Journal of Electronic Testing (2023) 39:371–385
https://doi.org/10.1007/s10836-023-06071-w

Multi-Objective Optimization Based Test Pattern Generation for Hardware Trojan Detection

Vijaypal Singh Rathor[1] · **Deepak Singh**[2] · **Simranjit Singh**[3] · **Mohit Sajwan**[3]

Received: 9 January 2023 / Accepted: 17 May 2023 / Published online: 16 June 2023

Abstract
Hardware Trojan (HT) is a severe security threat during the development of an integrated circuit that can deviate the IC from its normal function and/or leak sensitive information during in-field operations. Trojans are often inserted during the fabrication phase, and to have Trojan-free ICs; it is highly desirable to detect them during post-silicon testing. Different test pattern generation-based HT detection techniques are reported in the literature to detect the Trojan during post-silicon testing. The existing methods provide low coverage and require a large number of test patterns. This paper proposes a new test pattern generation-based HT detection technique that provides high coverage while requiring less number of patterns. The proposed technique generates the optimal number of test patterns that activate the rare events by framing the problem as multi-objective optimization and solving it through a non-dominated sorting genetic algorithm (NSGA-II). The Trojans are mostly inserted using rare-triggered nodes (highly vulnerable, low controllable, and low observable). Thus, our technique applies the generated patterns during post-silicon testing to activate Trojans. Further, we also present the use of checker (detection) logic along with a proposed approach to effectively detect the Trojan during testing. The experimental evaluation on ISCAS benchmarks shows that the proposed technique provides 12 times higher trigger coverage with 1/3 fewer test patterns than the best-known existing genetic algorithm-based technique.

Keywords Hardware Trojan · Rare-triggered nets · Test pattern generation · Multi-objective optimization · Genetic Algorithm · Hardware testing

1 Introduction

The globalization in semiconductor industries makes the development of integrated circuits (ICs) vulnerable to a malicious inclusion called hardware Trojan (HT) insertion [3, 23]. Due to their stealthy nature, HTs are mostly inserted at the rare nodes; thus, they are not detected during conventional verification and validation [3]. The HT can be inserted during different phases of IC design [3]. The attacker can use the inserted Trojan to infer confidential information and/or to maliciously modify the original design functionality during in-field operations or harm the company's reputation [15]. However, several design techniques such as logic obfuscation, camouflaging, etc., are reported [3, 21, 22, 31] that prevent the Trojan insertion during the IC life cycle at the cost of some design overhead. HT detection is a prominent approach for identifying malicious inclusion without introducing any overhead. Several HT detection techniques are reported in the literature to detect the Trojan during pre-silicon [19, 31, 36]

Responsible Editor: U. Guin

✉ Vijaypal Singh Rathor
vrathor@iiitdmj.ac.in

Deepak Singh
dsingh.cs@nitrr.ac.in

Simranjit Singh
simranjit.singh@bennett.edu.in

Mohit Sajwan
mohit.sajwan@bennett.edu.in

[1] PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, India

[2] Department of CSE, National Institute of Technology, Raipur, India

[3] Department of Computer Science Engineering, Bennett University, Greater Noida, India

and post-silicon validation using logic testing/automatic test pattern generation (ATPG) [4, 18] and/or side-channel analysis (SCA) [14, 33]. Although pre-silicon detection techniques can save high fabrication costs if a Trojan is detected during the design phase, the attacker mostly inserts the Trojan in the fabrication house due to outsourcing of the fabrication facility. Therefore, the detection of HT during post-silicon testing is most important.

The SCA-based post-silicon techniques measure physical parameters such as power, delay, etc., and compare them with the golden model to identify the malicious inclusion [7, 33]. The Trojan insertion affects these parameters. Any significant deviation in these parameters will be considered a Trojan. Due to environment noise and process variation, the detection of stealthy and small combinational Trojans (having few gates) is challenging using SCA-based approaches [4, 14]. On the other hand, the ATPG based methods are most suitable to detect small and combinational Trojans during post-silicon. However, they are ineffective in detecting sequential and analog Trojans [31]. Therefore, this paper considers the problem of detecting the small and combinational Trojans using ATPG based methods during post-silicon testing.

An adversary can insert the Trojan anywhere in the design. Detecting all combinational Trojans in a design is an exponential problem [37]. However, the non-judicious/random insertion of Trojan either may make it invalid (no impact on the design) or may make it easy to detect it with normal functional testing [4, 5, 9, 22]. Therefore, an adversary inserts the Trojan judiciously at the rare nodes to make it stealthy or evade functional testing as reported in [4, 9, 22]. Hence, the ATPG based techniques generate test patterns, also referred to as test vectors/cases, to activate the rare nodes for detecting the small/stealthy combinational Trojans during functional testing [4, 24]. The test pattern denotes the input value/pattern or sequence of bits (*i.e.*, 0101010) applied at the primary input of the digital circuit to validate its output. It means the test patterns are the set of input patterns applied during testing to verify the digital circuit functionality. Hence, the IC designer aims to generate such test patterns whose application can activate rare events and detect the existence of combinational Trojans in digital circuits/IC during testing.

Several ATPG based techniques such as multiple excitations of rare occurrences (MERO) [4], genetic algorithm (GA) based [24] and advised GA (AGA) based, [18] techniques are proposed in the literature, that generate the patterns for combinational HT detection. The GA-based techniques have gained more attention due to their high coverage over the other methods in all these techniques. But the existing GA based techniques generate a significantly large number of test patterns to achieve the desired trigger coverage. This is because they do not simultaneously focus on the minimization of patterns along with achieving high coverage during testing. Further, they are also inefficient in activating hard-to-detect or low-triggering probability Trojans. It is highly desirable to achieve high trigger/Trojan coverage along with detecting hard-to-activate HTs with minimum patterns to reduce the testing time/cost. Furthermore, the HT detection requires activating and propagating its effect to an output [4, 6]. But considering the Trojan-effect propagation while generating the test patterns, one needs to set the logic values of the signals responsible for propagation, resulting in reduced Trigger coverage. Since the existing techniques consider effect propagation at the output for HT detection, they may fail to generate the patterns that provide high trigger coverage.

For example, consider a circuit, as shown in Fig. 1(a), where a functional Trojan is inserted using trigger T1 and payload T2. In this case, the pattern a=1, b=1 activates and propagates the effect of inserted Trojan to the output. Suppose we assume there are multiple combinational Trojans in a design and there exists another pattern that activates more Trojans (but not propagates effect) than the a=1 and b=1. In that case, this pattern provides high trigger coverage over the a=1 and b=1. However, Trojan can only be detected when its effect can be observed at the output. It is not always possible/required propagating effect of some Trojans (called non-functional/parametric) at the output. The sample example of a non-functional Trojan is shown in Fig. 1(b).

The aim of inserting such Trojans may be to increase the design overhead or leak sensitive information. These Trojans are not inserted to change the functionality; they are just inserted to observe the signal value or other side channel parameters to leak sensitive information. Lin et al. [15] has presented a Trojan called Malicious Off-Chip

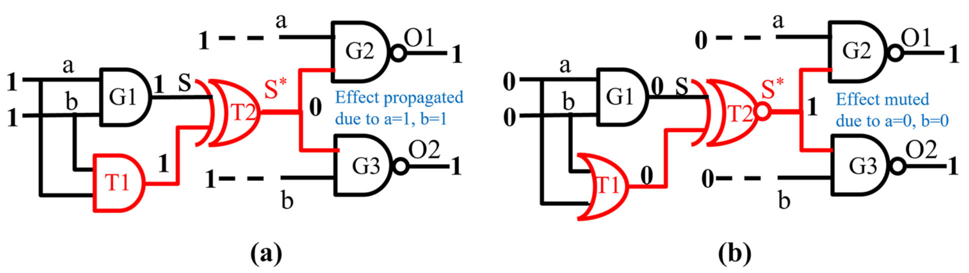
a
b
G1
S
T1
T2
S*
G2
G3
O1
O2
1
0
Effect propagated due to a=1, b=1
Effect muted due to a=0, b=0
(a)
(b)

Fig. 1 **a** Functional Trojan, effect propagate to the output and Trojan activation change the output **b** Non-Functional Trojan, the effect does not propagate to the output and Trojan activation does not change the output. These Trojans are inserted using trigger (T1) and payload (T2) at signal S

Leakage Enabled by Side-Channels (MOLES) to leak the cryptographic key. Such type of Trojans does not require propagation effect at the primary output. Therefore, generating the patterns without considering the Trojan effect's propagation increases trigger coverage significantly.

Although it is important to detect the Trojan in the IC before its deployment, the ATPG based techniques cannot ensure the detection of all types of HTs due to the large space and diverse forms of Trojans [6]. Thus, the researchers have also used design-for-trust (DFT), where an additional detection (Checker) circuitry is judiciously inserted in design to observe/detect the Trojan during in-field operations [6]. HT payload modifies the original logic value at a node on its activation. It also disrupts the correlation between the logic values of that node and its neighbourhood nodes [6]. Therefore, this checker monitors the relationship between neighbourhood signals logic values, and any deviation in expected correlation is reported as an error signal. However, Chakraborty et al. [6] only use this checker logic for detecting the Trojan during online monitoring; they do not attempt to detect the Trojan during post-silicon testing. Since online (in-filed during application execution time) detection can only bypass the HT or disable the chip, it does not provide any opportunities to fix or reject the design.

Problem Statement and Major Contributions Since this paper considers the threat model as the attacker is in the fabrication house, where he can insert the Trojan in multiple ICs, it is required to test every chip after fabrication to ensure it is Trojan-free. If the number of test patterns is more, it will significantly increase the testing time. Therefore, the ATPG based HT detection technique aims to achieve high coverage by generating minimum patterns. The test generation problem for HT detection looks like an optimization problem because we have to generate minimum patterns and achieve high trigger coverage. Therefore, we can formulate this problem as a multi-objective optimization problem, where two complementary objectives (minimum test patterns and maximum coverage) must be simultaneously satisfied. However, the existing ATPG based techniques mainly focus on achieving high coverage. They do not simultaneously focus on both objectives. In other words, they have not considered ATPG as a multi-objective optimization problem. Therefore, they require large patterns to achieve desired coverage. Further, due to consideration of the propagation HT effect at the output, they failed to generate patterns to trigger extremely hard-to-activate Trojans. Therefore, this paper formulates the problem test generation for HT detection as a multi-objective optimization problem. To the best of our knowledge, we are the first to solve HT detection problems using multi-objective optimization.

The following are the major contributions of this paper:

1. Present the analysis of the existing ATPG and DFT based HT detection approaches.
2. Propose a first multi-objective optimization-based technique for test pattern generation that provides high trigger coverage with minimum patterns.
3. The test generation problem is framed as a multi-objective optimization problem and uses the non-dominated sorting genetic algorithm (NSGA-II) to generate test patterns to activate the combinational Trojan.
4. Propose a procedure that combines the proposed ATPG and checker insertion approach to provide complete protection against the hardware Trojan threat during the IC life cycle.
5. Experimental evaluation of the proposed HT detection technique on the ISCAS-85 benchmarks provides, on average, 12 times higher trigger coverage with 1/3 reduced patterns over the best-known existing technique.

The remainder of this paper is organized as follows. Section 2 presents an analysis of existing HT detection techniques. Section 3 presents the proposed multi-objective optimization based test pattern generation technique for HT detection. The experimental results and analysis are given in Section 4. Finally, Section 5 concludes and presents the future work of this paper.

2 Analysis of Existing HT Detection Techniques

Various techniques are reported in the literature to detect hardware Trojans during pre-silicon and post-silicon. Hicks et al. [13] first addressed the problem of identifying HTs inserted at design time by formulating it as an unused circuit identification problem. However, it can not be safely assumed that "used circuits" are free of HT and "unused circuit" contain HT [29]. To reduce the false positive, Waksman et al. [30] present a technique called functional analysis for nearly unused circuit identification (FANCI) that applies Boolean function analysis to flag suspicious signals which have a weak impact on output. The attacker can insert the Trojan so that the HT-related signals exhibit the same effect on output as functional signals [34]. Further, Zhang et al. [35, 36] present a technique called VeriTrust, that detects an HT by identifying the inputs in the combinational logic cone that seems redundant for the normal functionality of the output wire under non-trigger condition. However, this technique also fails to detect the Trojans with implicit malicious behaviour [11].

Besides the above, recently, several machine learning-based HT detection techniques are also reported in the literature. These techniques use different Trojan features to classify the circuits into different groups and identify the group with Trojan-infected circuits. Oya et al. [19]

present a score-based technique that identifies the nets included in HTs based on several extracted features. The major challenge in this approach is setting the correct threshold for classification. Further, Oya et al. [12] also extract the five structural Trojan features and learn the support vector machine (SVM) to identify the Trojan nets in the gate-level netlist. Instead of five features, Salmani [26] statically analyze the combinational controllability (CC) and combinational observability (CO) values of each signal and classify it as a Trojan-infected or Trojan-free using K-Means clustering. However, this technique cannot classify the nets where the inter-cluster distance between the signals is less than 100. Further, choosing a proper threshold for classification is also a big challenge. To overcome these limitations and improve the detection rate, Xie et al. [32] use the SVM with additional features and K-Means clustering.

However, all these techniques only detect the Trojan inserted during the design phase, whereas the attacker can also insert the Trojan during the fabrication phase. Therefore, various test pattern generation based techniques have been reported in the literature to detect the Trojan during post-silicon testing. First, Chakraborty et al. [4] present a technique called MERO that generates the test vectors to activate the low triggering probability conditions multiple times during post-silicon to facilitate Trojans detection. Since MERO uses a simple heuristic, i.e., perturbs one bit for test generation, it is not adequate to activate all possible rare nodes and fails to excite hard-to-activate sites [24]. To improve the Trojan and trigger coverage, Saha et al. [24] present a genetic algorithm (GA) and Boolean satisfiability (SAT) based test pattern generation technique. Though this approach improves the coverage, the SAT requires a long time to generate the patterns that activate hard-to-trigger sites. Therefore, an advised GA (AGA) based approach is reported that also includes CC and CO parameters in the fitness function while generating the test vectors [18].

Besides the above, Liu et al. [16] have applied the genetic algorithm in the training phase of a probabilistic neural network for detecting the hardware Trojan using the side channel parameters. Further, Pan and Mishra [20] have proposed reinforcement learning based automatic test generation for HT detection. This technique provides high trigger coverage while reducing test generation time. A detailed survey of different HT detection methods can also be found in [8]. All the above existing approaches generate the patterns without minimizing them or considering only objective, i.e., trigger coverage. Thus, they require large patterns to provide a given trigger coverage. However, Shi et al. [28] proposed a correlation analysis and genetic algorithm based ATPG method that reduces the test patterns by reordering them to achieve a given coverage. But this technique also uses a genetic algorithm only for a single objective, *i.e.*, test reordering to improve Trojan activation or trigger coverage.

Since the existing HT detection techniques cannot detect all the HTs due to their diverse forms, several DFT techniques are also reported, which increases the triggering probability of Trojans improving their detection during post-silicon testing [9, 22, 25]. However, these techniques are ineffective and could not facilitate detection during online monitoring [6, 27]. Therefore, to facilitate detection during in-field operation, a checker logic is inserted into the design that observes the Trojan behaviour [2, 6]. Any undesirable behaviour is reported with a high error signal. Unfortunately, these techniques are ineffective in detecting the Trojan (discussed above) and do not provide any opportunities to fix and reject the design before its deployment.

To overcome the limitations of existing methods, a new technique for HT detection has been proposed that utilizes the Non-Dominated Sorting Genetic Algorithm (NSGA-II) to generate patterns. These patterns provide high trigger coverage with a low number of patterns. Additionally, the proposed technique can be easily combined with a DFT-based online monitoring technique to observe Trojan behaviour during testing effectively.

3 Proposed Multi-Objective Optimization Based HT Detection Technique

This section first presents the identification of rare nodes, the proposed multi-objective optimization framework for test pattern generation and the NSGA-II algorithm. Further, it presents the insertion of existing checker logic [6] for observing Trojan behaviour followed by the procedure for HT detection.

3.1 Rare-Triggered Nodes Identification

Since Trojans are mostly inserted at the rare-triggered nodes [22, 24, 25], we first identify the rare-triggered nodes in a given circuit using the vulnerability factor (VF) metric [22], i.e., $VF = P_0 - P_1$ along with controllability (CC) and observability (CO) [10]. Here, CC and CO denote the difficulty for setting and observing a circuit net to logic ‘0’ or ‘1’ respectively [10]. Here, P_0 and P_1 are the probability of getting logic (*i.e.*, transition probability) ’0’ and ’1’ at a node, respectively. The value of VF lies between -1 to 1. The higher $|VF|$ (absolute) value indicates higher vulnerability/rareness to Trojan insertion. Figure 2 shows the insertion of Trojan using G5 and G6. Activating this Trojan with trigger Tg1 will change the internal signal of the circuit if it is functional. However, alteration in the output of IC depends on the propagation of the effect of Trojan. If the effect of the Trojan is not propagated at the output on activation, then

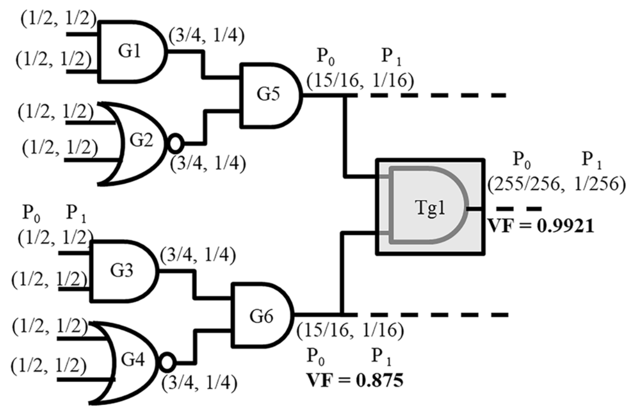

Fig. 2 The example circuit for showing how the Trojan can be inserted using the nets G5 and G6. Here, Tg1 denotes the Trigger of the Trojan. This Trojan will activate with the pattern “11001100". On activation it will flip the original signal of the circuit

such Trojan will be dormant forever. Therefore, the attacker generally inserts the Trojan such that its effect will propagate at the output on its activation. Since activation of rare nodes activates the Trojan, there is a high (or 100%) probability of altering the output of IC on activation of rare nodes/Trojan.

In the given example, we use the nodes G5 and G6, whose VF values are 0.875, as rare nodes to insert the Trojan. But it is required to have a specific value that can decide which nodes are rare. Similar to the existing methods [18, 22, 24], we also use the vulnerability threshold (V_{Th}) value to classify a node whether it is rare-triggered or not. If the value of *VF* of a node is more than V_{Th}, then that node is considered rare-triggered (i.e., vulnerable to Trojan insertion); otherwise, it is not. It is reported in [18, 24] that the Trojans inserted at the nodes having transition probability or *VF* less than 0.9 cannot evade functional testing. Therefore, the attacker uses very low triggering probability nodes to insert the Trojan. Thus, we consider two V_{Th} values, *i.e.*, 0.90 and 0.95 to identify the rare nodes in simulation.

3.2 Proposed Multi-Objective Optimization Based Test Generation Method

We propose a multi-objective optimization based test generation technique that generates the test vectors to activate the rare nodes to facilitate Trojan detection during post-silicon testing. The proposed method formulates the problem of test pattern generation to optimize the following two objectives:

- **Objective-1:** Maximizing the trigger coverage so that the Trojan detection/activation could be improved during post-silicon testing.
- **Objective-2:** Minimizing the number of test patterns in a subset or solution so that testing time can be reduced.

This problem perfectly matches the multi-objective optimization problem. Meta-heuristic algorithms are computational intelligence methods especially employed for solving advanced optimization problems [1]. They efficiently explore the search space in order to find near-optimal solutions. The Non-Dominated Sorting Genetic Algorithm (NSGA-II) [17] is the most popular Meta-heuristic Pareto-based multi-objective optimization algorithm effective in solving such problems. Thus, we used NSGA-II to solve our problem, as detailed in the next subsection. The NSGA-II is very effective over the previously used GA and AGA based methods in handling the noise, finding diverse solutions with multiple objectives by generating Pareto fronts. The main advantage of NSGA-II over the previously proposed GA based test generation methods is that it generates the optimal patterns by handling multiple objectives (*i.e.*, maximum trigger coverage with minimum test patterns). Whereas the GA and AGA based methods generate the patterns only considering a single objective, i.e., trigger coverage. In addition, the broader applicability suggests that the NSGA-II is an acceptable choice for generating the optimal test patterns.

3.2.1 NSGA-II Based Algorithm for Optimal Test Generation

The proposed method randomly considers the *M* number of test patterns of each *D* length equal to the number of primary inputs (PIs) in the circuit. Though the value of *M* can vary, we used the $M = 50$ to evaluate the proposed method. A total $M \times 2^D$ number of test patterns will be used as a search space for a circuit with a *D* number of PIs. For the practical size of *D*, a huge pool combination of different sets of *M* test vectors with high trigger coverage is possible from the total patterns. Here, we employ NSGA-II as shown in Algorithm 1, to identified the optimal subset of test patterns (size *N*) ranging between $0 < N <= M$ from this huge set. NSGA-II was developed to address the limitations of previous evolutionary algorithms, which lacked elitism and utilized a sharing parameter to maintain a diverse Pareto set. The algorithm employs a quick non-dominated sorting algorithm, as well as sharing, elitism, and crowded comparison. Elitism improves the algorithm’s convergence rate by preserving the best solutions from the previous iteration. Furthermore, its adoption of a rapid non-dominated sorting algorithm results in a substantial decrease in computational complexity. NSGA-II is a population-based search approach where the selected subsets constitute the population, and each solution of the population is termed as an individual/chromosome.

Algorithm 1 : Multi-Objective Evolutionary based Test Pattern Generation (MOETPG) Algorithm

```
Procedure: MOETP (X, npop, ntime, bound)
    Input: X,        ▷ Dataset of size m instances and n attributes
           npop,     ▷ Population size
           ntime,    ▷ Maximum Iterations
           bound;    ▷ Lower and upper bound for Initialization
1:  $pop \leftarrow Initalize\_Pop\_Random(npop, range)$
2:  $fit\_np \leftarrow Compute\_Fitness(pop, X)$    ▷ Calling procedure CEF and CAF for fitness
3:  $front \leftarrow Fast\_Non\_Domination\_Sort(pop, fit_np)$;   ▷ Front Generation
4:  $rank \leftarrow Assign\_Rank(front, pop)$;   ▷ Each individual according to their fronts
5:  $dist \leftarrow Crowding\_Distance(pop, rank)$
6:  $set\ t \leftarrow 0$
7:  while ($t < ntime\ or\ termination\ criteria\ met$ ) do
8:      $qpop(t) \leftarrow Genetic\_Reproduction(pop(t), fit_np)$;   ▷ Selection, crossover, mutation
9:      $fit_nq \leftarrow Compute\_Fitness(qpop(t), X)$;
10:     $R(t) \leftarrow P(t)Q(t)$;   ▷ Merge the newly generated solution with the original solutions
11:     $front(t) \leftarrow Fast\_Non\_Domination\_Sort(R(t), fit_nq)$
12:     $rank(t) \leftarrow Assign\_Rank(front(t), R(t))$
13:     $pop(t+1) \leftarrow [\ ]$; $i \leftarrow 1$   ▷ Initialize next population
14:     while ($|pop(t+1) + front(t)| < npop$) do
15:         $fron_i \leftarrow Crowding\_Distance(R(t), rank(t))$
16:         $pop(t+1) \leftarrow pop(t+1)\ U\ fron_i$
17:         $i \leftarrow i+1$
18:     end while
19:     Sort and add top npop elements from the front in pop(t+1)
20:     $t \leftarrow t+1$
21: end while
22: Report the final members of pop as final solution
```

Individual/Chromosome Representation Figure 3 presents the structure of an individual used in the proposed method. Each individual in the population represents the set of test patterns. The individual is a combination of hundreds and thousands of genes (*i.e.*, test patterns), and the size of a gene can be varied. In this figure, the single chromosome consists of length $M = 10$ (columns) genes, denoting the maximum number of test cases possible in a test set. In the coding of individual representation, the numbers of genes are fixed to the number of test patterns in a subset. Each gene will be of $D + 1$ dimensions encoded as the vector of a binary string. The first dimension of the gene is a binary value (*i.e.*, '1'/'0') indicating the valid/favourable or invalid/unfavourable test pattern in the individual. In the example, the five patterns as described with blue colour are assumed to be favourable/valid. The rest D dimension of the gene represents the bit values treated as an input test pattern. By default, we use 50 genes, and each gene will have a test pattern to build the test generator model.

Now, the NSGA-II as shown in Algorithm 1 starts with the random initialization of chromosomes. Further, the initial population is updated by performing selection, crossover, and mutation operations. Next, the fitness of each chromosome is calculated as defined below.

Fitness Function The selection of diverse test patterns in a subset for the next generation is vital to achieving high trigger coverage. Since we have formulated the problem of test pattern generation based on the above-mentioned two objectives, *i.e.*, maximum coverage with minimum patterns, we also define the two fitness functions. We consider the first fitness as an average number of test patterns used in a subset. Considering M_{sp} is the number of patterns selected/favourable from the total patterns M in an individual. Thus, the first fitness is evaluated as follows:

$$minimize \quad obj_1(individual) = M_{sp}/M \tag{1}$$

where $obj_1(individual)$ is the average number of the test patterns selected by the optimal model, we consider this as one of the fitness criteria. The goal is to minimize the value of obj_1 by reducing the test patterns in the individuals. Now the second objective of our framework is to have a maximum trigger coverage or activation of maximum rare combinations by the corresponding test pattern. Before defining the trigger coverage, we quantify the rare nodes as follows:

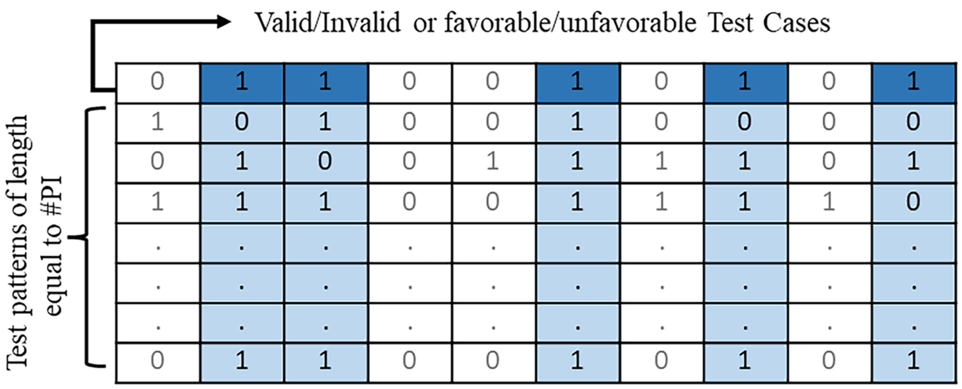

Fig. 3 The representation of an individual/chromosome or a test-set in our multi-objective optimization technique. The binary value (*i.e.*, '1'/'0') in first row indicates the valid/favorable or invalid/unfavorable test pattern

$$RV_i = \begin{cases} \frac{\alpha}{|VF|+c_1} + \beta(CC_0 + CO), & R_i \to 0 \\ \frac{\alpha}{VF+c_1} + \beta(CC_1 + CO), & R_i \to 1 \end{cases} \quad (2)$$

where R_i represents the i^{th} rare node. VF denotes the vulnerability factor of a rare node [22]. The CC_0 and CC_1 are the controllability values for logic '0' and '1', and CO is the observability value of the rare nodes, also known as SCOAP parameters for measuring the testability of the circuit [10]. Further, the c_1, α and β are constants used for making normalized values and matching two parts of the relationship [18]. Finally, based on the above-computed value of rare nodes, we define the fitness function for evaluating the test vectors of an individual, as shown below.

$$maximize \quad obj_2(individual) = \sum_{t \in M_{sp}} \sum_{i \in R} RV_i \quad (3)$$

where R is the number of rare nodes and RV_i is the rareness value of a rare node i. The $obj_2(individual)$ denotes the fitness value of an individual in the population. An individual is more valuable if it exhibits/activates more rare nodes/combinations. The value of obj_2 should be maximized for achieving high trigger coverage.

Based on the computed fitness, the best-fit individuals are identified for the reproduction process. The tournament selection is used to select the fittest individual, where the parent is selected using fitness value. The selected parent then carries out a crossover to become a new solution. A Gaussian mutation is done after crossover to avoid the local minima problem. Due to the elitism-based selection technique, the updating techniques will vary compared to the initial generation. The elitism technique is implemented by comparing the current population with previously found best non-dominated solutions. Finally, operations like front calculation by non-domination sorting, computation of the rank, and removing the underfit solution from the pool are carried out in an iterative way to generate an optimal set of test patterns. The complete framework/workflow of the proposed multi-objective optimization method is presented in Fig. 4.

The NSGA-II basically generates the Pareto-based solutions (as shown in Results Section) to identify the optimal set of patterns that provide high trigger coverage with minimum patterns. The Pareto fronts denote the trade-off between values of objectives, which plays a vital role in identifying the optimal solution. The best Pareto front is identified by conducting empirical experiments (10 runs) for each dataset with the proposed model. Though running for ten times increases the cost of test generation, it is vital to reduce the testing time (due to many copies of IC) rather than reducing the test generation time. This is because test generation is a one-time process, whereas testing of IC required to test all copies of IC. The NSGA-II algorithm constructs the new set of non-dominated solutions/Pareto by reproduction, crossover, and mutation operations until an optimal Pareto front is obtained [17]. The newly generated population is merged

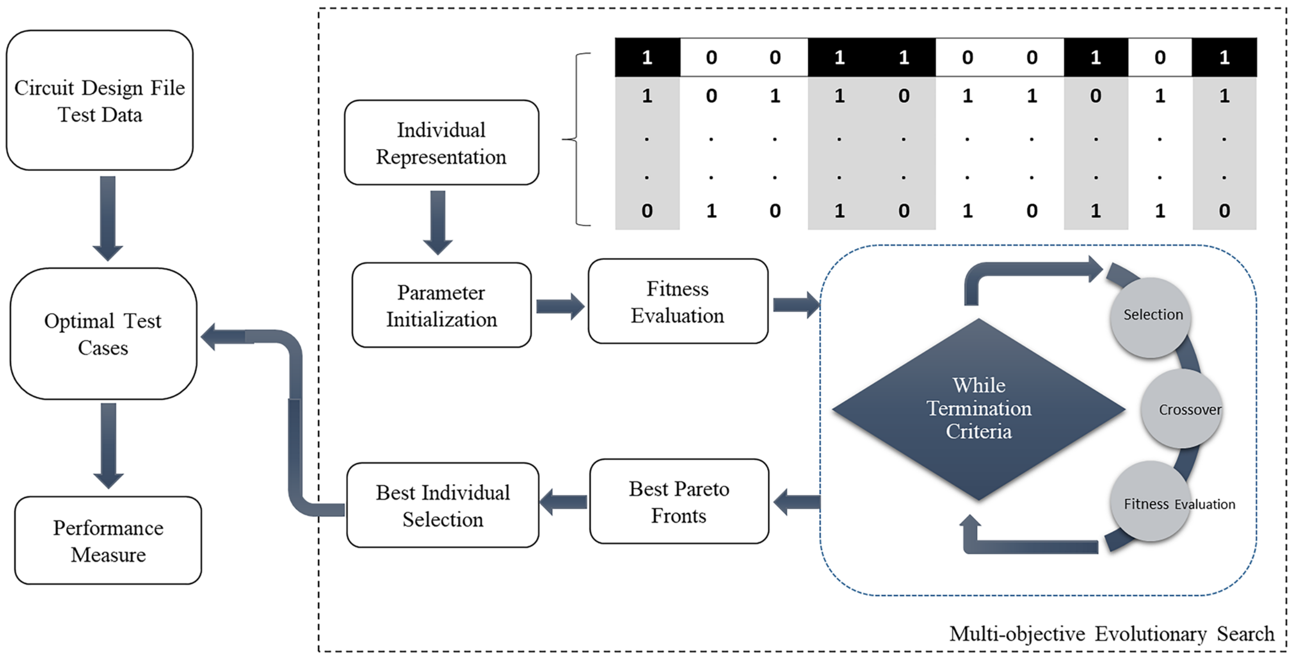

Fig. 4 Process framework/flow diagram of the proposed multi-objective optimization based test patterns generation approach. Here, the circuit files, and population as individual as provided as inputs to the search method. Based on the inputs, the optimal patterns are generated after multiple iterations

with the parent population, and then it is sorted by evaluation criteria called the fitness function of each individual in the population. The survival of a solution in the population depends on its fitness value. The best individuals (having the highest fitness) are considered for the next generation. The final generation corresponds to the set of evolved individuals with the generated Pareto front. The solutions in the Pareto front are evaluated for the coverage and test patterns criteria. A solution that provides high coverage with minimum patterns is considered an optimal solution.

Since we generate the test patterns without considering the propagation of the Trojan effect, we also proposed to insert the checker logic in the low-observable nodes to detect/monitor malicious behaviour, as discussed in the next section.

3.3 Checker Circuit Insertion

The above proposed multi-objective NSGA-II based approach generates the test patterns to activate the Trojans to detect them during testing. However, the attacker may also make the Trojan stealthy by connecting their payload at the low observable nodes. In this case, the activation of Trojans may not be sufficient to detect the Trojans effectively. Therefore, a designer can also consider inserting the checker logic [6] as shown in Fig. 5 at the low observable nodes to detect the Trojan effectively during testing as well as in-field operation.

This checker logic monitors the Trojan behaviour based on the correlation between the neighbourhood signals (X_1, X_2 and Y_1, Y_2) of the target node (S). Any undesirable behaviour reported with an error signal, *i.e.*, $error = 1$. Since the payload of the Trojan is generally connected with low observable nodes, the insertion of checker logic at these nodes is very effective in detecting the Trojan behaviour on its activation [6]. The proposed multi-objective optimization based HT detection technique generates the test patterns that effectively activate the Trojans. It is observed that the checker insertion is very effective for detecting malicious

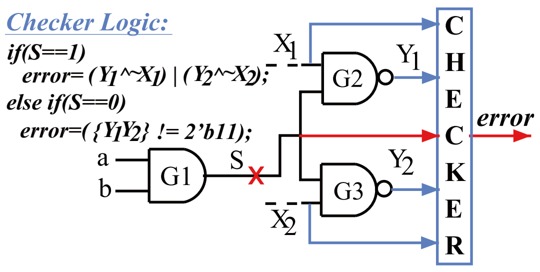

Fig. 5 The insertion of checker logic for monitoring the Trojan behaviour at the low observable or target node S [6]

behaviour, whereas test generation is very effective for Trojan activation. Therefore, the proposed HT detection combines these best features of test generation and DFT to detect the Trojan during post-silicon testing effectively. We propose using the inserted checker *i.e.*, design-for-trust (DFT) to identify Trojan behaviour on its activation. Since this work mainly focuses on generating test patterns using multi-objective optimization for Trojan activation, further analysis/discussion of checker logic insertion or DFT is out of the scope of this paper.

The complete procedure of the proposed HT detection technique is provided in the next subsection.

3.4 Complete Steps for Hardware Trojan Detection

The complete procedure of the proposed hardware Trojan detection technique from test pattern generation to hardware Trojan detection is presented in Procedure 1. The proposed technique first inserts the checker logic in design at the low observable nodes. Further, it generates the optimal test patterns for the rare-triggered nodes using the proposed multi-objective optimization-based approach. Finally, we apply the generated test patterns during post-silicon testing to check whether a given design contains a Trojan or not.

Procedure 1: Complete Procedure/Steps for HT Detection

```
Step 1. Fist takes the Design Net-list, Vulnerability Threshold ($V_{Th}$) as input of the procedure.
Step 2. Identify the low observable nodes or the nodes which are most suitable for connecting Trojan payload in design.
Step 3. Insert the Checker logic at the identified low observable nodes.
Step 4. Identify the rare-triggered nets in the design using $V_{Th}$.
Step 5. Applies the proposed multi-objective optimization based approach to generate optimal test patterns.
Step 6. Finally, apply the generated patterns during the post-silicon testing and check the status of checker logic as well as primary output to detect the Trojan.
Step 7. If there is an undesired behaviour either at checker logic or at the primary output, the design is considered as Trojan infected, otherwise considered as Trojan free.
```

Since the proposed HT detection technique combines the best features of test generation and DFT based online monitoring, it addresses all the limitations (mentioned in Section 2) of existing techniques and effectively detects the Trojan during post-silicon testing. In contrast to existing GA-based test generation approaches, the proposed approach considers multiple objectives, *i.e.*, the number of test patterns and trigger coverage during test generation. Thus, it provides high trigger coverage with minimum patterns. Since we insert the checker logic at low observable nodes to observe the Trojan behaviour, it eliminates the need to propagate the HT effect at the output. Hence, in the proposed approach, generating the test patterns only for Trojan activation (without considering effect propagation) achieves high trigger coverage and activates stealthy Trojans. Additionally, the proposed approach can generate patterns for activating or detecting the Trojan whose effect does not propagate at the output. Finally, the proposed HT detection technique detects the Trojan during post-silicon testing, which also provides an opportunity to fix or reject the design before its deployment in case of any malicious inclusion. The combination of test pattern generation and DFT based online monitoring in HT detection provides additional protection against Trojan hardware.

In this paper, we mainly focus on activating the combinational Trojans only. However, it is reported in [4] that exciting rare events multiple times can also activate the sequential Trojan. Further, the proposed method functions in the same way as existing GA [24] and AGA [18] based methods function. Therefore, in a similar way, the proposed approach can be extended to detect the sequential Trojan. The simulation results and comparative analysis of the proposed technique are presented in the next section.

4 Experimental Results and Analysis

This section presents the experimental setup, simulation results and a comparative analysis of the proposed HT detection technique.

4.1 Experimental Setup

The proposed test vectors generation algorithm is implemented in MATLAB, and its effectiveness is evaluated for trigger coverage on various ISCAS-85 benchmark circuits. The proposed method mainly focus on activating combinational Trojans. Since ISCAS-85 benchmarks are most relevant to our research problem and used by existing MERO [4], GA [24] and AGA [18] methods, we have used these benchmarks for evaluation and comparison purposes. The rare nodes in benchmarks are identified using VF_{Th} metric [22] for different V_{Th} values, i.e, 0.85, 0.9, 0.95. We identified the optimal sets of test patterns while considering a maximum of 50 test patterns in each set. We also implemented the two variants of GA-based techniques [18, 24] by considering 200 population sizes and running for 1000 iterations to evaluate the effectiveness of the proposed method.

4.2 Simulation Results and Discussion

Our algorithm first identifies the number of rare nodes in benchmarks for different V_{Th}, as shown in Table 1. It can be observed from this table that the c432 is the only benchmark that does not exhibit any rare node for $V_{Th} = 0.95$.

The proposed multi-objective optimization-based technique is employed on ISCAS-85 benchmarks, and the simulation results are computed with three different V_{Th}, i.e., 0.85, 0.90, 0.95 for all these benchmarks. To display the ability to search by multiplicative genetic algorithm, we obtained the optimal Pareto front obtained during the evaluation of the benchmark circuits. The simulation results of the three circuits are presented in Fig. 6. The organization of the Pareto fronts is made to demonstrate the performance on different thresholds 0.85, 0.90 and 0.95 of c499, c2670, and c7552, respectively. The three circuits are selected to observe the Pareto performance on the diverse sizes (number of gates and the number of inputs) circuits. The analysis of the Pareto fronts mainly focuses on two aspects: (1) the convergence of the obtained solutions; (2) the diversity of the non-dominated solutions. Although these observations are not exhaustive, they provide a good basis to assess the performance of a multi-objective algorithm in optimal test pattern generation for Trojan detection. The Pareto plots show that NSGA-II obtains a good approximation to the true global PFs on any size circuit. In addition, NSGA-II also obtains a set of non-dominated solutions with good diversity and convergence. It can also be observed from these Pareto graphs that out of 50 solutions, only 35 optimal solutions are identified in all the benchmarks. These results demonstrate that the proposed model maintains a good diversity of solutions for test pattern generation for hardware Trojan detection.

From the computed Pareto solutions of different V_{Th}, we extracted the solutions with maximum coverage, minimum test cases, and the solution with average coverage and test patterns, as shown in Tables 2, 3 and 4. We have computed trigger coverage by considering the maximum possible trigger size in each benchmark. It can be observed from these tables that the proposed technique provides on an average 4.0625E+27, 4.35E+26 and 8.90E+20 trigger coverage while requiring only 45.3, 44.2 and 42.8 test patterns for $V_{Th} = 0.85$, $V_{Th} = 0.90$ and $V_{Th} = 0.95$ respectively in case of maximum coverage. The proposed technique provides an average of 8.9E+11 and 2.3E+11 coverage while requiring 6, 5.2, and 5.1 test patterns only in case of minimum test patterns for $V_{Th} = 0.85$ and $V_{Th} = 0.90$ and $V_{Th} = 0.95$ respectively. In the average case, it can be observed the proposed technique provides 4.3E+16, 1.8E+16 and 4.3E+15 coverage with 23.3, 22.1 and 23.4 patterns for $V_{Th} = 0.85$, $V_{Th} = 0.90$ and $V_{Th} = 0.95$, respectively.

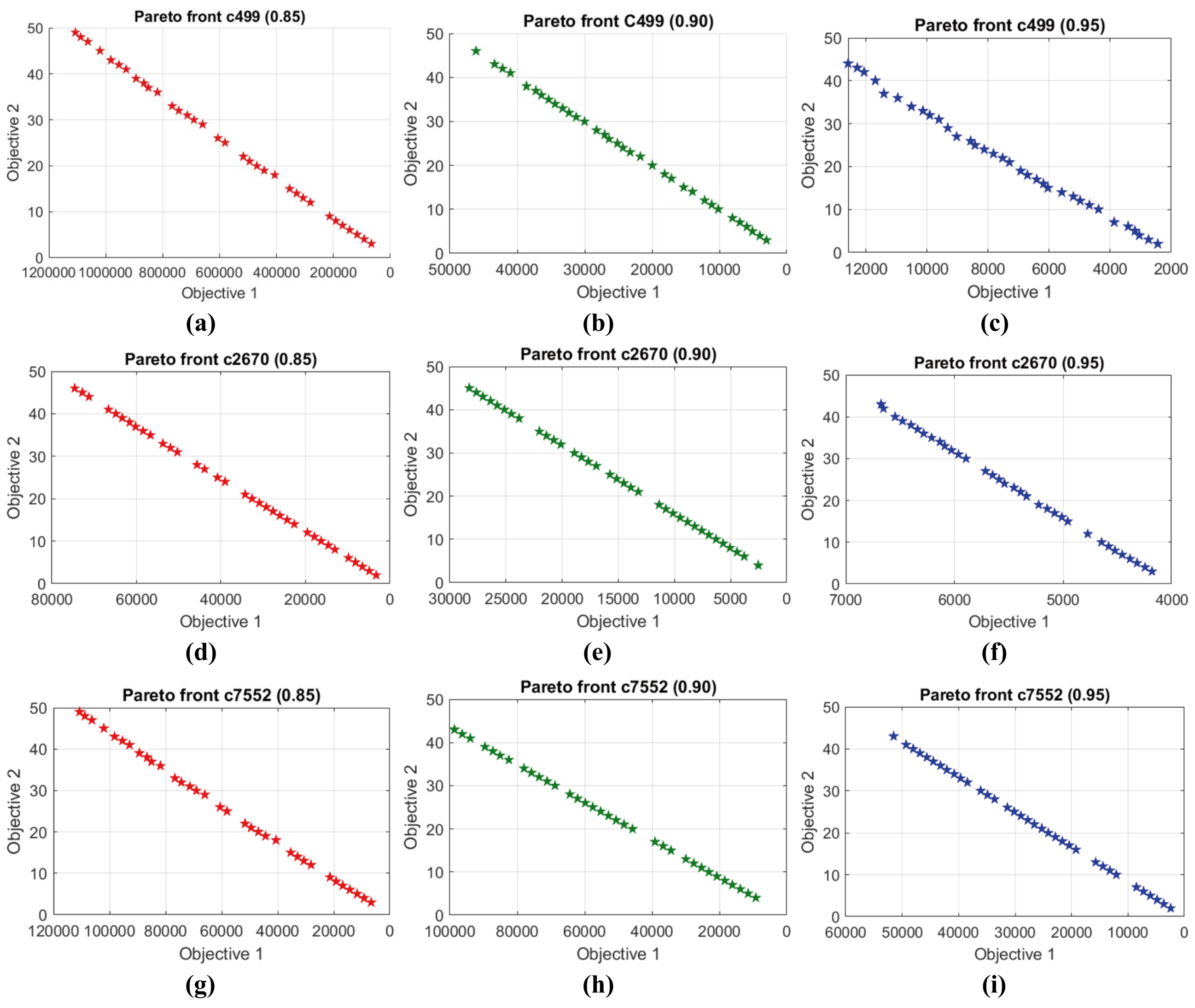

Fig. 6 Pareto charts of c499, c2670 and c7552 benchmarks generation of after exploring optimal solutions with $V_{Th} = 0.85$, $V_{Th} = 0.9$ and $V_{Th} = 0.95$

Table 1 The number of rare-trigger nodes in different benchmark circuits for different V_{Th}

Circuit	Number of Nets above the V_{Th}		
	V_{Th}= 0.85	V_{Th}= 0.90	V_{Th}= 0.95
c432	36	9	0
c499	48	40	40
c880	71	49	35
c1355	112	112	104
c1908	109	104	103
c2670	94	49	19
c3540	371	288	151
c5315	92	69	36
c7552	241	214	129

Though the above results of trigger coverage are presented for fixed or a maximum of 50 patterns, it is observed from the simulation that a slight increase in test patterns significantly increases the trigger coverage. The relation between the trigger coverage and the test cases for the c1908 benchmark is presented in Fig. 7. It can be observed from the figure that every increase in test patterns approximately doubles the trigger coverage each time. Since this figure presents the coverage for varying test patterns, it also shows maximum coverage for the given number of patterns. Hence, the proposed technique can also help the designer to achieve high trigger coverage according to the limit of test cases or the minimum number of test cases.

Besides the above, we also identify the optimal solution set for each benchmark, which exhibits maximum coverage

Table 2 Simulation results after exploring 35 optimal Pareto solutions for V_{Th} = 0.85

Circuit	Solutions and Total Cases		Coverage Criteria		Test case Criteria		Average Coverage and Test cases	
	Solutions	Test patterns	Max Coverage	Test Cases	Coverage	Min test Cases	Avg. Coverage	Test Cases
c432	35	927	478	46	146	2	243.74	26.49
c499	35	804	3.78E+09	45	2.35E+06	6	8.74E+07	22.97
c880	35	887	4.33E+10	44	2.97E+08	8	4.23E+09	25.34
c1355	35	703	972	48	346	3	643.17	20.09
c1908	35	794	7.85E+10	46	3.47E+08	7	7.86E+09	22.69
c2670	35	897	9.85E+07	44	3.25E+05	8	1.76E+06	25.63
c3540	35	689	3.25E+28	46	7.18E+12	6	3.46E+17	19.69
c5315	35	843	1.46E+09	47	3.48E+06	4	1.32E+08	24.09
C7552	35	923	6.67E+21	43	1.33E+09	6	3.22E+14	26.37

Table 3 Simulation results after exploring 35 optimal Pareto solutions for V_{Th} = 0.90

Circuit	Solutions and Total Cases		Coverage Criteria		Test case Criteria		Average Coverage and Test cases	
	Solutions	Test patterns	Max Coverage	Test Cases	Coverage	Min test Cases	Avg. Coverage	Test Cases
c432	35	807	383	41	95	7	113.63	23.06
c499	35	743	137	43	47	4	87.38	21.23
c880	35	872	2.15E+09	45	1.37E+06	6	2.61E+07	24.91
c1355	35	679	876	47	257	3	387.45	19.4
c1908	35	759	5.34E+08	44	1.70E+07	6	3.41E+07	21.69
c2670	35	855	9.24E+06	45	4.52E+04	4	8.86E+04	24.43
c3540	35	654	3.92E+27	46	2.1E+12	5	1.65E+17	18.69
c5315	35	807	4.19E+07	42	2.81E+05	5	1.05E+06	23.06
c7552	35	789	5.45E+19	48	1.45E+07	7	2.17E+12	22.54

per test pattern and compare it with the existing MERO [4], GA [24], and AGA [18] based test pattern generation techniques as shown in Table 5. It can be observed from this table that the proposed technique provides higher trigger coverage with a low number of patterns over MERO and both the existing GA based techniques. However, the

Table 4 Simulation results after exploring 35 optimal Pareto solutions for V_{Th} = 0.95

Circuit	Solutions and Total Cases		Coverage Criteria		Test case Criteria		Average Coverage and Test cases	
	Solutions	Test patterns	Max Coverage	Test Cases	Coverage	Min test Cases	Avg. Coverage	Test cases
c499	35	721	129	41	78	3	83.45	20.6
c880	35	783	1.18E+07	41	2.63E+05	6	5.53E+06	22.37
c1355	35	855	567	42	233	4	407.24	24.43
c1908	35	806	4.85E+07	47	1.30E+05	5	2.73E+06	23.03
c2670	35	835	6.75E+04	42	1.18E+03	7	2.09E+03	23.86
c3540	35	870	7.12E+21	42	1.87E+12	6	3.42E+16	24.86
c5315	35	853	7.32E+07	41	4.89E+04	6	2.32E+06	24.37
c7552	35	827	4.67E+17	42	1.89E+06	4	1.93E+10	23.63

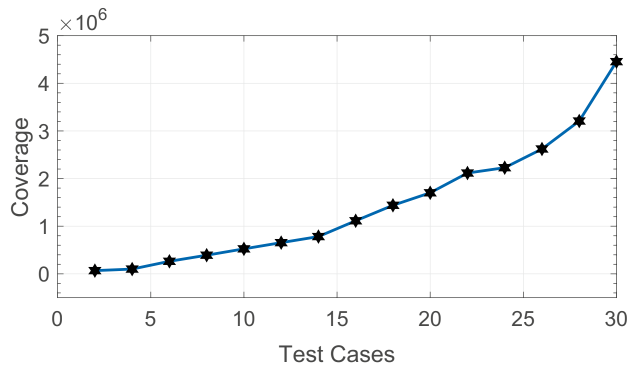

Fig. 7 Coverage progress with respect to test patterns for c1908 ISCAS-85 benchmark circuit

coverage of the proposed technique for some benchmarks is less than the existing techniques in the case of Vth = 0.9. This is because the number of patterns in the proposed method is less. However, on average, the proposed method provides high trigger coverage, as shown in the last row of this table. Additionally, the proposed technique provides high trigger coverage in most benchmarks over the existing methods for Vth = 0.95. On average, the proposed technique achieves 4.4E+26 and 8.9E+20 coverage only with 17.6 and 17.9 test patterns for $V_{Th} = 0.90$ and $V_{Th} = 0.95$, respectively. Whereas best known existing techniques, *i.e.*, AGA on an average provides only 3.4E+23 and 1.7E+20 coverage while requiring 80.6 and 58.9 test cases for $V_{Th} = 0.9$ and $V_{Th} = 0.95$ respectively.

Though the proposed technique only reduces around 60 and 40 patterns over the existing methods, it provides significantly high coverage over the existing methods. Since each

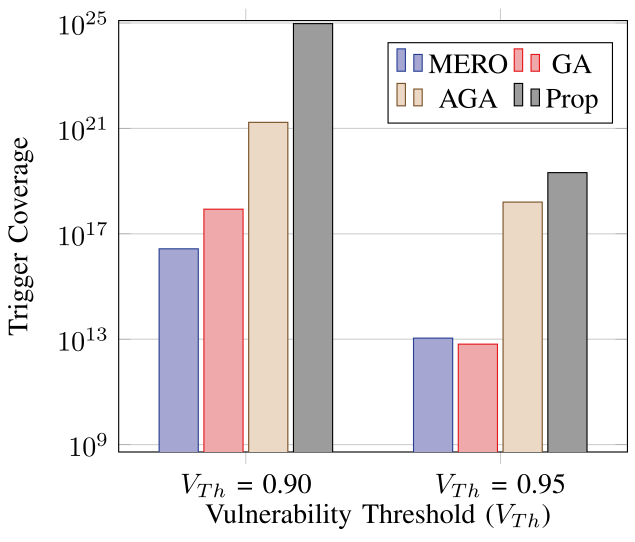

Fig. 8 Comparison of average coverage of proposed and existing HT detection techniques for different V_{Th} values

Table 5 Comparison of proposed multi-objective NSGA and existing GA based Techniques

Circuit	$V_{Th} = 0.90$								$V_{Th} = 0.95$							
	Test cases				Coverage				Test cases				Coverage			
	MERO	GA	AGA	Prop	MERO	GA	AGA	Prop	MERO	GA	AGA	Prop	MERO	GA	AGA	Prop
c432	31	7	19	2	167	16	288	37	–	–	–	–	–	–	–	–
c499	32	32	32	7	72	72	72	87	32	32	32	3	72	72	72	78
c880	80	17	6	11	1.3E+09	2.1E+07	1.2E+07	2.3E+07	31	12	3	9	3.3E+05	45525	1.7E+05	1.5E+06
c1355	78	25	80	8	695	266	704	287	64	22	64	14	328	134	328	315
c1908	53	36	33	9	3.3E+07	1.1E+07	2.1E+08	7.7E+07	55	35	33	11	4.6E+07	5.7E+06	1.1E+08	3.2E+07
c2670	87	11	51	29	5.3E+06	1.5E+06	8402196	7.2E+06	8	3	9	28	3019	645	1297	1.0E+04
c3540	565	65	197	46	1.4E+20	4.9E+20	3.1E+24	3.9E+27	287	44	103	42	2.3E+16	2.3E+15	1.3E+21	7.1E+21
c5315	135	48	108	19	1.5E+07	207734	2.9E+06	9.9E+05	44	17	29	13	15417	1162	122542	8.9E+05
c7552	850	43	199	27	1.1E+11	4.6E+18	9.9E+19	3.8E+19	363	21	198	23	1.5E+07	2.8E+11	4.4E+14	2.7E+15
Average	212.3	31.6	80.6	17.6	1.5E+19	5.6E+19	3.4E+23	4.3E+26	110.5	23.3	58.9	17.9	2.9E+15	2.9E+14	1.7E+20	8.9E+20

Table 6 Comparison of Coverage Per Test Case

Circuit	$V_{Th} = 0.90$				$V_{Th} = 0.95$			
	MERO	GA	AGA	Prop	MERO	GA	AGA	Prop
c432	5.4	2.29	15.16	18.5	–	–	–	–
c499	2.3	2.25	2.25	12.43	2.3	2.25	2.25	26
c880	1.60E+07	1.25E+06	2.00E+06	2.13E+06	1.06E+4	3.79E+03	5.73E+04	1.63E+05
c1355	8.9	10.64	8.8	35.88	5.1	6.09	5.13	22.5
c1908	6.24E+05	2.95E+05	6.37E+06	8.54E+06	8.29E+05	1.63E+05	3.21E+06	2.95E+06
c2670	6.12E+4	1.33E+05	1.65E+05	2.50E+05	377.4	215	144.11	358.57
c3540	2.44E+17	7.65E+18	1.56E+22	8.52E+25	8.07E+13	5.20E+13	1.30E+19	1.70E+20
c5315	1.14E+05	4.33E+03	2.67E+04	5.21E+04	350.4	68.35	4225.59	68230.77
c7552	1.27E+08	1.08E+17	4.98E+17	1.40E+18	3.99E+04	1.31E+10	2.22E+12	1.19E+14

pattern provides different coverage, we cannot compare the number of patterns required for achieving equal trigger coverage. But we have computed and compared the trigger coverage per pattern as shown in Table 6. The comparison of the proposed multi-objective optimization-based technique and exiting MERO and GA-based techniques for average trigger coverage per test pattern on different V_{Th} is shown in Fig. 8. It is clear from this figure that AGA based technique provides high trigger coverage over the GA and MERO based approaches. But the proposed technique provides significantly high trigger coverage over AGA based approach for $V_{Th} = 0.9$ as well as $V_{Th} = 0.95$. The AGA-based approach provides only 1.7E+21 coverage for $V_{Th} = 0.9$, whereas the proposed multi-objective optimization-based technique provides 9.4E+24 coverage. The proposed technique provides approximately 12 times more trigger coverage with 1/3 reduced test patterns on $V_{Th} = 0.9$ over the AGA-based technique. From the above analysis, it can be observed that the proposed multi-objective optimization-based technique outperforms the existing techniques and would be very effective in detecting the stealthy Trojan during post-silicon testing.

5 Conclusion

The existing GA based test pattern generation techniques suffer from low trigger coverage and require large patterns. As a solution, we framed the test pattern generation problem into multi-objective optimization and solved it through the NSGA-II algorithm. The proposed technique generates optimal test patterns for activating rare events to facilitate Trojan detection during post-silicon testing. Our NSGA-II based approach generates the test patterns based on Bi-Objective criterion *i.e.*, minimal test patterns and maximum coverage. Therefore, the proposed multi-objective-based method overcomes the limitations of existing GA-based methods and provides high coverage with a low number of patterns. Further, we also insert the checker logic at the low observable nodes in the design to observe the Trojan behaviour on its activation during post-silicon testing and during online monitoring. The simulation results on various ISCAS benchmarks show that the proposed multi-objective optimization-based technique provides, on average, 12× high trigger coverage while requiring 1/3 reduced test patterns over the best-known existing method.

In this paper, we have implemented the proposed approach only for the combinational benchmarks, i.e., c-series benchmarks, which can contain only combinational Trojans. Since sequential benchmarks (e.g., ISCAS-89 and ITC'99) may also contain sequential Trojans, implementing the current version directly on these benchmarks is challenging and may require significant time to generate the patterns. However, the proposed NS-GA method functions like the existing GA and Advised GA; thus, we consider scaling the implementation of the proposed method to generate the patterns for sequential benchmarks even for detecting sequential Trojans in our future work. Further, the multiple empirical runs to generate the optimal solution increases the cost of test generation. Therefore, our future work will also focus on reducing the test generation time.

Acknowledgements This research is supported by Data Security Council of India (DSCI) under the project titled "HT-Pred: A Complete Defensive Machine Learning Tool for Hardware Trojan Detection".

Data Availability This manuscript has no associated data.

Declarations

Conflict of Interest None.

References

1. Abdel-Basset M, Abdel-Fatah L, Sangaiah AK (2018) Chapter 10 - metaheuristic algorithms: A comprehensive review. In: Sangaiah AK, Sheng M, Zhang Z (eds) Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications. Intelligent Data-Centric Systems. Academic Press, pp 185–231. [Online]. https://www.sciencedirect.com/science/article/pii/B9780128133149000104

2. Bhunia S, Abramovici M, Agrawal D, Bradley P, Hsiao MS, Plusquellic J, Tehranipoor M (2013) Protection against hardware trojan attacks: Towards a comprehensive solution. IEEE Design & Test 30(3):6–17
3. Bhunia S, Hsiao MS, Banga M, Narasimhan S (2014) Hardware trojan attacks: threat analysis and countermeasures. Proceeding of the IEEE 102(8):1229–1247
4. Chakraborty RS, Wolff F, Paul S, Papachristou C, Bhunia S (2009) Mero: a statistical approach for hardware trojan detection. In: Proceedings on Cryptographic Hardware and Embedded Systems-CHES 2009. Springer, pp 396–410
5. Chakraborty RS, Bhunia S (2011) Security against hardware trojan attacks using key-based design obfuscation. J Electron Test 27(6):767–785
6. Chakraborty RS, Pagliarini S, Mathew J, Rajendran SR, Devi MN (2017) A flexible online checking technique to enhance hardware trojan horse detectability by reliability analysis. IEEE Trans Emerg Top Comput 5(2):260–270
7. Cui X, Koopahi E, Wu K, Karri R (2018) Hardware trojan detection using the order of path delay. ACM Journal on Emerging Technologies in Computing Systems (JETC) 14(3):33
8. Dong C, Xu Y, Liu X, Zhang F, He G, Chen Y (2020) Hardware trojans in chips: a survey for detection and prevention. Sensors 20(18):5165
9. Dupuis S, Ba P-S, Di Natale G, Flottes M-L, Rouzeyre B (2014) A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In: Proceedings on IEEE 20th International On-Line Testing Symposium (IOLTS). IEEE, pp 49–54
10. Goldstein LH, Thigpen EL (1980) Scoap: Sandia controllability/observability analysis program. In: Proceedings on 17th Design Automation Conference. pp 190–196
11. Haider SK, Jin C, Ahmad M, Shila DM, Khan O, van Dijk M (2014) Hatch: a formal framework of hardware Trojan design and detection. University of Connecticut Cryptology ePrint Archive Technical Report vol 943. p 2014
12. Hasegawa K, Oya M, Yanagisawa M, Togawa N (2016) Hardware trojans classification for gate-level netlists based on machine learning. In: Proceedings on 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, pp 203–206
13. Hicks M, Finnicum M, King ST, Martin MM, Smith JM (2010) Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In: Proceedings on IEEE Symposium onSecurity and Privacy (SP). IEEE, pp 159–172
14. Jin Y, Makris Y (2008) Hardware trojan detection using path delay fingerprint. In: Proceedings on IEEE International Workshop on Hardware-Oriented Security and Trust(HOST). IEEE, pp 51–57
15. Lin L, Burleson W, Paar C (2009) Moles: malicious off-chip leakage enabled by side-channels. In: Proceedings on International conference on computer-aided design. ACM, pp 117–122
16. Liu Y, He J, Ma H, Zhao Y (2020) Golden chip free trojan detection leveraging probabilistic neural network with genetic algorithm applied in the training phase. SCIENCE CHINA Inf Sci 63(2):1–3
17. Meyarivan T, Deb K, Pratap A, Agarwal S (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
18. Nourian M, Fazeli M, Hely D (2018) Hardware trojan detection using an advised genetic algorithm based logic testing. J Electron Test 34(4):461–470
19. Oya M, Shi Y, Yanagisawa M, Togawa N (2015) A score-based classification method for identifying hardware-trojans at gate-level netlists. In: Proceedings on Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium, pp 465–470
20. Pan Z, Mishra P (2021) Automated test generation for hardware trojan detection using reinforcement learning. In: Proceedings on 26th Asia and South Pacific Design Automation Conference. pp 408–413
21. Rathor VS, Garg B, Sharma G (2017) New light weight threshold voltage defined camouflaged gates for trustworthy designs. J Electron Test 33(5):657–668
22. Rathor VS, Garg B, Sharma GK (2020) A novel low complexity logic encryption technique for design-for-trust. IEEE Trans Emerg Top Comput 8(3):688–699
23. Rostami M, Koushanfar F, Karri R (2014) A primer on hardware security: models, methods, and metrics. Proc IEEE 102(8):1283–1295
24. Saha S, Chakraborty RS, Nuthakki SS, Anshul, Mukhopadhyay D (2015) Improved test pattern generation for hardware trojan detection using genetic algorithm and Boolean satisfiability. In: Proceedings on International Workshop on Cryptographic Hardware and Embedded Systems. Springer, pp 577–596
25. Salmani H, Tehranipoor M, Plusquellic J (2012) A novel technique for improving hardware trojan detection and reducing trojan activation time. IEEE Trans Very Large Scale Integr VLSI Syst 20(1):112–125
26. Salmani H (2017) COTD: reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. IEEE Trans Inf Forensics Secur 12(2):338–350
27. Shekarian SMH, Zamani MS, Alami S (2013) Neutralizing a design-for-hardware-trust technique. In: Proceedings on 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013). IEEE, pp 73–78
28. 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 Transactions on Embedded Computing Systems (TECS) 20(4):1–20
29. Sturton C, Hicks M, Wagner D, King ST (2011) Defeating UCI: building stealthy and malicious hardware. In: Proceedings on IEEE Symposium on Security and Privacy (SP). IEEE, pp 64–77
30. Waksman A, Suozzo M, Sethumadhavan S (2013) Fanci: identification of stealthy malicious logic using Boolean functional analysis. In: Proceedings on ACM SIGSAC Conference on Computer & Communications Security. ACM, pp 697–708
31. Xiao K, Forte D, Jin Y, Karri R, Bhunia S, Tehranipoor M (2016) Hardware trojans: lessons learned after one decade of research. ACM Transactions on Design Automation of Electronic Systems (TODAES) 22(1):6
32. Xie X, Sun Y, Chen H, Ding Y (2017) Hardware trojans classification based on controllability and observability in gate-level netlist. IEICE Electronics Express 14(18):20170682–20170682
33. Zhang J, Yu H, Xu Q (2012) HtOutlier: hardware trojan detection with side-channel signature outlier identification. In: Proceedings on IEEE International Symposium on Hardware-Oriented Security and Trust. IEEE, pp 55–58
34. Zhang J, Yuan F, Xu Q (2014) Detrust: defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In: Proceedings on ACM SIGSAC Conference on Computer and Communications Security. ACM, pp 153–166
35. Zhang J, Yuan F, Wei L, Sun Z, Xu Q (2013) Veritrust: verification for hardware trust. In: Proceedings on 50th ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, pp 1–8
36. Zhang J, Yuan F, Wei L, Liu Y, Xu Q (2015) Veritrust: Verification for hardware trust. IEEE Trans Comput Aided Des Integr Circuits Syst 34(7):1148–1161
37. Zhou Z, Guin U, Agrawal VD (2018) Modeling and test generation for combinational hardware trojans. In: Proceedings on IEEE 36th VLSI Test Symposium (VTS). IEEE, pp 1–6

Vijaypal Singh Rathor received his Ph.D. from ABV-Indian Institute of Information Technology and Management Gwalior and M. Tech. in department of Computer Science & Engineering from Maulana Azad National Institute of Technology, Bhopal, India in 2020 and 2014 respectively. At present, He is working as Assistant Professor in department of Computer Science & Engineering at PDPM Indian Institute of Information Technology, Design and Manufacturing (IIITDM), Jabalpur, India. Before, Joining IIITDM, he was working as an Assistant Professor at Thapar Institute of Engineering and Technology (Thapar University), Patiala, India. He has over three years of teaching experience to his credit. He has published over 10 research articles in International Journals/Conferences of repute. His research interests include trustworthy circuit design techniques to thwart hardware-based attacks, hardware Trojan detection, applying machine learning for digital designs, cloud computing.

Deepak Singh received the Ph.D. degree at Department of Computer Science and Engineering from National Institute of Technology Raipur. He is an Assistant Professor in Department of Computer Science & Engineering at National Institute of Technology (NIT), Raipur, India. Before, joining NIT, he was working as Assistant Professor with the School of Engineering and Applied Science, Bennett University Greater Noida. He has over six years' experience in various academic institutes. He has published over 10 referred articles and served as a reviewer of several journals. His research interests include Evolutionary Computation, Machine Learning, and Data Mining.

Simranjit Singh was born in Patiala, Punjab India. He graduated and received his M.Tech degree in 2015 in the field of Computer Science and Engineering from Punjabi University, Patiala. He has completed his Ph.D. from Thapar Institute of Engineering and Technology, Patiala, Punjab, India. He is working as Assistant Professor in the Computer Science Engineering department at Bennett University, Greater Noida. He has more than two years of teaching experience to his credit. He has published research articles in international journals/conferences of repute. His research interest is focused on Image Processing, particularly on the classification and mining of hyperspectral images.

Mohit Sajwan received his B.Tech. degree in Computer Science Engineering from Amrapali Institute of Technology, Uttrakhand Technical University, Dehradun and M.E. (software engineering) from Birla Institute of Technology, Mesra, Ranchi. He has completed his Ph.D. Degree in wireless sensor networks from National Institute of Technology, Delhi. Currently, he is working as an assistant professor at Bennett University, Greater Noida, India. He has also published research articles in international journals/conferences of repute. His research interest lies in the domain of energy-efficient routing protocol in wireless sensor networks.