An Evolutionary Approach to Hardware Encryption and Trojan-Horse Mitigation

Andrea Marcelli, Marco Restifo, Ernesto Sanchez, Giovanni Squillero

Politecnico di Torino
Corso Duca degli Abruzzi, Torino 10129, Italy
{andrea.marcelli, marco.restifo, ernesto.sanchez, giovanni.squillero}@polito.it

***Abstract*—New threats, grouped under the name of *hardware attacks*, became a serious concern in recent years. In a global market, untrusted parties in the supply chain may jeopardize the production of integrated circuits with intellectual-property piracy, illegal overproduction and hardware Trojan-horses (HT) injection. While one way to protect from overproduction is to encrypt the design by inserting logic gates that prevents the circuit from generating the correct outputs unless the right key is used, reducing the number of poorly-controllable signals is known to minimize the chances for an attacker to successfully hide the trigger for some malicious payload. Several approaches successfully tackled independently these two issues. This paper proposes a novel technique based on a multi-objective evolutionary algorithm able to increase hardware security by explicitly targeting both the minimization of rare signals and the maximization of the efficacy of logic encryption. Experimental results demonstrate the proposed method is effective in creating a secure encryption schema for all the circuits under test and in reducing the number rare signals on six circuits over nine, outperforming the current state of the art.**

I. Introduction

Due to the high costs involved in the manufactoring process of Integrated Circuit (ICs), most of the semi-conductor companies prefer to design only their Intellectual Properties (IPs), and outsource to a third-party company the fabrication phase. Moving the manufactoring process to low-cost areas has become a major trend in the last decade in IC industry. This radical change raises the concerns about untrusted foundries and hardware piracy: the thefts of masks for unauthorized overproduction [1] [2], as well as the insertion of Hardware Trojan-horses (HTs), malicious circuitry, or alterations [3] [4] became a major cause of concern. Since the fabrication process becomes untrusted, IC vendors face the challenges of protecting and verify the trustworthiness of their devices. It is estimated that the semi-conductor industry loses about $4 billion annually due to these attacks [1] [2].

According to a model presented in [5], the HT conceals its malicious behavior during normal operations; it monitors specific circuit inputs or internal signals, and activates the payload only when a predefined pattern is detected. Logic testing is customarily performed before deployment in the field, and it can be used in order to detect the presence of potential HTs: if an erroneous behavior of the IC is observed, it can be inferred that a HT has been inserted in the IC. However, to decrease the HT's chance of being activated during such logic test, usually the trigger is connected to signals with very low controllability, and the payload is connected to signals with very low observability. As traditional ATPG test vectors may not be sufficient to detect threats, such logic testing is required to use test vectors that maximize the chances of activation. Specific *design-for-trust* techniques consist in incorporating into the ICs features that specifically improve the HT detectability [6] [7].

On the other hand, a common way to protect ICs from illegal overproduction is to encrypt the design so that only authorized users are able to use its functionalities: a lock is added to the circuit in the form of additional inputs, and users are required to know the correct key. Logic encryption techniques can be broadly classified either as *sequential* or *combinatorial*. In the former, additional logic states are introduced in the state transition graph in such a way that the circuit reaches a valid state only whether the correct sequence of key vectors is applied. In the latter, all outputs are garbled unless the values on the extra inputs do not match a correct key vector. Combinational logic encryption can be easily applied to any kind of circuits after the design phase, as it does not require analyses of the states transition graph.

This paper mixes combinational logic encryption with design-for-trust, and proposes an approach able to tackle illegal overproduction and, at the same time, to improve HT detectability. The proposed methodology exploits a multi-objective Evolutionary Algorithm (EA) to maximize the effectiveness of encryption, maximize HT detectability, and minimize the number of additional inputs. That is, the evolutionary algorithm inserts new inputs and gates in order to guarantee an average 50% Hamming distance against correct outputs when a wrong key is used, and to minimize the number of signals that almost always have the same value (the so-called *rare* signals).

Experimental results demonstrate the effectiveness of the proposed approach: the multi-objective EA outperforms previous works; the selected changes have a reduced overhead both in terms of area occupation and slack time. Moreover, the modifications do not change the functionality of the design, while they do represent a strong deterrent to malicious attacker.

The rest of the paper is organized as follows. Section II provides the necessary background about logic encryption. Section III introduces the previous works highlighting their

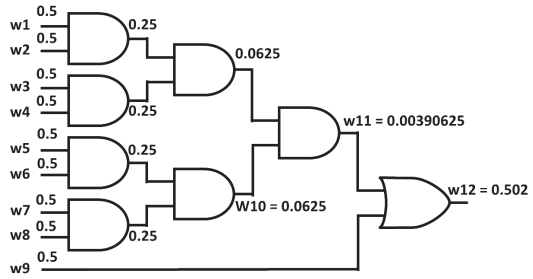
978-3-9815370-8-6/17/$31.00 ©2017 IEEE

advantages and disadvantages and motivates the necessity of the proposed technique. Section IV introduces the multi-objective evolutionary algorithm for gate injection and further discusses possible security limitations, while section V shows the effectiveness of the proposed method on a benchmark of combinational circuits. Section VI concludes the paper.

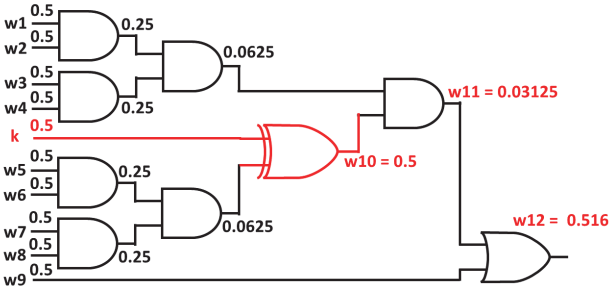
II. Logic Encryption

The goal of logic encryption is to protect ICs from mask theft and overproduction, preventing the unauthorized use. A lock is added to the circuit in the form of additional inputs, and only authorized users who know the key are able to use its functionalities.

In sequential logic encryption, the protection is applied on the state transition graph by adding extra *invalid* states. Only the key sequence is able to drive the circuit to valid and working states. Conversely, in combinational logic encryption, some XOR/XNOR, AND/OR, or multiplexer *key gates* are inserted in the circuit, each one connected with an extra input so that the correct key neutralize its effect. An incorrect input, on the other hand, changes the circuit functionality.



(a)



(b)

Fig. 1: Combinational logic encryption example

Figure 1 shows an example of a possible sequence of logic encryption: the original circuit is shown in Figure 1a, while the encrypted version is shown in Figure 1b. The figures also show for every net in the circuit its probability to be **1**, assuming unbiased inputs. The insertion of the XOR key gate on the signal w10 in Figure 1b increases the probability of the signals w11 and w12, assuming that k, the new input, has 50% of probability to be **1**.

III. Previous Works

The combinational encryption technique has been originally proposed in [1], and it consists on the random insertion of XOR/XNOR gates into the design. As described in the previous section, a new Primary Input is connected to one of the inputs of the inserted key gate. In order to functionaly activate a chip, the manufacturer must send its ID to the holder of the IP rights, who then sends the key that only activates that precise chip. This process allows the IP-holder to control exactly how many chips are made and prevents others from making functional copies.

An alternative approach is presented in [2]; it improves the previous technique by ensuring that wrong keys corrupt the outputs impairing functionality. The approach correlates logic encryption to IC testing, and proposes a logic encryption technique based on the analysis of fault propagation: XOR/XNOR gates are carefully placed to achieve an average 50% Hamming distance between the circuit outputs using a wrong key with respect to the expected ones. Furthermore, the result is achieved adding a smaller number of gates compared to previous method.

The reduction of rare signals through logic encryption is proposed in [8]. As in this paper, it is assumed that an attacker tries to exploit signals with low controllability in order to hide the HT. The probabilities of each signal to be **1** are changed without affecting the slack time through the insertion of AND/OR gates in the design following a sharp algorithm developed by the authors. However, the method efficacy in terms of average Hamming distance between the outputs produced with and without the correct key is never evaluated.

Rajendran, et.al in [9] presented a technique based on fault analysis that can be used to improve the logic encryption by reaching the 50% Hamming distance. The main idea is to find the circuit locations where adding a fault implies a change of 50% of the outputs. This methodology inserts XOR/XNOR key gates at the location with the highest fault impact. In this way, an invalid key likely has the most impact on the circuit outputs.

Finally, the so-called *Enhanced Logic Encryption* [10] uses 128-bit keys to obfuscate the circuits in a similar way to the one proposed in [8], but it inserts XOR/XNOR gates instead of AND/OR gates. In more details, at every iteration up to the length of the encryption key it inserts a new key gate on the rarest signal, and then recomputes the probabilities of all signals.

IV. Proposed Approach

This paper introduces a new methodology based on a multi-objective evolutionary algorithm able to improve the circuit security level.

The method proposed herein is based on the following assumptions:

- Rare signals are prone to be used as triggers for HTs; then, the higher is the activation probability of all the

circuit signals, the lower is the risk that rare signals be used as HT triggers.
- An encrypted circuit should provide as output wrong values 50% different to the expected ones if activated with a wrong key.

In the proposed approach, the original circuit is modified by inserting XOR/XNOR gates conforming an encryption key in some of the circuits nets. The implemented modifications are oriented to improve the circuit security by tackling the following goals:

- Eliminating rare signals:
  elimination of places where a HT can be inserted.
- Inserting a secure logic encryption key:
  improving the circuit security by the inclusion of a secure code for correct elaboration.
- Minimizing of the number of input values of the key:
  reducing the hardware overhead due to the key insertion.

Optimizing these goals is a non-trivial task since they are often contradictory. For instance, minimizing the number of bits in the cryptographic key may negatively impact the reduction of rare signals, since every new key gate contributes to an incresae in the activation probability of the modified signal, eliminating, in this way the tackled rare signal. Accordingly, the proposed approach optimizes the three goals using a multi-objective evolutionary algorithm.

The whole process starts with the circuit translation from the original circuit description to the internal format. Then, exploiting a multi-objective evolutionary algorithm the circuit is modified by adding the encryption key. The steps to achieve this are described in the next section.

A. Circuit description

The combinational circuits addressed in these experiments are described in an internal format that allows the evolutionary core to modify every circuit net. In this way, the evolutionary tool is able to modify any net in the circuit by adding one of the two possible gates (XOR or XNOR) as previously described in section III. Thus, three different values may be assigned to each net:

- 0: the net is not modified;
- 1: a XOR gate is added in the considered net;
- 2: a XNOR gate is added in the considered net.

As previously mentioned, the insertion of these gates leaves the original circuit functionality unchanged, while affecting only the net probabilities.

B. Multi-Objective Evolutionary Algorithms

Many real-life optimization problems have more than one aspect to be considered at the same time, i.e., more than one objective to be minimized or maximized. The simplest possible solution, known as *scalarization approach*, is to aggregate all objectives with a single function, such a weighted sum; an alternative solution, *lexicographical approach*, puts the objectives in some order of preference. Both approaches require a domain expert to prioritize the objectives, and the introduced bias might be harmful if the objectives are conflicting. Moreover, such approaches do not provide a set of solutions that trade off between the different goals, but rather they tend to offer few solutions not homogeneously distributed in the objective space.

Unlike the scalarization and lexicographical, *multi-objective* algorithms [11] are based on the assumption that all the objectives of a problem should be considered at the same time. In other words, in order to capture the complexity of the search space of conflicting objectives, an algorithm should generate a Pareto front, that is, a set of solutions such that none of the objective functions can be further improved without degrading some other objective value. The user is eventually presented the set of *non-dominated solutions*, each one corresponding to a different trade-off between the different, conflicting goals. Due to their versatility, evolutionary algorithms have been demonstrated a valid tool for solving many hard, real-world multi-objective problems [12].

In the proposed experiments, we specialize a general-purpose evolutionary algorithm called μGP (also spelled *MicroGP*). The program was originally devised to evolve assembly-language programs for test program generation, and later expanded to a general-purpose toolkit[1] and exploited for several applications [13]. The μGP framework can be easily coupled with an external evaluator, which makes the integration with a simulator straightforward. μGP features built-in support for two or more fitness values, which can be evaluated either in lexicographical order or with a multi-objective approach.

The multi-objective algorithm makes use of a *crowding distance* in order to push the exploration of the Pareto front: for each individual, the closest neighbors are located on the same front in the fitness space [14]; such neighbors are then used as vertexes to compute the hyper-volume of the resulting cuboid; individuals associated with larger hyper-volumes will be favored for reproduction. The underlying idea is that individuals that are the farthest from others are positioned on segments of the Pareto front not yet well explored. By favoring such individuals, an even exploration of the Pareto front is promoted.

C. Evolutionary process

The evolutionary core starts the evolution creating a random population of individuals that represent the modified circuit following the previous considerations. Then, the circuit is transformed again into a simulatable format, and a logic simulator is used to compute the different fitness functions.

At the end of the process, the evolutionary core provides as output the Pareto front of individuals. In this way, the user may evaluate which is the solution that better fits the circuit requirements and use this as final solution.

[1] μGP is available under the *GNU Public License* and it hosted on SourceForge at *http://ugp3.sourceforge.net/*

D. Fitness functions

Three different fitness functions are evaluated for every candidate.

1) Key length: This fitness function provides the evolutionary core with the number of bits composing the encryption key. This value should be minimized.

2) Hamming distance: The level of goodness of the circuit encryption key is evaluated here by computing the Hamming distance between the circuit outputs in the presence of a wrong key and the expected ones. For every circuit, a significant set of inputs values, as well as a predetermined set of wrong keys are used in order to compute the circuit answers. Once computed the circuit answers, the exponential average of the Hamming distance between the circuits outputs with respect to the expected ones is computed $\overline{HD}_e = \frac{1}{n}\sum_{i=1}^{n} e^{d_i}$, where n is the number of output of the circuit and d_i is the average Hamming distance between the expected value and the one obtained when a wrong key is provided.

The goal here is to reach an exponential average of 50% on the considered set of inputs. The idea behind this fitness function is to obtain a smooth variation of all the wrong circuit answers around the 50% of the Hamming distance with respect to the expected ones.

3) Signals probabilities: The third fitness value considers the rare signals by computing the exponential average of the signals probability of the whole circuit. This average value helps the evolutionary process to eliminate most of the rare signals in the considered circuit.

The third fitness component is the exponential average signal probability $\overline{SP}_e = \frac{1}{n}\sum_{i=1}^{n} e^{p_i}$, where n is the number of output of the circuit and p_i is the signal probability of every net in the circuit assuming unbiased inputs.

E. External evaluator

A logic simulator is used in order to compute the stated fitness values; the simulator analyzes the new version of the circuit and extracts the signal probabilities of the considered circuit. Additionally, a set of predefined experiments are elaborated using the set of input values, and wrong keys to obtain the required values to compute the Hamming distance values. Finally, the gathered values are used to compute the three fitness values presented in the previous section.

V. Experimental Results

The proposed approach has been experimentally evaluated on a set of combinational circuits that belong to the well-known ISCAS-85 benchmark. These circuits have also been used in some of the previous works detailed in section III. Table I provides detailed information about them.

The first column of Table I reports the circuit name, the second the number of Primary Inputs (*PI*), the third the number of Primary Outputs (*PO*), while the fourth the number of Logic Gates (*LOGIC GATES*). The final two columns show the number of Rare Signals (*RS*) assuming a probability threshold equal to 0.20 and 0.01.

TABLE I: Combinational circuit description

Circuit	PI	PO	LOGIC GATES	RS 0.20	RS 0.01
c432	36	7	160	67	0
c499	41	32	202	48	32
c880	60	26	383	152	27
c1355	41	32	546	122	96
c1908	33	25	880	152	98
c2670	233	140	1193	311	24
c3540	50	22	1669	561	45
c5315	178	123	2307	552	8
c6288	32	32	2406	30	17

The proposed approach has been implemented using μGP as multi-objective evolutionary optimizer. For the circuit evaluation, an open source logic simulator was slightly modified. The logic simulator, written in *Phyton*, has been developed starting from the project combinational-logic-simulator by Bryrong Phung, freely available in GitHub [2]. Additional scripts, also developed in *Phyton*, were used to support the evolutionary process: these provide the interface between the evolutionary engine and the circuit simulator and are in charge of the post-processing of the raw data generated by the simulator. Finally, all performed experiments were run on a workstation based on 2 Intel Xeon E5450 CPUs running at 3.00 GHz, equipped with 64 GB of RAM.

A. Evolutionary process

The evolutionary process for every circuit was characterized by the following parameters: a population size of $\mu = 100$ individuals, $\lambda = 100$ new individuals in each generation, with a standard, fixed selective pressure of $\tau = 2$. The evolutionary algorithm runs until it reaches the maximum number of generations $G_{\max} = 10,000$ or 12 hours have elapsed.

The genome of individuals created by μGP are a list of the same length as the number of gates of the circuit under test. Each locus may contain three different alleles: a `XOR` or a `XNOR`, to indicate that a gate of that type must be added to the circuit as a key gate; a `void`, to denote that no gate needs to be added. Alleles containing `void` are more frequent in the initial population.

B. Evaluation of the proposed method

Figure 2, Figure 3 and Figure 4 show the evolution of the Pareto front in seven eras along the evolution process: the first generation; the last; and five relevant moments, depending on the length of the experiment. Different evaluation criteria are shown in each of them: average Hamming distance, exponential average of the number of rare signals and number or Rare Signals with a threshold probability of 0.01. While at the beginning individual are sparse, with the ongoing evaluations those concentrate in small area that at the end is identified as the Pareto front.

Figures 5 and Figure 6 compare the pareto front from the last generation for six circuits, showing both the Hamming

[2] *https://github.com/ByronPhung/combinational-logic-simulator*

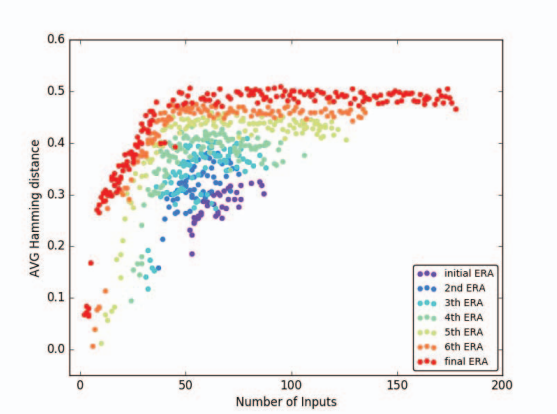


Fig. 2: Evolution over generations of the average Hamming Distance for c499 circuit.

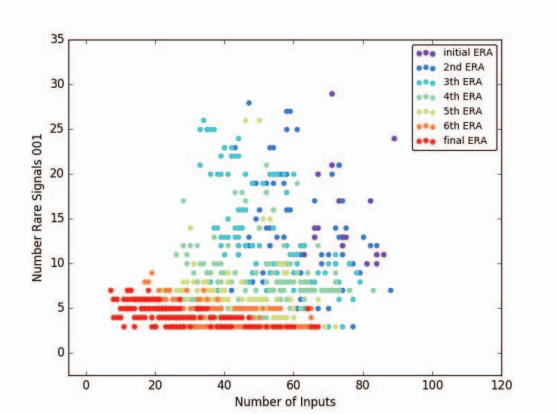


Fig. 4: Evolution over generations of the number of Rare Signals with prob=0.01 for c2670 circuit.

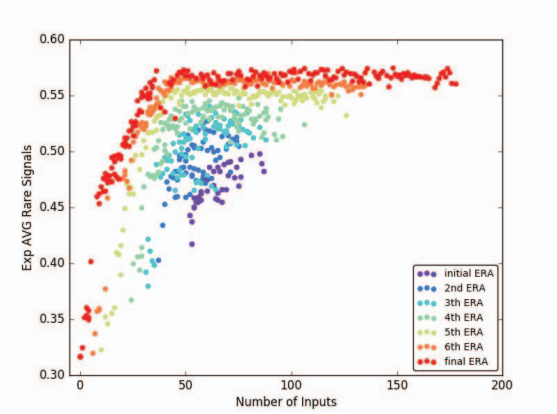


Fig. 3: Evolution over generations of the exponential average of Rare Signals with prob=0.01 for c499 circuit.

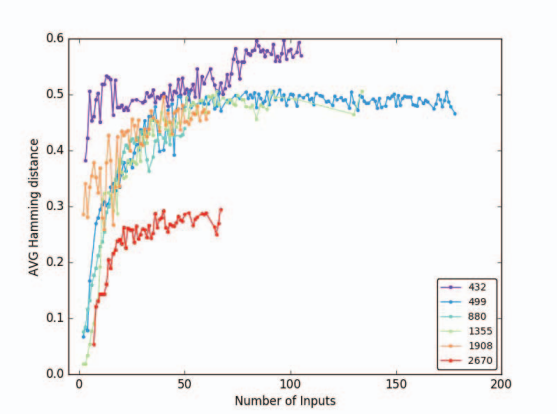


Fig. 5: Average hamming distance for ISCAS-85 combinatorial circuit.

Distance and the number of Rare Signals with a threshold probability of 0.01. Interestingly it is possible to notice that an incresing the length of the key allows it to reach the desired behaviour, while at one point the values do not change anymore and remains steady.

Table II compares the results related to the evalution of the Hamming Distance. Recalling that both 0% and 100% Hamming distances imply the least and 50% implies the most obfuscation, the proposed method has an average Hamming distance of 44.67%, which results in an improvement over the current state of the art approach [10].

Table III compares the lowest number of Rare Signals with probability bellow 0.01 and their related key length. The proposed approach completely outperforms previous result, showing the effectiveness of the evolutionary algorithm.

Table IV compares the lowest number of Rare Signals with probability bellow 0.20 and their related key length. The proposed approach outperforms [10] and shows comparable results with current state of the art result [8] except for the circuits c2670, c3540, c5315. Due to the large number of gates in these circuits, the evolutionary engine requires more time to achieve a local optimum solution. In fact, a possible disadvantage of the proposed method may be the required execution time that mainly depends on the maximum number of individuals evaluated, while in the state of the art proposals, it grows up linearly with the number of Rare Signals. However, the reader may notice that parallelizing the individuals evaluation the consumed time may be reduced considerably. Finally, for the best of our knowledge, there is no information about the optimum solutions for the performed experiments, then, it is not possible to state that the proposed approach found them.

VI. Conclusion

This paper proposes an evolutionary-based technique able to improve the security of combinational circuits by tackling two main problems at the same time: first, mitigating the circuit rare signals prone to be used as Hardware Trojan triggers; and second, the development of a strong enough logic encryption key to prevent unauthorized usage and avoid overproduction. The experimental results demonstrate that the proposed method is able to significantly reduce the number

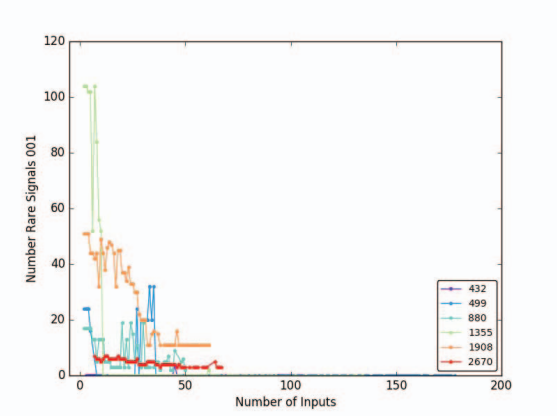


Fig. 6: Number of Rare Signals for ISCAS-85 combinatorial circuit.

TABLE II: Comparison with [8] and [10] of best solution with the closest HD equal to 50%

Circuit	[8]				[10]		Proposed Method	
	prob=0.01		prob=0.20					
	HD	key	HD	key	HD	key	HD	key
c432	0	0	4	1	50	60	50	48
c499	47	31	1	3	30	47	50	97
c880	4	7	11	49	34	116	44	50
c1355	50	60	1	5	43	95	50	51
c1908	0	0	4	65	48	123	50	40
c2670	1	12	8	110	9	109	29	40
c3540	12	15	20	52	30	125	50	23
c5315	2	7	10	98	14	127	31	129
c6288	24	17	25	28	47	67	48	50

TABLE III: Comparison with [8] of lowest number of Rare Signals with probability bellow 0.01 and their related key

Circuit	Original	[8]		Proposed Method	
	#RS	#RS	Key	#RS	Key
c432	0	0	0	0	2
c499	32	0	0	0	4
c880	27	30	27	2	30
c1355	96	48	56	0	11
c1908	98	186	98	11	19
c2670	24	24	26	3	11
c3540	45	27	45	4	22
c5315	8	26	8	12	38
c6288	17	16	17	0	0

of rare signals in most of the considered circuits, and even completely remove them in some cases.

Regarding the second goal, the approach is able to insert logic keys causing the output of the circuit to be significantly different from the correct one whenever a wrong key is used. The proposed method is experimentally evaluated using a classical benchmark of circuits and in most of the cases outperforms the state of the art approaches.

Considering the performed experiments, it is possible to state that the proposed approach may scale well for larger circuits than the ones used here by simply parallelizing the individual evaluations.

Authors are currently improving the proposed method, in particular by experimenting different setups of the evolutionary core aiming at reducing the number of rare signals on the largest circuits, while reducing the generation time. In addition, the method will be improved in order to deal with sequential circuits.

TABLE IV: Comparison with [8] and [10] of lowest number of Rare Signals with probability bellow 0.20 and their key.

Circuit	Original	[8]		[10]		Proposed Method	
	#RS	#RS	Key	#RS	Key	#RS	Key
c432	67	50	63	0	60	3	53
c499	48	68	48	0	48	2	40
c880	152	101	128	0	128	2	110
c1355	122	186	128	0	128	2	120
c1908	152	199	128	7	128	13	121
c2670	311	142	128	3	128	*7*	*251*
c3540	561	223	128	4	128	*358*	*36*
c5315	552	134	128	5	128	*504*	*129*
c6288	30	112	128	16	128	10	36

ACKNOWLEDGMENT

Part of this work was done at the Joint Open Lab SWARM and was supported by a fellowship from TIM.

REFERENCES

[1] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1069–1074.

[2] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 953–958.

[3] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 15–19.

[4] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[5] S. Dupuis, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "Identification of hardware trojans triggering signals," in *First Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*, 2013.

[6] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 113–116.

[7] H. Salmani, M. Tehranipoor, and J. Plusquellic, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[8] S. Dupuis, P.-S. Ba, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*. IEEE, 2014, pp. 49–54.

[9] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[10] M. S. Samimi, E. Aerabi, Z. Kazemi, M. Fazeli, and A. Patooghy, "Hardware enlightening: No where to hide your hardware trojans!" in *On-Line Testing and Robust System Design (IOLTS), 2016 IEEE 22nd International Symposium on*. IEEE, 2016, pp. 251–256.

[11] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[12] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing (2nd edition)*. Springer, 2015, vol. 53.

[13] E. Sanchez, M. Schillaci, and G. Squillero, *Evolutionary Optimization: the μGP toolkit*. Springer Science & Business Media, 2011.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.