuable contribution to the field of hardware security.

4 Proposed Work

In this section, our GA based methodology is described. The fitness function is defined using a SCOAP based metric which also incorporates the distance of the rare gates from the primary inputs. We identify primary inputs that contribute most towards generating a transition on rare nets which allows us to optimize the performance of the GA. Detailed description of the proposed technique is provided next.

4.1 HT Model

Hardware Trojans are considered to have two parts - trigger and payload. The payload gets activated when the triggering condition is satisfied. In our work, we have considered the presence of combinational Trojans, having a 2-input AND gate as the trigger. The HT circuit is activated when the rare value of the AND gate (logic 1) is generated by an input test vector.

4.2 Circuit Representation

The circuit netlist is expressed as a directed graph denoted by $G = \{V, E\}$ where V represents the collection of gates

(vertices), and E represents the ensemble of nets (edges). This facilitates the use of graph traversal algorithms like Breadth First Search (BFS) on the circuit netlist. This allows us to identify the primary inputs that are linked to the rare gates. We have considered circuits from ISCAS '85, ISCAS '89, and ITC '99 benchmarks to validate the proposed methodology.

4.3 Rare Net Selection





A net (or wire) is defined as a rare net if it fails to produce many transitions during normal operation. Such nets can be identified by simulating the circuit using random test vectors and observing the transitions. The following Table shows the rare values generated from different logic gates (Fig. 4).

In our experiment, we have simulated each circuit with 1000 random test vectors and set the threshold as 0.2, and 0.1 respectively. These values have also been used previously in [23] for hard-to-trigger Trojans. We created a set R comprising rare gates, which is used as input in Algorithm 1.

4.4 Primary Input Selection

It has been observed in [16], that only a handful of primary inputs in a circuit primarily influence the triggering of rare gates. Therefore, to reduce the overhead of the GA, we use the selected set of inputs (*effective_input*) instead of the entire input space. We modify the standard Breadth First Search (BFS) algorithm for the circuit netlist (Algorithm 1). Our algorithm takes the netlist (as a directed graph), the selected inputs, and the rare gates as inputs and returns 1 if a path exists between the select input (*in*) and the rare gate (*rg*) (0 otherwise).

Fig. 4 Rare values of different Gates

Gate	Truth Table			Rare Value	Gate	Truth Table			Rare Value
	I/P		O/P			I/P		O/P	
	0	0	0	0		0	0	0	1
	0	1	1			0	1	0	
	1	0	1			1	0	0	
	1	1	1			1	1	1	
	0	0	1	1		0	0	1	0
	0	1	0			0	1	1	
	1	0	0			1	0	1	
	1	1	0			1	1	0	

Algorithm 1 ImprovedBFS

```

1: procedure IMPROVEDBFS( $G, in, rg$ )
2:    $Queue \leftarrow InitializeQueue$ 
3:    $traversed \leftarrow Emptyset$ 
4:    $traversed \leftarrow in$ 
5:    $Queue.enqueue(in)$ 
6:   while not  $Queue.isEmpty()$  do
7:      $cv \leftarrow Dequeue(Queue)$ 
8:     for each  $nv \in G.neighbors(cv)$  do
9:       if not  $traversed$  then
10:         $traversed \leftarrow nv$ 
11:         $Queue.enqueue(nv)$ 
12:        if  $nv$  equals  $rg$  then
13:          return 1
14:        end if
15:      end if
16:    end for
17:  end while
18:  return 0
19: end procedure

```

Algorithm 2 is iterated for all the rare gates(R) and inputs(IN) to determine the set of *effective_inputs* to be used in the GA.

Algorithm 2 EffectiveInput

```

1: procedure EFFECTIVE( $G, R, IN$ )
2:    $effective\_input \leftarrow null$ 
3:   for every  $rg \in R$  do
4:     for every  $inputline\ in \in IN$  do
5:       if  $ImprovedBFS(G, in, rg)$  then
6:          $effective\_input \leftarrow in$ 
7:       end if
8:     end for
9:   end for
10:  return  $effective\_input$ 
11: end procedure

```

An Illustrative Example:

To showcase the concept of *effective_input*, we utilize the smallest benchmark circuit from ISCAS 89 (s27). As an example, in Fig. 5, a NAND gate (gate No. G9) is shown in the color red to represent its rarity. It is directly controlled by G16 and G15. These two gates are further controlled by B1, B2, B3, B4, and B5, which are connected to the Red-colored and Blue-colored nets originating from the primary inputs. On the other hand, input B0 and B6 are considered as

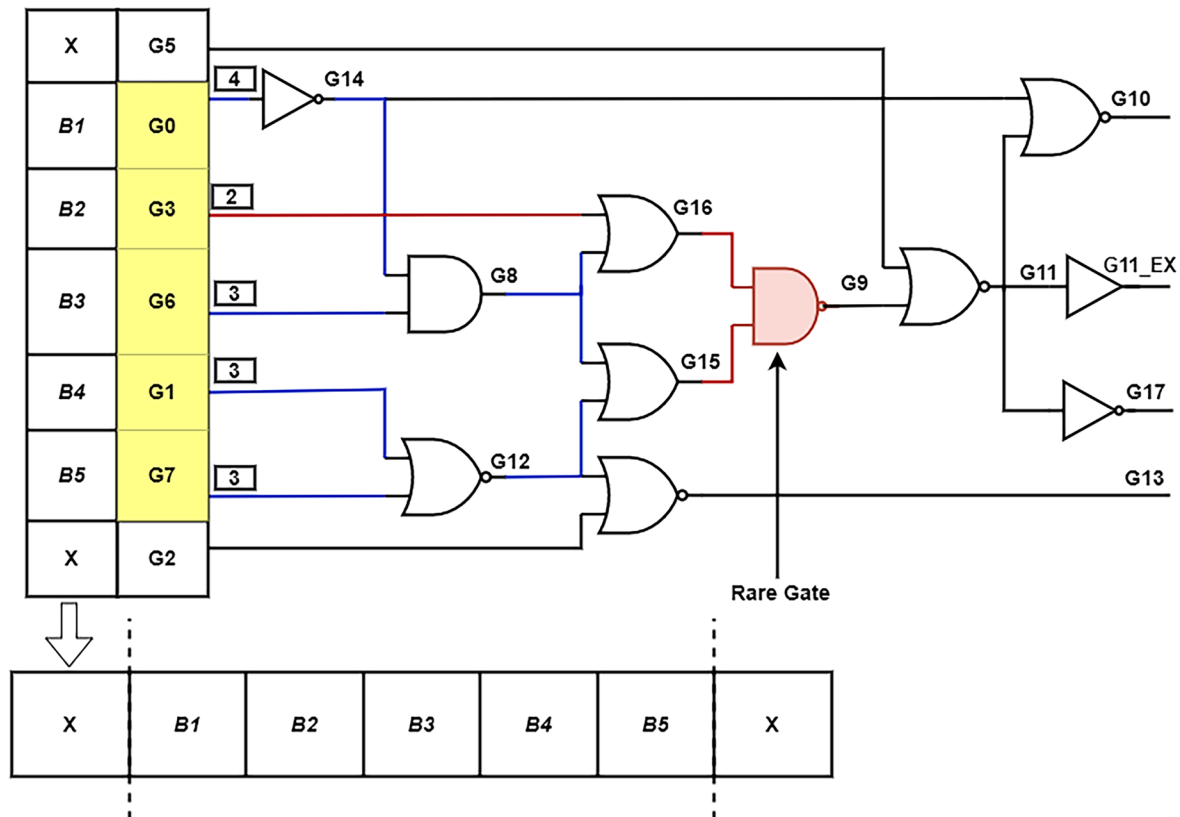


Fig. 5 Illustration using the s27 circuit from the ISCAS'89 benchmark, demonstrating our concept of *effective_input*. B1, B2, B3, B4, and B5 are identified as *effective_input*, and the GA is executed using the *effective_inputs*, mitigating computation overhead compared to the entire input set

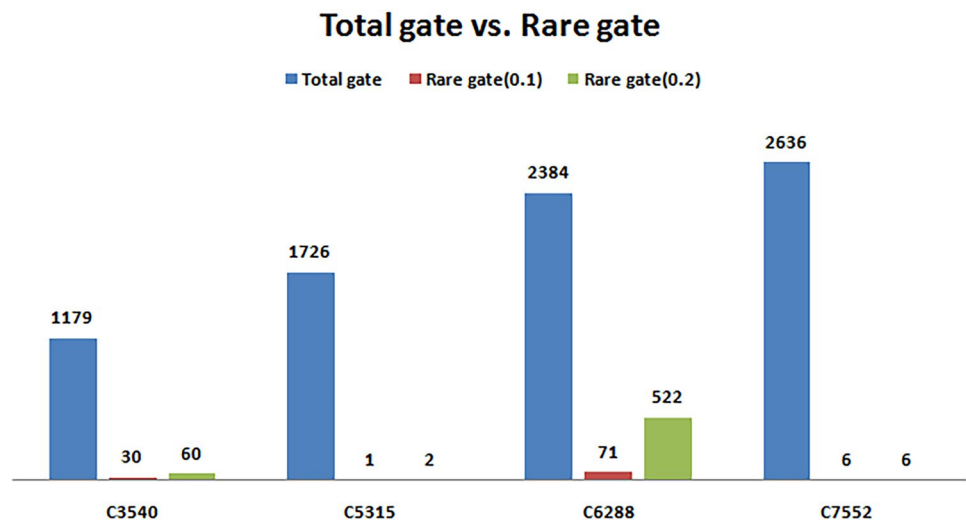
Table 2 Abbreviations and Descriptions Used in the Algorithms

Abbreviations used in Algorithms	Descriptions
<i>in</i>	Select input
<i>rg</i>	Rare Gate
<i>G</i>	Directed Graph
<i>R</i>	List of all Rare Gates
<i>EI</i>	List of Effective inputs
<i>IN</i>	List of all Inputs
<i>P</i>	Population(list of Chromosomes)
<i>FI</i>	list of fittest individuals

don't care (X) bits as they have no impact on triggering the NAND gate $G9$. The GA chromosomes consists of input bit $B1, B2, B3, B4, B5$ only, rather than the complete input set.

The introduction of select input helps us to reduce the run time while maintaining the quality of the produced vectors. The rationale behind this is number of such select inputs (say s) is much smaller than the total input (say $i, s \ll i$), which helps in a faster convergence. The number of rare gates is directly dependent on the transition probability. Since the Trojan circuits are expected to have a small transition probability, we specifically target low-probability nets. An example of such rare nets found in ISCAS 85 circuits is shown in Fig. 6 which shows that no of rare gate is much smaller than total gate. This implies that size of the select input is also expected to be quite small compared to input size.

Fig. 6 Total no of gate vs. rare gate for threshold value 0.1 and 0.2



4.5 Genetic Algorithm

As stated already, Genetic Algorithm (GA) is used to generate input test vectors for a given circuit netlist. It is a heuristic search algorithm that aims to optimize searching by reducing the search space. GA can be easily mapped to the problem of test generation. However, with increasing size of the circuit, it also suffers from high run time. To mitigate this issue, we utilize the proposed *effective_input(EI)* which allows faster computation. Algorithm 3 outlines the steps involved in our proposed GA-driven approach.

Algorithm 3 Genetic Algorithm

```

1: procedure GA(circuitNetlist,  $R, EI, IN$ )
2:   Set  $n$  as the length of  $EI$ 
3:   Set  $k$  as the length of  $IN$ 
4:   Initialize GA with diverse population  $P$  (1000 vectors of size  $n$ )
5:   for each  $i$  ranging from 1 to 100 do
6:     Determine the fitness of each chromosome in  $P$  using a
       pseudo-vector of size  $k$  comprising  $n$  bits and  $(k - n)$  don't care
       bits.
7:     Arrange the population  $P$  in descending order based on their
       fitness values.
8:     randomly select parents from the top 10% and bottom 90% of
       the  $i$ -th generation.
9:     Perform two-point crossover to generate offspring.
10:    Apply mutation based on the mutation rate.
11:    Evaluate fitness for offspring in the  $(i + 1)$ -th generation.
12:    Fittest individuals of the generation are stored in  $FI$ 
13:  end for
14:  return  $FI$ 
15: end procedure

```

The Table 2 provides a comprehensive list of abbreviations utilized in the algorithms, accompanied by their respective descriptions.

4.5.1 Initial Parameters

We start with an initial population of 1000 random test vectors of individual length, similar to the number of *effective_input* of a circuit netlist.

4.5.2 Fitness

To achieve optimal results in GA, a fitness function is utilized that incorporates SCOAP values and the distance of rare gates from the input. We define a metric *div* which is the ratio of *CC0* and *CC1* values for a logic gate. This ratio of controllability values of logic 0 and logic 1 indicates the probable activity of a net (since it indicates the imbalance between two values [9]).

$$\text{div} = \frac{\max(\text{CC0}, \text{CC1})}{\min(\text{CC0}, \text{CC1})} \quad (1)$$

A high *div* value for a logic gate indicates that it may be easily manipulated to produce incorrect output and hence, is a potential HT trigger. We define the fitness function for the GA as the product of *div* and the reciprocal of the shortest distance between inputs and rare gates (*dist*). *dist* is used as a weight factor since SCOAP values tend to increase as level increases.

$$\therefore \text{fitness} = \text{div} \times \frac{1}{\text{dist}} \quad (2)$$

We illustrate the idea in Fig. 5. For the rare gate, *G9* is affected by 5 input lines out of the total 7 and we consider *dist* as the distance between *G3* and *G9* as it is the lowest.

4.5.3 Selection

The process of selecting vectors for further breeding is determined by their fitness. Each vector is given a fitness value, and the population is subsequently sorted in descending order based on these values. From there, one vector is randomly selected from the top 10% of the population (also known as the elite), while another vector is chosen from the remaining 90% of the population. These two selected vectors are then subjected to a two-point crossover, which allows for the exchange of genetic information between them, ultimately resulting in a new generation of vectors with potentially improved fitness.

4.5.4 Crossover

To generate the next generation of vectors in our proposed algorithm (Fig. 7), we employed a two-point crossover between two parent vectors. The process involves selecting *n* effective bits (i.e., *effective_input*) from the total *k* bits in both parent vectors, with the remaining (*k* − *n*) input bit positions considered as "don't care". Subsequently, two crossover points are randomly chosen, and the crossover process

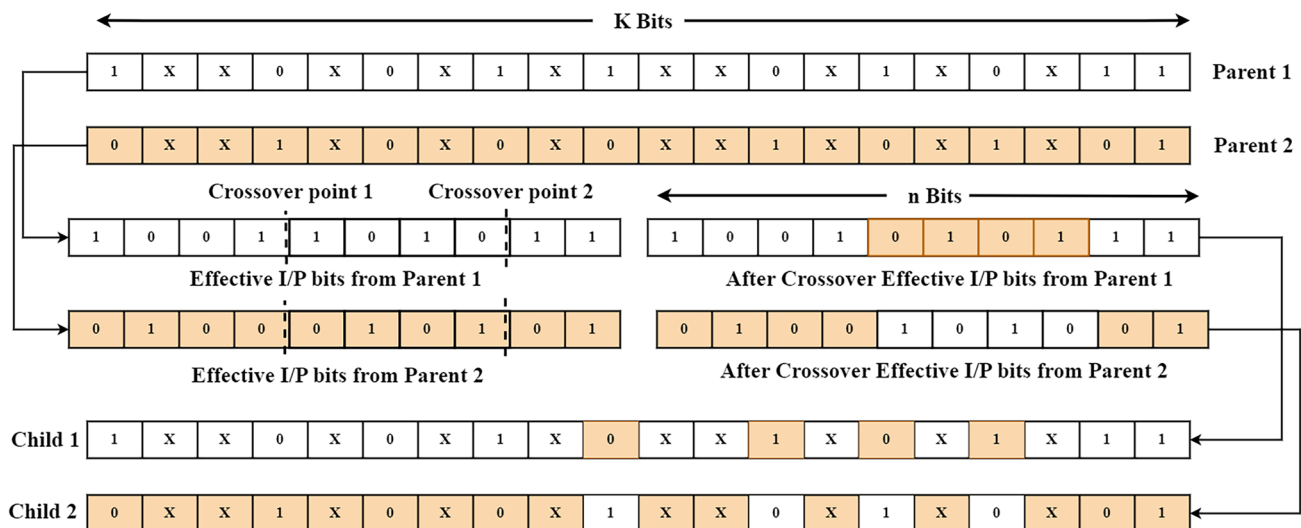


Fig. 7 The presented diagram depicts the technique employed in our proposed algorithm to execute a two-point crossover. The crossover rate used is 0.8

is performed exclusively on the selected n bit positions. The resulting offspring vectors are then formed by combining the modified n bits with the remaining $(k - n)$ bits.

4.5.5 Mutation

Mutation is an essential operator in the genetic algorithm, as it enables the exploration of the search space locally, which helps to avoid getting trapped in a local minimum.

To implement mutation in our proposed approach, we set a mutation probability of 0.1. This means that for any given vector (with effective n bits), we randomly select and flip 10 percent of its bits. This approach helps to ensure that the algorithm maintains diversity and avoids convergence to a local minimum.

In the context of computational complexity, our proposed algorithm exhibits a time complexity of $O(np_g)$, where n is the size of the *effective_input* (chromosome size), p is the population size, and g is the number of generations (Fig. 8).

5 Experimental Results and Discussion

5.1 Experimental Setup

For comparative purposes, we implemented the widely recognized MERO [3] and TRIAGE [19] algorithms. All the algorithms were implemented in Python 3, utilizing an Intel Core i5 CPU operating at a frequency of 2.40 GHz. To extract controllability values from a netlist, we employed the open-source “Testability Measurement Tool” [25]. The netlists were obtained from openly accessible ISCAS benchmark circuits, formatted as Bench files and can be freely obtained from the following source: <http://www.pld.ttu.edu/~maksim/benchmarks/>. Out of the total of 12 circuits employed, we selected 5 circuits from the ISCAS’85, 4 circuits from the ISCAS’89, and 3 circuits from the ITC’99 benchmark suite. These benchmarks encompass diverse circuit instances representing various functionalities, from

arithmetic and logic units to processor subsets. We have also incorporated the b17 circuit from ITC 99, which consists of well over 100k gates

5.2 Rare Gate Coverage Performance

After implementing our proposed approach, a thorough comparison between our algorithm, TRIAGE and MERO is performed. Considering a rareness threshold of 0.1, we observed substantial improvements in rare gate coverage across different benchmark circuits. The details of the results are presented in detail in Table 3. The enhancements are visually depicted in Fig. 9. It is evident that we achieved improvements exceeding 80% in certain instances. It highlights the extent to which our fitness function helps outperform the alternatives, demonstrating the significant advancements made in enhancing rare gate coverage.

Similarly, we conducted the same experiment at a threshold of 0.2. Figure 10 visually presents the significant improvements achieved in rare gate coverage, with certain circuit instances showcasing improvements nearing 90%. To delve into further details and provide a comprehensive analysis, Table 4 is used to show a comprehensive comparison with TRIAGE and MERO. It is evident, that even in this threshold, our proposed algorithm outperforms both existing techniques.

5.3 Execution Time Related Performance

As stated already, considering a circuit with k inputs, it is important to note that it has 2^k input combinations. This exponential growth in the number of possible combinations emphasizes the impracticality of exhaustively searching the entire solution space. To overcome this challenge, our proposed method takes a different approach. Instead of exploring the entire solution space, we optimize the Genetic Algorithm (GA) to operate within a smaller, more focused search space by selecting a subset of the input space. By doing so, we significantly reduce the computational burden, which

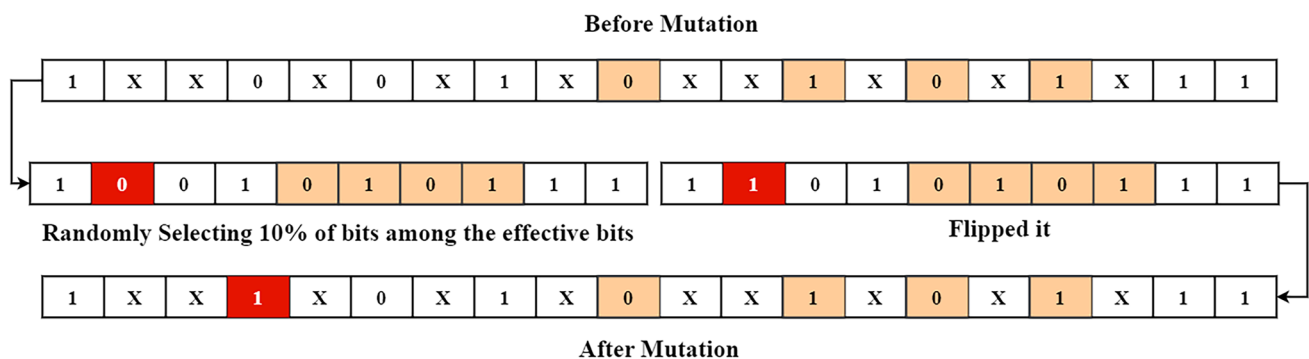


Fig. 8 The diagram illustrates the approach utilized in our proposed algorithm for executing mutations. A mutation rate of 0.1 is applied

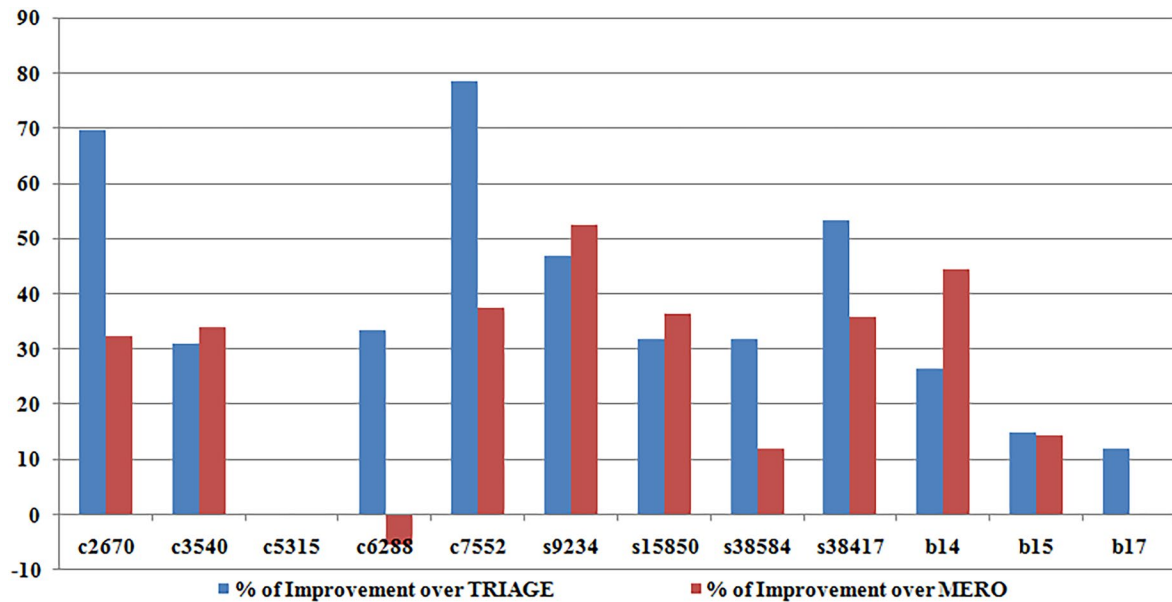


Fig. 9 Enhancement in rare gate coverage by the Proposed Algorithm Compared to MERO and TRIAGE (Rareness Threshold = 0.1)

effectively reduces the run time. Comparing the performance of our method with the TRIAGE and MERO algorithms, we observe a remarkable improvement in execution time.

We initiated the algorithm with a population of 1000 vectors, running it for 100 generations. We run both TRIAGE and MERO with similar parameters for comparison. The results, showcased in Figs. 11 and 12, demonstrates a noteworthy improvement of over 50 percent in runtime for

both threshold values (We have used dual Y-axis graphs in order to show the performance of 3 algorithms. Using the primary Y-axis we measure the performance of the Proposed algorithm and TRIAGE, whereas the secondary Y-axis is used for MERO.). Due to scaling limitations, direct time comparisons for the b17 circuit were not feasible in the graphs. However, for a 0.1 threshold value, GA and TRIAGE took 19763.74 and 43695.98 s, respectively. For

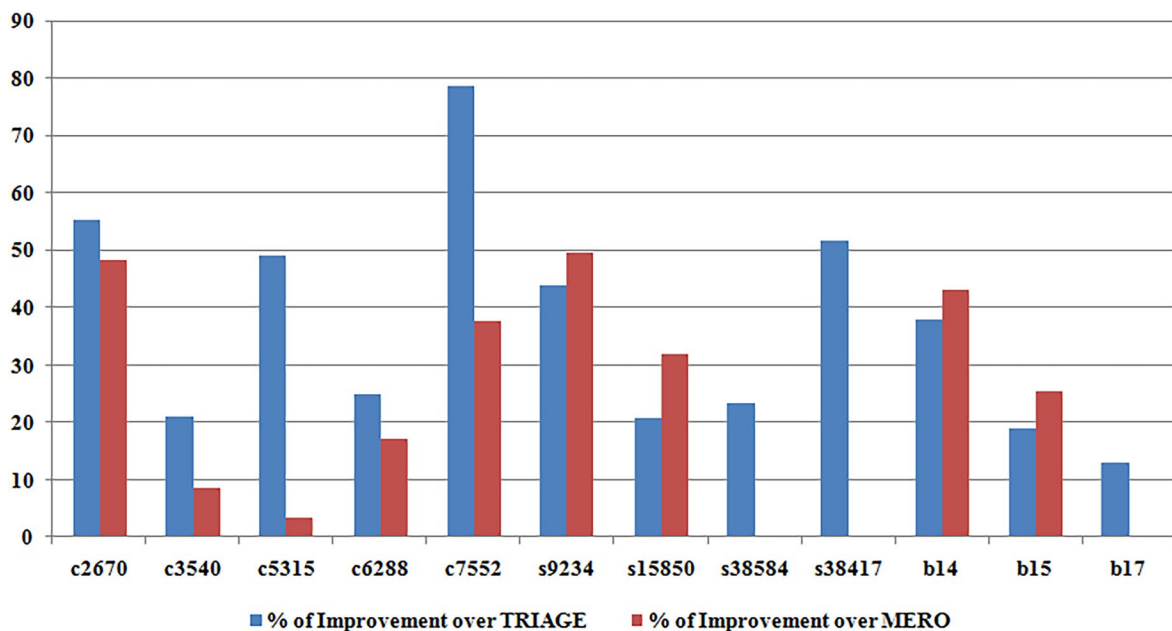


Fig. 10 Enhancement in rare gate coverage by the Proposed Algorithm Compared to MERO and TRIAGE (Rareness Threshold = 0.2)

Table 3 Comparison for Threshold 0.1

Ckt. Instance	Total Rare Gates	Proposed Algorithm		TRIAGE		MERO	
		Rare Gates Coverage	% of Rare Gate Coverage	Rare Gates Coverage	% of Rare Gate Coverage	Rare Gates Coverage	% of Rare Gate Coverage
c2670	4	4	100	1.22	30.5	2.71	67.75
c3540	30	20.13	67.1	10.85	36.16	9.97	33.23
c5315	1	1	100	1	100	1	100
c6288	71	32	45.07	8.35	11.76	35.96	50.64
c7552	6	5.94	99	1.23	20.5	3.69	61.5
s9234	47	35.99	76.57	14	29.78	11.28	24
s15850	123	61.23	49.78	22	17.88	16.52	13.43
s38584	100	52.95	52.95	21	21	41.05	41.05
s38417	28	22.5	80.35	7.6	27.14	12.48	44.57
b14	616	521.86	84.71	358.70	58.23	247.74	40.21
b15	220	101.89	46.31	69.15	31.43	70.05	31.84
b17	397	167.45	42.18	119.49	30.10	×	×

a 0.2 threshold value, GA and TRIAGE took 20502.14 and 45143.92 s, respectively.

5.4 Trigger Coverage Performance

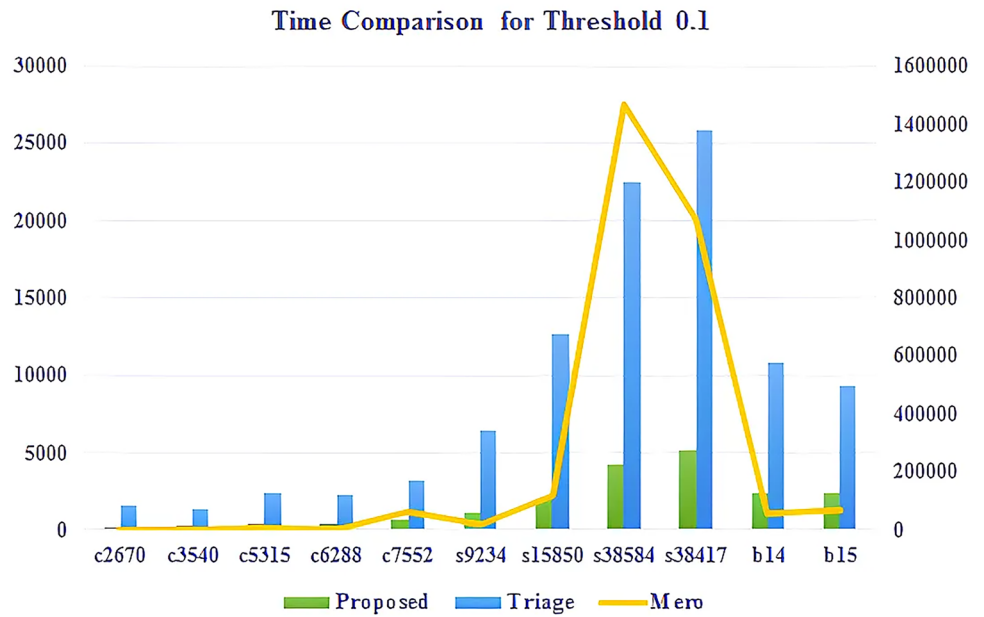
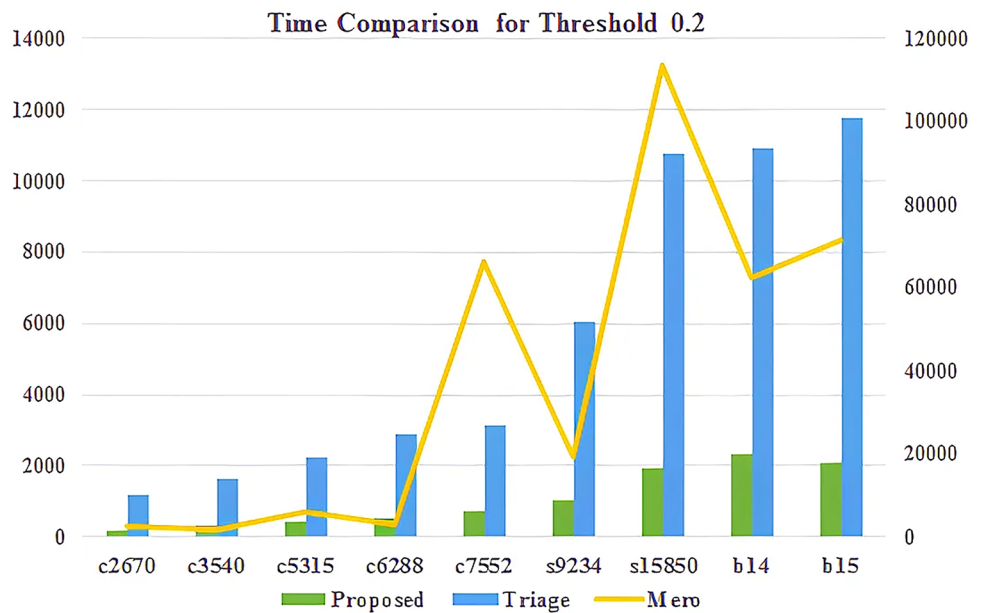
We investigate the presence of combinational HTs in the circuit, with a 2-input AND gate acting as the trigger. The triggers are inserted into the rare nets randomly, for both threshold values. We considered 100 triggers for each benchmark, which is used to measure the efficacy of the proposed method. The HT is triggered when the input test vector generates the rare value of the AND gate, i.e., when the logic value 1 is obtained at the trigger. Table 5 depicts the average

trigger coverage achieved by a vector compared with existing methods. A near 80% improvement on trigger coverage is observed for these circuits which is shown in Fig. 13. Such improvements make our proposed method more suitable for larger circuits since it allows to generate a more compact test suite.

For s38584 and s38417, we set a cut-off run time of 21 days for MERO, but it failed to produce any result (Threshold = 0.2). We also performed another experiment with a set of randomly generated Trojan triggers as per [27]. A total of 4950 2-input, 5456 3-input, and 4845 4-input AND triggers were generated. ATALANTA ATPG [11] generated test patterns were used to remove easy-to-trigger circuits. Comparative

Table 4 Comparison for Threshold 0.2

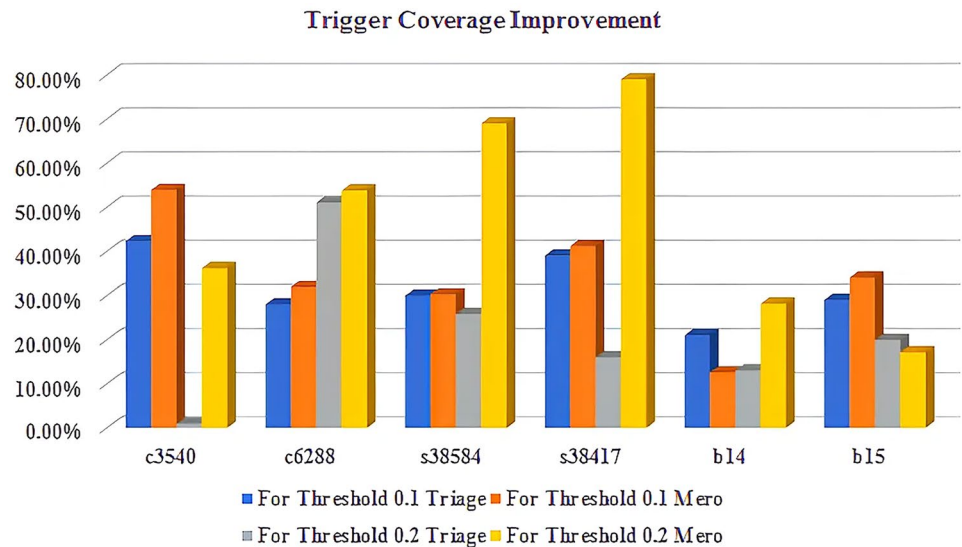
Ckt. Instance	Total Rare Gates	Proposed Algorithm		TRIAGE		MERO	
		Rare Gates Coverage	% of Rare Gate Coverage	Rare Gates Coverage	% of Rare Gate Coverage	Rare Gates Coverage	% of Rare Gate Coverage
c2670	7	6.86	98	3	42.85	3.48	49.71
c3540	60	33.95	56.58	21.40	35.66	28.8	48
c5315	2	2	100	1.02	51	1.93	96.5
c6288	522	291.13	55.77	161.75	30.98	201.62	38.62
c7552	6	5.94	99	1.23	20.5	3.69	61.5
s9234	55	41.95	76.27	17.83	32.41	14.7	26.72
s15850	153	73.88	48.28	42	27.45	24.91	16.28
s38584	335	210.65	62.88	132.08	39.42	×	×
s38417	44	39.6	90	16.9	38.40	×	×
b14	729	592.63	81.29	316.10	43.36	277.93	38.12
b15	284	152.91	53.84	99.33	34.97	81.01	28.52
b17	719	345.76	48.09	252.80	35.16	×	×

Fig. 11 Time comparison for threshold 0.1**Fig. 12** Time comparison for threshold 0.2**Table 5** Trigger Coverage Comparison

Ckt. Instance	Threshold 0.1			Threshold 0.2		
	Proposed Algorithm	TRIAGE	MERO	Proposed Algorithm	TRIAGE	MERO
c3540	70%	27.72%	16.06%	50%	49%	13.85%
c6288	42%	14%	10.09%	63%	12%	9.23%
s38584	58%	28%	27.68%	69%	43.19%	×
s38417	74%	35%	32.8%	79%	63%	×
b14	62%	41%	49.43%	73%	60%	44.83%
b15	66%	37%	31.95%	71%	51.05%	53.83%

Table 6 Multi-Input Hard-to-trigger Trigger Coverage

CKT. Instance	Number of uncovered triggers by ATALANTA Vectors	GA Vectors Covered	TRIAGE Vectors Covered	MERO Vectors Covered
C2670	14913	603	595	601
C3540	14375	72	46	21
C5315	13975	178	131	130
C6288	7174	675	266	320
C7552	13667	7	4	3

Fig. 13 Enhancement in Trigger Coverage by the Proposed Algorithm Compared to MERO and TRIAGE (Rareness Thresholds = 0.1 and 0.2)

assessment showed that test vectors generated by the proposed method are more effective in detecting and activating such dormant triggers as shown in Table 6. However, overall performance is still not satisfactory since the generated vectors target specific rare gates while the inserted triggers were sampled randomly from all over the circuits

6 Conclusion

The paper proposes a method of generating input test vectors for the detection of Hardware Trojans. The proposed algorithm identifies the rare gates in a circuit benchmark, selects inputs directly affecting the output of the rare gates, and performs a genetic algorithm on the selected inputs to identify test vectors to detect the presence of Hardware Trojans. Our proposed method outperforms state-of-the-art methods of test generation, viz., MERO and TRIAGE in terms of rare gate coverage ($\approx 72\%$) which is 2.3 times better than TRIAGE and 1.6 times better than MERO. The improvement of our proposed algorithm is

also reflected by the increased trigger coverage for hard-to-trigger Trojans (threshold = 0.1). Our proposed algorithm covers $\approx 62\%$ of the triggers whereas TRIAGE and MERO manage only $\approx 27\%$. Moreover, our proposed algorithm generates better input test vectors in less than 20% of the time as compared to TRIAGE. The low run time of our proposed algorithm makes it suitable for larger circuits as is shown by the ISCAS 89 and ITC 99 benchmark circuits. Despite its efficacy, logic testing for HT detection faces significant limitations due to the unbounded number of potential hardware Trojans. This underscores the need for advanced methodologies and further research to enhance detection precision and reliability. In the future, we would like to extend our work to analyze rare gate-switching activity used in side-channel measurement.

Funding Additionally, they have stated that no funding was received for this research.

Data Availability The trigger-inserted data sets generated or analyzed during the course of this study are accessible upon reasonable request from the corresponding author.

Declarations

Conflicts of Interests The authors declare that they have no conflicts of interest

References

- Bao C, Forte D, Srivastava A (2015) On reverse engineering-based hardware trojan detection. *IEEE Trans Comput Aided Des Integr Circuits Syst* 35(1):49–57
- Bazzazi A, ManzuriShalmani MT, Hemmatyar AMA (2017) Hardware trojan detection based on logical testing. *J Electron Test* 33:381–395
- Chakraborty RS, Wolff F, Paul S, Papachristou C, Bhunia S (2009) Mero: a statistical approach for hardware trojan detection. In: *Proc of cryptographic hardware and embedded systems-CHES 2009: 11th international workshop Lausanne, Switzerland, September 6-9, 2009 Proceedings, CHES'09*. Springer, pp 396–410
- Chakraborty S, Ghosh A, Mondal A, Sen B (2022) Test pattern generation for detection of hardware trojans based on improved genetic algorithm. In: *Proc of IEEE Bombay section signature conference (IBSSC), IBSSC'22*. IEEE, pp 1–6
- Cruz J, Farahmandi F, Ahmed A, Mishra P (2018) Hardware trojan detection using atpg and model checking. In: *Proc of 2018 31st international conference on VLSI design and 2018 17th international conference on embedded systems (VLSID), VLSID'18*. IEEE, pp. 91–96
- Goldstein L (1979) Controllability/observability analysis of digital circuits. *IEEE Trans Circuits Syst* 26(9):685–693
- Hu W, Chang C-H, Sengupta A, Bhunia S, Kastner R, Li H (2020) An overview of hardware security and trust: threats, countermeasures, and design tools. *IEEE Trans Comput Aided Des Integr Circuits Syst* 40(6):1010–1038
- Huang K, He Y (2019) Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection. *IEEE Trans Inf Forensics Secur* 15:3387–3400
- Huang K, He Y (2020) Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection. *IEEE Trans Inf Forensics Secur* 15:3387–3400
- Karimian N, Tehranipoor F, Rahman MT, Kelly S, Forte D (2015) Genetic algorithm for hardware trojan detection with ring oscillator network (ron). In: *Proc of 2015 IEEE international symposium on technologies for homeland security (HST), HST'15*. IEEE, pp. 1–6
- Lee H, Ha D (1993) Atalanta: an efficient atpg for combinational circuits. Technical Report, 93-12, Dept of Electrical Engineering, Virginia Polytechnic
- Li H, Liu Q, Zhang J (2016) A survey of hardware trojan threat and defense. *Integration* 55:426–437
- Lyu Y, Mishra P (2021) Maxsense: side-channel sensitivity maximization for trojan detection using statistical test patterns. *ACM Trans Des Autom Electron Syst (TODAES)* 26(3):1–21
- Majid AY, Saaybi S, Francois-Lavet V, Prasad RV, Verhoeven C (2023) Deep reinforcement learning versus evolution strategies: a comparative survey. *IEEE Trans Neural Netw Learn Syst*, pp. 1–19
- Mondal A, Biswal RK, Mahalat MH, Roy S, Sen B (2021) Hardware trojan free netlist identification: a clustering approach. *J Electron Test* 37(3):317–328
- Mondal A, Kalita D, Ghosh A, Roy S, Sen B (2023) Toward the generation of test vectors for the detection of hardware trojan targeting effective switching activity. *ACM J Emerg Technol Comput Syst* 19(4):1–16
- Mondal A, Mahalat MH, Mandal S, Roy S, Sen B (2019) A novel test vector generation method for hardware trojan detection. In: *Proc of 2019 32nd IEEE international system-on-chip conference (SOCC), SOCC'19*. IEEE, pp. 80–85
- Mukherjee R, Rajendran SR, Chakraborty RS (2022) A comprehensive survey of physical and logic testing techniques for hardware trojan detection and prevention. *J Cryptogr Eng* 12(4):495–522
- Nourian M, Fazeli M, Hély D (2018) Hardware trojan detection using an advised genetic algorithm based logic testing. *J Electron Test* 34:461–470
- Oya M, Shi Y, Yanagisawa M, Togawa N (2015) A score-based classification method for identifying hardware-trojans at gate-level netlists. In: *Proc of 2015 design, automation & test in Europe conference & exhibition (DATE), DATE'15*. IEEE, pp. 465–470
- Pan Z, Mishra P (2021) Automated test generation for hardware trojan detection using reinforcement learning. In: *Proc of the 26th Asia and South Pacific design automation conference, ASP-DAC'21*. Association of International Associations (UIA), pp. 408–413
- Saha S, Chakraborty RS, Nuthakki SS, Mukhopadhyay D (2015) Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability. In: *Proc of cryptographic hardware and embedded systems-CHES 2015: 17th international workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17, CHES'15*. Springer, pp. 577–596
- Saha S, Chakraborty RS, Mukhopadhyay D (2016) Testability based metric for hardware trojan vulnerability assessment. In: *2016 Euromicro conference on digital system design (DSD)*. IEEE, pp. 503–510
- Shi Z, Ma H, Zhang Q, Liu Y, Zhao Y, He J (2021) Test generation for hardware trojan detection using correlation analysis and genetic algorithm. *ACM Trans Embed Comp Sys (TECS)* 20(4):1–20
- Testability Measurement Tool sourceforge (2016) <http://sourceforge.net/projects/testabilitymeasurementtool>. Accessed 05 Jan 2016
- Xiao K, Forte D, Jin Y, Karri R, Bhunia S, Tehranipoor M (2016) Hardware trojans: lessons learned after one decade of research. *ACM Trans Des Autom Electron Syst (TODAES)* 22(1):1–23
- Zhou Z, Guin U, Agrawal VD (2018) Modeling and test generation for combinational hardware trojans. In: *2018 IEEE 36th VLSI test symposium (VTS)*, pp. 1–6

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Sandip Chakraborty pursued his Ph.D. in Computer Science and Engineering at the National Institute of Technology, Durgapur. Alongside his academic pursuits, he served as an Assistant Professor in the Department of Information Technology at Dr. B.C. Roy Engineering College, Durgapur. He earned his B.Sc.honors in Computer Science from Burdwan University in 2005 and later completed an M.Sc. in Computer Science from Sam Higginbottom University of Agriculture, Technology, and Sciences in 2012. His current research interests include Hardware security, Hardware Trojan, and Soft computing.

Archisman Ghosh is a Ph.D. student in the Department of Computer Science and Engineering at Pennsylvania State University. He completed his B.Tech. In Computer Science and Engineering from the National Institute of Technology Durgapur, India, in 2023. He works in the field of emerging technologies like hardware security, machine learning, and quantum computing.

Anindan Mondal received his B.Tech. degree in computer science and engineering from Kalyani Government Engineering College, Nadia, India, in 2013, and ME degree in computer science and engineering from Jadavpur University, Kolkata, India, in 2015. He received his Ph.D. in computer science and engineering at the National Institute of Technology, Durgapur, India, in 2023. He works as Assistant Professor

in Computer Science and Engineering at Asansol Engineering College. His current research interests include Hardware security, Hardware Trojan, and Physically Unclonable Function (PUF).

Bibhash Sen received the B.Tech. degree in computer science and engineering from NERIST, Nirjuli, India, in 2002, and the M.E. and Ph.D. degrees in computer science and engineering from IIST, Shibpur, India, in 2007 and 2015, respectively. He is currently an Associate Professor at the Computer Science and Engineering Department, NIT Durgapur, Durgapur, India. His current research interests include hardware security, security systems for wireless sensing, hardware Trojan, quantum-dot cellular automata, reversible logic, and fault-tolerant architectures for emerging nanodevices.