Robust Logic Obfuscation for Trusted Design Synthesis

Sanjana Sekar
Electrical Engineering and
Computer Science Department
University of Cincinnati
Cincinnati, Ohio 45221, USA
Email: sekarsa@mail.uc.edu

Ranga Vemuri
Electrical Engineering and
Computer Science Department
University of Cincinnati
Cincinnati, Ohio 45221, USA
Email: ranga.vemuri@uc.edu

***Abstract*—Logic obfuscation/encryption/locking refers to a class of techniques to prevent IC counterfeiting, reverse engineering, piracy, unauthorized over production or use. In these methods, a logic design is encrypted during or after logic synthesis and a key is generated. The encrypted logic design may be fabricated at a foundry which is untrusted. The key generated at the time of encryption is provided to a trusted user and should be suppled during operation. Valid outputs are produced by the IC if and only if a valid key is supplied. In the past few years, several proposed logic obfuscation methods were shown to be vulnerable to a variety of attacks including the satisfiability attack, skew attack, logic removal attack etc. Further, it is hard to integrate some of the proposed obfuscation methods with standard synthesis tools and methodologies.**

In this paper, we report our efforts to develop robust obfuscation methods compatible with COTS logic synthesis tools. We explore the use of sub-circuit obfuscation using look-up tables and logic duplication to improve the quality of encryption. Experimental results show negligible area overhead and robust encryption for several benchmark circuits. In addition, we discuss how these methods can be integrated with a COTS logic synthesis tools.[1]

I. Introduction

Fabless design approach has become an integral part of semiconductor industry due to increase in design complexity and maintenance costs of fabrication facilities. Outsourcing of IC fabrication to a potentially untrusted foundry allows an attacker to exploit the layout information for nefarious purposes. In addition, attackers can reverse engineer the design through depackaging, delayering, imaging the individual fabrication layers, gathering all images through stitching and extracting the netlist [1]. A design sent to an untrusted foundry is vulnerable to the addition of malicious circuits (known as Hardware Trojans) causing malfunctioning of circuits [2]. They are also vulnerable to piracy resulting in cheaper versions of the same design in the black market [3]. A miscreant from an untrusted foundry can also reverse engineer the functionality of the IP and claim false ownership of it [4]. Some of the significant supply chain attacks are reverse engineering, chip counterfeiting, Intellectual Property (IP) piracy, physical attack, IC overbuilding and insertion of Hardware Trojans [1], [2], [5]. This has prompted researchers to come up with various Design-For-Trust (DfTr) [1] techniques as countermeasures to such attacks. One such approach is *Logic Encryption.*

[1]STATEMENT A. Approved for public release: distribution is unlimited.

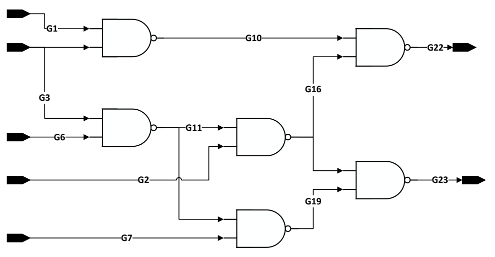
Logic Encryption (sometimes referred to as "Logic Locking" or "Logic Obfuscation") is a technique that inserts extra gates in the original netlist, aiming to hide the original functionality of the circuit [6]–[8]. These additional gates inserted are known as *key-gates*. Therefore, the overall circuit structure contains some primary inputs, pseudo primary inputs known as *key inputs* and primary outputs. In order to observe the original circuit functionality, one must provide the correct key values to the encrypted design. Circuit malfunctions if incorrect keys are applied to the encrypted circuit.

As an example, consider the c17 circuit from ISCAS85 benchmark encrypted as per [3] in Figure 1. Here, $G1$, $G2$, $G3$, $G6$, $G7$ are primary inputs and $G22$ and $G23$ are primary outputs. K1 and K2 are the *key inputs* for the corresponding XOR *key-gates*. Only upon making $K1 = 0$ and $K2 = 0$, the correct outputs can be observed. For other cases of K1 and K2 values the output gets corrupted. In this way, the designer has control on unlocking the IC.

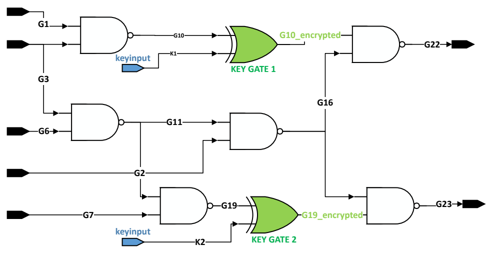
Logic Encryption aims to (a) thwart IC piracy - the attacker is unaware of the original functionality of the IC unless and until he/she knows the way to unlock it with the correct key scheme. (b) prevent insertion of malicious Hardware Trojans as the design is hard to analyze. Only after the IC is manufactured, a designer can unlock the circuit by applying the correct key values. Hence, the objective of an attacker is to figure out a way to deduce the correct (or equivalent) key values of the circuit.

Various logic encryption methods have been proposed over the years [7]–[10]. Each technique proposes a specific method of traversing the existing netlist and find possible ways of inserting key gates/programmable barriers. Though they aim to thwart various types of attacks proposed over the years, they have all been proven to be weak logic encryption methodologies after [11] proposed a logic decryption algorithm using Boolean satisfiability (SAT) solvers.

This paper focuses on a defensive technique against the SAT attack [11].The core contributions of this paper are the following:



(a) Original Circuit



(b) Encrypted with 2 keys K1 & K2

Fig. 1. Logic Encryption Illustration using the EPIC method [3]

1) We analyze the SAT attack against programmable logic. Based on the discriminating ability of individual input patterns in deciphering the programming sequence, we present a scenario that is hard for the SAT attack when programmable logic is used for obfuscation.
2) We propose a novel logic encryption method employing the logic duplication technique and using the intuition obtained about programmable logic against the SAT attack.
3) We demonstrate the effectiveness of the proposed encryption technique against the SAT attack.
4) We discuss the performance of this approach and its compatibility with COTS synthesis tools.

II. Prior Work

The SAT attack aims to reveal the correct key scheme of the encrypted circuit by carefully selecting input patterns to eliminate a set of wrong key candidates in every iteration. These input patterns are known as *Distinguishing Input Patterns (DIP)*. In other words, it is only the correct key for which the output is said to be consistently matching with the output of the unlocked IC. Therefore each iteration through the algorithm tries to determine one DIP for which incorrect keys are uniquely determined and eliminated. Since it uses the formal method of Boolean satisfiability, the final key determined by the attack is guaranteed to be correct.

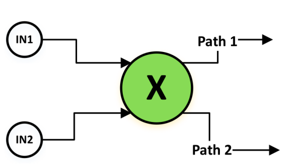
As the SAT Attack was unable to decrypt a few benchmarks, the authors of [11] found that circuits containing *AND-Tree* structures become hard instances for the SAT attack to decrypt. AND-Trees with N inputs being encrypted would require to evaluate 2^N equivalence classes to deduce the correct key.

The very first SAT attack countermeasure was proposed by [10] where insertion of the Advanced Encryption Standard (AES) based circuits to the encrypted netlist was contemplated. The netlist is encrypted by their proposed clique based insertion such that the key values do not propagate to the outputs of the circuit. But, the bottleneck of this method is the area overhead of the AES module is very high. The AND tree property which makes the SAT attacks exponential in time has become the key concept for the evolution of SAT Resistant Logic Encryption and Camouflaging techniques. This was exploited by the authors of [12] by introducing an Anti-SAT block design [12]. Rajendran et al. [13] proposed the SAT Attack Resistant Logic Locking (SARLock) based method that tries to invert the output for all incorrect keys and ensures proper behavior for the correct key. This way, SARLock becomes a single point function thereby ensuring that SAT Attack eliminate no more than one key in every iteration. The authors from [14] proposed a security metric for evaluating the SAT attack complexity called *De-camouflaging or Decryption Complexity* using the motivation from active learning concepts. They also came up with an algorithmic approach for detecting isolated AND tree structures within the existing netlist. Based on the gate type and the fan-out conditions their method finds the least non decomposable AND tree in an existing circuit. Shamsi et al. showed that the given circuit netlist can be obfuscated with combinational loops such that it is difficult for the SAT attack to be launched [15]. The authors of SARLock had come up with a better countermeasures to SAT attacks by proposing a new sub-circuit named Tenacious and Traceless Logic Locking (TTLock) [16].

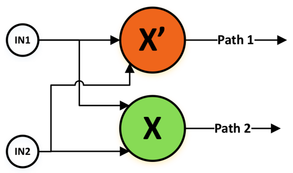
III. Analysis of SAT Attack Resilience of Programmable Logic

Look-up tables are extensively used as a programmable block in ASICs. Though the previous countermeasure techniques tried to add additional circuitry to the existing netlist, there has not been much exploration on finding sub-circuits within the netlist that may be making SAT attack exponential. Although, the authors from [14] proposed finding AND-Tree structures within the existing netlist, this tree search process has been purely based on the type of gates present in the netlist. However, not all circuits are guaranteed to contain such AND-Tree structures. We propose to find sub-circuits within the netlist which, upon hiding, could possibly make it SAT attack tolerant. Hiding the sub-circuit can be correlated to making this portion to be programmable. In order to make the sub-circuit programmable, Look-up Tables (LUTs) can be used. Though CMOS SRAM LUTs pose much area overhead, emerging devices could possibly improve the area. This will be discussed in a later section. Upon hiding a portion of the circuit with LUTs, the attacker cannot decipher the circuit unless and until they know the exact programming sequence.

When a sub-circuit is hidden using a traditional LUT, the configuration bits of the LUT serve as the key bits and the inputs of original sub-circuit serve as the select lines of the LUT. In general, if a k-input LUT is used to replace a k-input function, the SAT attack in the worst case requires 2^k iterations. 2^K becomes large enough only if K is large enough to make it difficult for SAT to decipher the functionality of



(a) Original Graph



(b) Logic Duplication of X circuit block

Fig. 2. Logic Duplication [18]

the LUT. But we cannot afford such huge LUTs as it leads large area overheads. Therefore, breaking them into m-input LUTs such that m is small enough to make it SAT-hard is a reasonable way of approaching this problem. Hence, we select a sub-circuit within the existing netlist such that the sub-circuit contains a given number of inputs and replace the gates in the sub-circuit with programmable logic. For our evaluation we use traditional LUTs.

IV. Logic Duplication for Sub-Circuit Selection

A. Motivation

Some sub-circuit gates may have fan-out nets to other gates which are outside the sub-circuit. These nets provide sneak paths to help the SAT attack to decipher the key. This can happen when the design has a large number of fan-out regions. Industrial circuits do have multiple fan-out regions which can cause the sub-circuit selection methodology to be ineffective. Hence, in order to combat these sneak paths, we use the concept of "logic duplication" which will be explained in detail in the next section.

B. Background on Logic Duplication

Logic Duplication [17] was proposed by Steven J. Perry in order to improve the circuit timing and size. Logic duplication is a process of duplicating selected logic gates in order to avoid sharing of hardware [18]. Also, according to [19] this technique does not increase area overhead all the time because the duplicated design would have lower routing overhead though the total gate count increases due to the duplication procedure.

To understand the concept of logic duplication, consider a portion of a circuit represented as a graph shown in Figure 2a. Here, X is a circuit block that has a fan-in from $IN1$ and $IN2$ nets. The output of X block has 2 paths, namely, $Path1$ and $Path2$. In other words, $Path1$ and $Path2$ are driven by sharing the same computing block namely X. Now, if the block X is duplicated to become X', $Path1$ and $Path2$ would have independent driving circuit blocks. This is the basic procedure used for logic duplication [18].

This technique is exploited in order to isolate the sub-circuit selected for LUT replacement from the portion of circuit that is getting exposed to the SAT attack. This way, sneak paths are eliminated from the sub-circuits and can improve the overall tolerance of the encryption from SAT attack.

C. Logic Duplication Method for Sub-Circuit Identification

Algorithm 1: Duplication for Cone of Influence at Highest Topology Net

```
Input: $C$ and $K_{constraint}$
Output: $C_{encypt}(X, K_{achieved}, Y)$
1  $N_{Ordered}$ ← TopologicalSort($C$)
2  $N_{HighestOrder}$ ← PickHighestOrder($N_{Ordered}$)
3  $K_{achieved}$ ← UpdateK(pow(2,FanIn($N_{HighestOrder}$)))
4  InfluenceNet ← $N_{HighestOrder}$
5  $i \leftarrow 0$
6  if $K_{achieved} \leq K_{constraint}$ then
7      SubC($X_i, Y_i$) ←
         AddToSubCircuit(Gate(InfluenceNet))
8      $i \leftarrow i + 1$
9      while $K_{achieved} \leq K_{constraint}$ do
10         ConeNet ← TraceConeOfInfluence(InfluenceNet)
11         $K_{achieved}$ ← UpdateK(pow(2,FanIn(ConeNet)))
12         if FanOut(ConeNet) $\geq 2$ then
13             ConeNetDuplicated ←
                 LogicDuplicationOf(ConeNet)
14             SubC($X_i, Y_i$) ←
                 AddToSubCircuit(Gate(ConeNetDuplicated))
15             $i \leftarrow i + 1$
16             InfluenceNet ← ConeNetDuplicated
17         else
18             SubC($X_i, Y_i$) ←
                 AddToSubCircuit(Gate(ConeNet))
19             $i \leftarrow i + 1$
20             InfluenceNet ← ConeNet
21         end
22     end
23 end
24 $C_{encypt}(X, K_{achieved}, Y)$ ←
     ReplaceSubCircuitGatesWithLUT($SubC(X_i, Y_i)$)
```

In the proposed method, gates are replaced by LUTs on a gate by gate basis. Hence, the size of the LUT would be based on the fan in of the gate being replaced. The user would be giving two inputs to the algorithm as follows:

1) Original Netlist of the circuit C with X primary inputs (PIs), G gates and Y primary outputs (POs). Let N be the total number of nets present in the circuit. This includes the primary inputs, primary outputs and the intermediate nets.
2) Total Configuration bits or key bits is a user's constraint denoted by $K_{constraint}$. Since every design requires unique performance targets, this is made to be a constraint so that the designer can trade off between the security and performance based on the requirements. These configuration bits of the LUTs can be viewed as the key bits of the encrypted circuit.

The sub-circuit selection algorithm has the following steps:

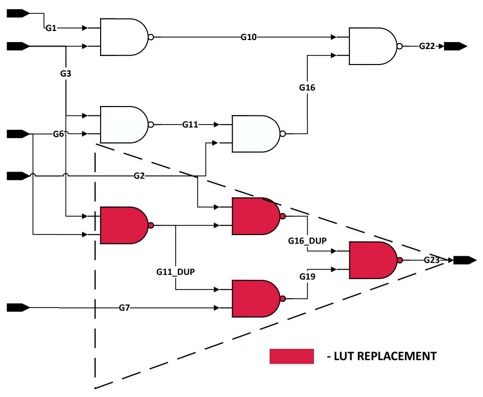


Fig. 3. Logic Duplication on COI illustration for c17 benchmark

1) The given circuit is topologically sorted using $TopologicalSort$ routine and the overall ordered nets correspond to $N_{Ordered}$.
2) The highest topological net is selected using $PickHighestOrder$ routine.
3) This net is made as the *Influence Net* and added to the sub-circuit. $K_{achieved}$ is updated based on the fan-in of the influence net using $UpdateK$ routine.
4) The Cone of Influence (COI) of this net is determined using the $TraceConeOfInfluence$ routine. During the cone traversal, if any net has fanout that goes outside the cone, logic duplication is applied and the duplicated version of the net is added to the sub-circuit. This is done using the $LogicDuplicationOf$ routine. This way, the sub-circuit gets isolated from the original circuit.
5) The cone traversal stops when the condition $K_{achieved} \leq K_{constraint}$ is satisfied. In case the cone traversal is completed and the condition still does not get satisfied, the algorithm terminates with the complete cone of the highest topology net made as the sub-circuit.
6) The selected sub-circuit is replaced with LUTs and the overall circuit becomes encrypted.

For example, consider the smallest benchmark c17 in ISCAS85 [20] which is illustrated in Figure 1a. Here nets $G22$ and $G23$ have the highest topological order. Say, the algorithm selects $G23$ for cone traversal. As the cone gets traversed for $G23$, there are fanout nets $G16$ and $G11$ that sneak to the other side of the c17's logic. Hence, the nets $G16$ and $G11$ get duplicated till the primary inputs $G2$, $G3$ and $G6$. The duplicated nets are denoted with the "$\_DUP$" footnote in the net name. Now, the cone of $G23$ gets isolated from c17's other logic region. The gates that drive $G23$ are now replaced with LUTs. This is indicated by the maroon color coding on the corresponding gates in Figure 3.

V. Evaluation

The sub-circuit selection algorithm was implemented in C++ and the SAT attack tool was obtained from the authors of [11]. All the experiments were carried out in the Linux Workstations operating at 3.3GHz and 16GB RAM. The SAT attack timeout was set to be 24 hours.

We have used the some of the ISCAS85 and MCNC sets of benchmarks [20], [21] similar to [12], [13], [16]. Apart from these, we also added several unrolled sequential circuits from ITC99 [22] into our benchmark list as this list contained larger circuits compared to the former two. The ITC99 benchmarks are reordered as per topology and unrolled hence we renamed the benchmarks with a footnote "_new".

A. Methodology

The LUT replacement was done using traditional MUX based LUT design and in order to make it compatible with the tool, a gate level representation of each MUX was required. The overall area overhead is reported as the % of gates being replaced with LUT.

B. Results and Discussion

Most of the benchmarks were tolerant with SAT attack with key bits of 256 for the proposed encryption. The overall percentage of gates being replaced with LUT decreases as the size of the benchmark increases. This trend can be observed in Figure 4.

Benchmarks c1908, c7552 and des are vulnerable to the attack with key being 256 bits. This is because the sub-circuit contains many 1-input gates utilizing the LUT configuration bits. Since 1-input LUT is just a MUX, it is easier for SAT attack to decrypt them in comparison to other gates which have multiple inputs. In order to check their tolerance towards SAT attack, the key constraint is increased to 650, 1024 and 2000. It is observed that the benchmark c1908 becomes SAT tolerant for the key size of around 2000. Similarly for benchmarks c7552 and des it requires a key size of 648 to tolerate SAT attack. These sub-circuits contained a large number of 1-input gates hence require more gates to be replaced, hence require more key bits to tolerate the attack.

The other insight in regards to large number of inverters in the sub-circuit is to combine these inverter functions with the previous gates. For instance, if there is a AND gate driving an inverter this logic can be replaced with a single NAND gate. This ensures single input LUTs are not a part of the sub-circuit. But the downside here is some sub-circuits may need both the inverted and non-inverted signals for the overall logic computation. Hence, combining inverted signals with its driving logic would lead to incorrect or missing logic within the overall encrypted circuit.

C. Performance Analysis

A concern on this approach would be the power and area overhead as SRAM LUTs are power hungry and consume significant amount of area. As we notice from the results, as the circuit size grows the overall percentage of sub-circuit that gets replaced becomes insignificant. However, SRAM LUTs consume significant power. But there are emerging technologies in which LUTs can be designed in order to reduce the power. This is discussed in the next section.

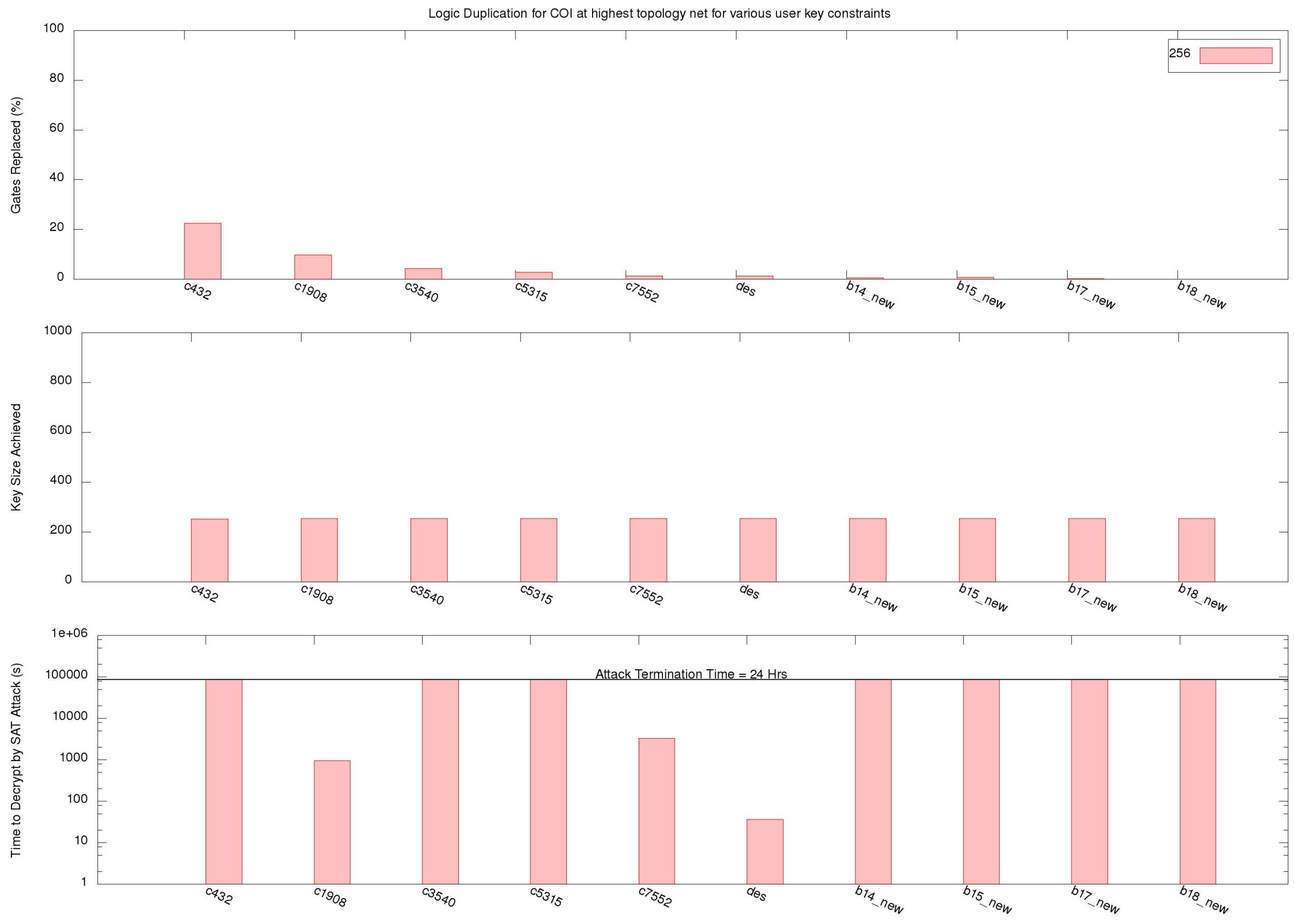


Fig. 4. Histogram illustrating SAT tolerance of various benchmarks for the proposed logic duplication based sub-circuit selection

D. Alternatives to LUTs

Traditional SRAM LUTs were used in this work in order to check if the functionality of a LUT would help in thwarting the SAT attack. For alternatives to these LUTs, camouflaging approaches can be used. Camouflaging tries to include as many functions as possible but does not cover all the functions as LUTs do.

There has been extensive research on LUTs based on new devices which are proven to be more area and power efficient. One such advanced LUT design was proposed by [23] where LUTs were designed using transfer-torque magnetic tunnel junction (STT-MTJ) devices. This device has low delay and power consumption in comparison to the traditional CMOS LUT design. Such emerging devices could possibly be a replacement to the traditional LUT in the above proposed encryption strategy thereby making it SAT tolerant with lower degradation in power and area metrics.

E. Incorporation with COTS Synthesis Tools

The proposed work selects sub-circuits from an existing netlist file. Therefore, this approach can be easily incorporated as a part of any COTS synthesis tool to encrypt subject to a user defined security metric. The user can specify whether he/she requires security for the design and also provide the maximum threshold of security he/she can accommodate. The maximum threshold of security thereby gets translated into the number of keys that the user wants for his/her design. The proposed tool flow is shown in Figure 5.

VI. Conclusion

Several countermeasures have been proposed to thwart the SAT Attack against logic encryption that aim to prevent IP piracy. We propose a novel method that identifies sub-circuits within the existing netlist that are good candidates for logic encryption. The idea of logic duplication is incorporated in order to make the sub-circuit independent of the other portions of the circuit that gets exposed to the attack thereby increasing resistance to the SAT attack.

References

[1] J. Rajendran, O. Sinanoglu, and R. Karri, “Regaining trust in vlsi design: Design-for-trust techniques,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, 2014.

[2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, “Hardware trojan attacks: threat analysis and countermeasures,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

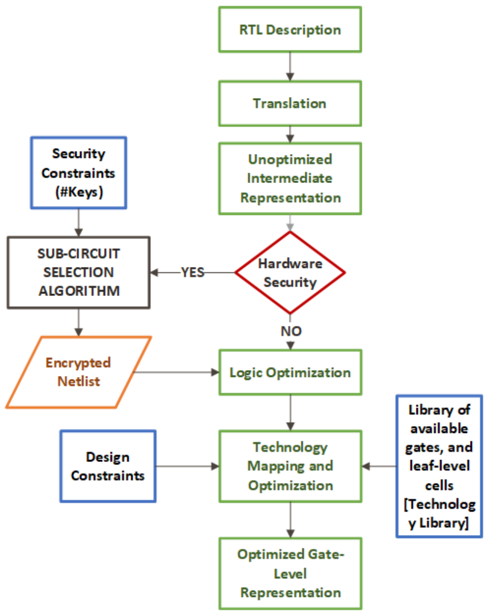

RTL Description
Translation
Unoptimized Intermediate Representation
Security Constraints (#Keys)
SUB-CIRCUIT SELECTION ALGORITHM
YES
Hardware Security
NO
Encrypted Netlist
Logic Optimization
Library of available gates, and leaf-level cells [Technology Library]
Design Constraints
Technology Mapping and Optimization
Optimized Gate-Level Representation

Fig. 5. Incorporation of Sub-Circuit LUT Encryption in Synthesis Flow

[3] J. A. Roy, F. Koushanfar, and I. L. Markov, “Ending piracy of integrated circuits,” *Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[4] R. Torrance and D. James, “The state-of-the-art in semiconductor reverse engineering,” in *Proceedings of the 48th Design Automation Conference*. ACM, 2011, pp. 333–338.

[5] Q. Yu, J. Dofe, Y. Zhang, and J. Frey, *Hardware Hardening Approaches Using Camouflaging, Encryption, and Obfuscation*, ser. Hardware IP Security and Trust. Springer, 2017, pp. 135–163.

[6] J. J. Rajendran and S. Garg, *Logic Encryption*, ser. Hardware Protection through Obfuscation. Springer, 2017, pp. 71–88.

[7] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, “Security analysis of logic obfuscation,” in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[8] 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.

[9] A. Baumgarten, A. Tyagi, and J. Zambreno, “Preventing ic piracy using reconfigurable logic barriers,” *IEEE Design & Test of Computers*, vol. 27, no. 1, 2010.

[10] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, “On improving the security of logic locking,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2016.

[11] P. Subramanyan, S. Ray, and S. Malik, “Evaluating the security of logic encryption algorithms,” in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 137–143.

[12] Y. Xie and A. Srivastava, “Mitigating sat attack on logic locking,” in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 127–146.

[13] M. Yasin, B. Mazumdar, J. J. Rajendran, and O. Sinanoglu, “Sarlock: Sat attack resistant logic locking,” in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 236–241.

[14] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, “Provably secure camouflaging strategy for ic protection,” in *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 2016, p. 28.

[15] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, “Cyclic obfuscation for creating sat-unresolvable circuits,” in *Proceedings of the on Great Lakes Symposium on VLSI 2017*. ACM, 2017, pp. 173–178.

[16] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, “What to lock?: Functional and parametric locking,” in *Proceedings of the on Great Lakes Symposium on VLSI 2017*. ACM, 2017, pp. 351–356.

[17] S. Perry, “Logic duplication method for reducing circuit size and delay time,” Jun. 22 1994, eP Patent App. EP19,910,310,405. [Online]. Available: https://google.com/patents/EP0486248A3?cl=no

[18] A. Balakrishnan and S. T. Chakradhar, “Retiming with logic duplication transformation: theory and an application to partial scan,” in *VLSI Design, 1996. Proceedings., Ninth International Conference on*. IEEE, 1996, pp. 296–302.

[19] S. Devadas, H.-K. Ma, A. R. Newton, and A. Sangiovanni-Vincentelli, “Synthesis and optimization procedures for fully and easily testable sequential machines,” in *Test Conference, 1988. Proceedings. New Frontiers in Testing, International*. IEEE, 1988, pp. 621–630.

[20] F. Brglez and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran,” in *Proceedings of IEEE Int'l Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[21] S. Yang, “Logic synthesis and optimization benchmarks user guide: Version 3.0,” MCNC Technical Report, Tech. Rep., Jan. 1991.

[22] F. Corno, M. Reorda, and G. Squillero, “Rt-level itc'99 benchmarks and first atpg results,” *Design Test of Computers, IEEE*, vol. 17, no. 3, pp. 44–53, Jul 2000.

[23] D. Suzuki, M. Natsui, and T. Hanyu, “Area-efficient lut circuit design based on asymmetry of mtj's current switching for a nonvolatile fpga,” in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*. IEEE, 2012, pp. 334–337.