Silicon Validation of LUT-based Logic-Locked IP Cores

Gaurav Kolhe, Tyler Sheaves, Kevin Immanuel Gubbi, Tejas Kadale[§], Setareh Rafatirad, Sai Manoj PD[*], Avesta Sasan, Hamid Mahmoodi[§], Houman Homayoun
Dept. of Electrical and Computer Engineering, University of California, Davis, CA, USA
[*]Dept. of Electrical and Computer Engineering, George Mason University, Fairfax, VA, USA
[§]School of Engineering, San Francisco State University, San Francisco, CA, USA
{gskolhe,tsheaves,kgubbi,ssalehi,srafatirad,asasan,hhomayoun}@ucdavis.edu;
spudukot@gmu.edu;{tkadale,mahmoodi}@sfsu.edu

ABSTRACT

Modern semiconductor manufacturing often leverages a fabless model in which design and fabrication are partitioned. This has led to a large body of work attempting to secure designs sent to an untrusted third party through obfuscation methods. On the other hand, efficient de-obfuscation attacks have been proposed, such as Boolean Satisfiability attacks (SAT attacks). However, there is a lack of frameworks to validate the security and functionality of obfuscated designs. Additionally, unconventional obfuscated design flows, which vary from one obfuscation to another, have been key impending factors in realizing logic locking as a mainstream approach for securing designs. In this work, we address these two issues for Lookup Table-based obfuscation. We study both Volatile and Non-volatile versions of LUT-based obfuscation and develop a framework to validate SAT runtime using machine learning. We can achieve unparallel SAT-resiliency using LUT-based obfuscation while incurring 7% area and less than 1% power overheads. Following this, we discuss and implement a validation flow for obfuscated designs. We then fabricate a chip consisting of several benchmark designs and a RISC-V CPU in TSMC 65nm for post functionality validation. We show that the design flow and SAT-runtime validation can easily integrate LUT-based obfuscation into existing CAD tools while adding minimal verification overhead. Finally, we justify SAT-resilient LUT-based obfuscation as a promising candidate for securing designs.

CCS CONCEPTS

• **Security and privacy → Hardware attacks and countermeasures**; • **Hardware → Hardware validation**.

KEYWORDS

Reverse Engineering, Logic Locking, SAT Attack, Emerging Devices, Dynamic Obfuscation, Power side channel attack

ACM Reference Format:
Gaurav Kolhe, Tyler Sheaves, Kevin Immanuel Gubbi, Tejas Kadale[§], Setareh Rafatirad, Sai Manoj PD[*], Avesta Sasan, Hamid Mahmoodi[§], Houman Homayoun. 2022. Silicon Validation of LUT-based Logic-Locked IP Cores. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC) (DAC '22), July 10–14, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530606

1 INTRODUCTION

Recent logic obfuscation schemes for digital circuits have focused on making the design resilient to various security threats. The most sophisticated and influential of these threats is the **Boolean satisfiability** (SAT) attack. However, the current practice for validating resiliency against SAT attacks consists of simulating the attack indefinitely until the attack can de-obfuscate the design [9]. The time for attack completion can be several days, weeks, or even months. Given the uncertainty of de-obfuscation time using SAT attacks, the simulation method for assessing and quantifying design security is not practical as it can introduce an indefinite delay in the IP design flow. To counter SAT attack, many obfuscation schemes have been proposed. These schemes are often disruptive to the design flow and involve modifications to design RTL, gate-level netlist, timing characterization, and even layout. Ideally, IC designers would perform obfuscation in a modular fashion as an additional stage in the ASIC flow that requires minimum alteration to the other stages. The lack of methods to validate SAT-resiliency along with unconventional physical design and validation flows discourages most IP designers from utilizing obfuscation primitives for securing their design.

Given the dire need for quantifying and validating the security against SAT-attack, a couple of challenges must be addressed: First, many obfuscation techniques have been proposed with sophisticated theories, rules, and heuristics, to name a few. The effect of such obfuscations and netlist are highly-nonlinear for conventional simple models (e.g., linear regression and support vector machine [1]) to characterize. Second, The input for a design security analysis tool could be of varying circuit size, and therefore, varying gate-level netlist obfuscation criteria. Therefore, we must perform the feature extraction methods that support varying-structured data without significant information loss and enables the model to learn the effect of obfuscation.

To validate the security of the obfuscated IP and address the open challenges of SAT runtime, this work proposes Design Security Analyzer. The Design Security Analyzer is based on the **Conjunctive Normal Form** (CNF) graph representation of the obfuscated circuit, where we train a graph neural network model on the obfuscated circuit to capture the trend and effect of the obfuscation on the design to be secured. This resulting model allows the IP owner to sweep various obfuscation metrics (such as obfuscation coverage, different obfuscation methods, gate selection) and helps validate the obfuscation's security quickly.

Further, this work addresses the need for a simple, effective, and modular obfuscation technique to thwart the SAT attack surface. This technique inserts the proposed obfuscation primitives [6] after the completion of the synthesis stage. The proposed strategy only requires predictable and straightforward modifications to the design test methodology. Therefore, the obfuscated design can be moved through the remaining stages of the physical design flow without any additional accommodations while allowing the ability to validate the functionality and security of the design. Most obfuscation works have presented their results from experimental results. However, this is the first work that validates the claims of **Lookup Table** (LUT)-based obfuscation by fabricating the test IC. While validating the claims of the LUT-based obfuscation [6], we test both volatile and non-volatile versions of LUTs while sweeping

This work is licensed under a Creative Commons Attribution International 4.0 License.
DAC '22, July 10–14, 2022, San Francisco, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9142-9/22/07.
https://doi.org/10.1145/3489517.3530606

the obfuscation key sizes. To sum up, this study provides a comprehensive study on the validation of security and functionality for obfuscated IP for various configurations of LUT-based obfuscation.

2 BACKGROUND

2.1 Logic Obfuscation & SAT Attack

In logic obfuscation, the insertion of additional logic helps to conceal the functionality of a target design. The addition of extra logic serves to introduce ambiguity in the design to avoid netlist extraction after de-layering, or other reverse engineering techniques [12]. The SAT attack [12] is used to find the correct key of the obfuscated or logic-locked circuit without brute-forcing through all key combinations. The SAT solver input is a Boolean formula in CNF obtained from the transformation of the obfuscated netlist. [12].

As aforementioned, most research work claims the resiliency of their proposed obfuscation against the SAT attacks and the variants of SAT-attack by the method of simulation. However, features such as circuit's topology, obfuscation method, and key size play a pivotal role in determining the SAT resiliency, and their effect cannot be justified through limited SAT simulations.

For the estimation of SAT-solver runtime, we investigated the SatZilla [10] framework, which is a portfolio-based SAT selection algorithm that chooses the best SAT-solver given a problem. The selection algorithm predicts the performance of different solvers, and this idea of predicting performance can be expanded to predict the runtime. However, in hardware security, the obfuscation strategies have been tailored based on the experiences and rules hand-crafted by domain experts. Therefore, the features extracted are heuristics in nature and extracted on a case-by-case obfuscation, making it hard to generalize the model. In this work, we use features used in SatZilla along with the determinant features extracted and learned by the **Graph Convolutional Network** (GCN) from the raw CNF input to predict the runtime. The following section introduces the Graph Neural networks and how we deploy and adapt them for SAT runtime prediction.

2.2 Graph Neural Networks

Recently, there has been an increasing interest in applying deep learning for various graph data [11], such as social networks, molecular structure, road networks, and brain connectivity. The spectral convolution methods [5, 8] are the mainstream algorithms developed as the graph convolution methods. The graph convolution methods are based on the graph Fourier analysis [4]. [3] proposed the polynomial approximation. Inspired by this, **Graph Convolutional Neural Networks** (GCNNs) [5] were able to leverage the idea of **Convolutional Neural Networks** (CNN) in dealing with the Euclidean data for modeling graph-structured data. Kipf and Welling proposed GCNs [8], which naturally integrates the connectivity patterns and feature attributes of graph-structured data and outperforms many state-of-the-art methods such as GCNN. By utilizing graph topology, attributes of the surrounding nodes, the graph convolutional networks can learn the features related to security validation of obfuscated IP against SAT-attack in the hardware security domain.

While validating the security solves one of the obfuscation issues, we need to validate the functionality to prove that they do not induce unintended timing or functional failures while making minimal modifications to the conventional design flow.

2.3 Validating Obfuscation in Silicon

A holistic verification approach is required to validate the functionality of the design when the design is received from the foundry. Ideally, the strategy would be similar to the convention Silicon Validation, but additionally, we must configure the obfuscation primitives and run the initial design tests.

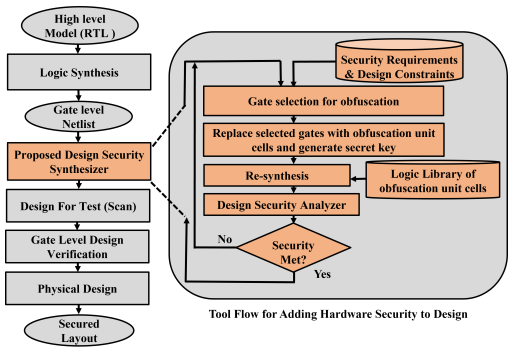


Figure 1: Security Oriented Design Flow for LUT-based Obfuscation

3 VALIDATION METHODOLOGY AND SECURE DESIGN FLOW FOR SAT-RESILIENT IPS

Figure 1 shows the Security Oriented Design Flow. Due to the addition of obfuscation primitives in the design, the physical design flow differs from the traditional Physical Design. This work studies the LUT-based obfuscation, as it requires minimalist changes to the design flow while being resilient to SAT-attack. While obfuscating with LUT, the Logic Synthesis stage now comes with the iterative security-driven flow, using the Design Security Analyzer. The gate selection, replacement, and security validation process is repeated until security constraints are met. Additionally, to avoid critical paths, the gate selection process can be adjusted in this stage.

3.1 LUT Unit Obfuscation Cell

To replace arbitrary logic gates, we implement the LUT with input obfuscation unit cell shown in Figure 2a. This cell includes a "logical" LUT configured to perform the same logical function of replacing the cell. Additionally, preceding "input" LUTs are configured to forward the proper inputs to the logical LUT. This cell allows the functionality of any n-input logic gate to be mapped to any m-input LUT through the programming of 2^n logical configuration bits (CBs) and n by $2\frac{m}{n}$ input configuration bits [6]. The proper inputs to the LUT are selected from the pool of valid input options via loading a configuration that allows the small input LUTs to buffer the selected input to the input of the larger logical LUT. The configured cell allows for a combination of interconnect and logical obfuscation resulting in a substantially longer SAT run time [6].

Another critical consideration of the unit obfuscation cell is CB management. CBs must not be accessible to the attacker and cannot be exposed to a system bus when the device is initialized. One approach to protect the CBs is to store them in a protected manner and load them via the scan chain externally in case of volatile LUT. Alternatively, the CBs themselves can take a **non-volatile** (NV) form. For example, CBs can be replaced with the non-volatile emerging devices that are becoming available on the market and offers a promising low overhead, secure and re-programmable solution [7].

After adding the LUT-based obfuscation primitive, we must verify the SAT resiliency. In the following discussion, we elaborate on Design Security Analyzer, which facilitates instantaneous design security validation.

3.2 Netlist Modelling

We propose to use the Machine Learning algorithm to predict the non-linear SAT runtime of an obfuscated IC. Before leveraging the ML to predict the SAT runtime, we must overcome the difficulty in representing the obfuscated netlist in an intact and structured way for an ML model representation. CNF used in SAT modeling, though typically written as a sequence, is mathematically not a sequence

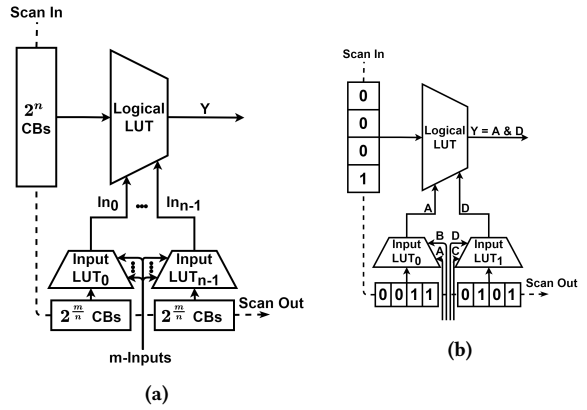


Figure 2: Shown in (a) is The generic LUT unit cell used in this work, while (b) shows an LUT unit cell configured as a 2-input AND gate with 4 input options.

as the order among different clauses is meaningless. Moreover, one literal can appear in multiple clauses with or without their negation forms, further complicating its representation. Moreover, unlike conventional inputs of machine learning models, CNF inherently endorses logical operators (typically discrete) instead of numerical operators. Thus it is challenging to automatically learn the determinant features that decide how "time-consuming" deobfuscating a CNF is.

3.3 Design Security Analyzer Overview

To address the challenges in validating the resiliency of obfuscated IP against SAT-attack, in this section, we elaborate our Design Security Analyzer shown in Figure 3.

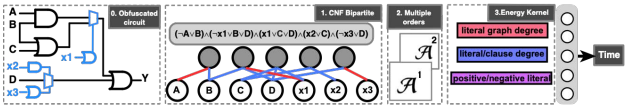


Figure 3: Architecture of SAT-runtime estimator

First, the converted CNF of an obfuscated circuit is modeled as an undirected and signed bipartite graph that uses one node type for clauses and the other for literals. This *CNF bipartite graph* is exemplified in Figure 4 and defined as $\mathcal{G}(E, V^{literal}, V^{clause})$, where $V^{literal}, V^{clause}$ indicate the set of literal and clause nodes, respectively. The sign of an edge between a literal l and a clause c denotes whether l is negated or not in c. That means, the edge value is: (1) (positively connected), if l is in c, and l is positive; (2) -1 (negatively connected), if l is in c, and l is negative i.e., $\bar{l}$; (3) 0 (disconnected), if l is not in c.

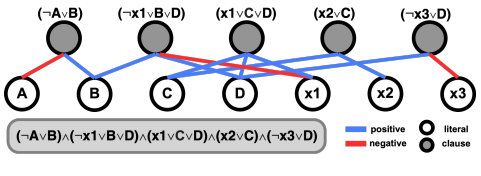


Figure 4: Example showing conversion from CNF to 1^{st} order graph.

The proposed CNF bipartite graph provides a comprehensive representation of the CNF without information loss. Moreover, we find that the CNF bipartite graph is a powerful representation such that its multi-order version can also capture additional meanings of a CNF. Representation in the CNF bipartite graph allows us to study the effect of previous stages'/logical gate(s) on a given gate. Multi-order expands the number of stages to be considered to represent the circuit effectively.

3.4 Energy-based Operators for CNF Graph

The energy model is used to encode the CNF bipartite graph such that the representation of varying-size CNF bipartite graphs can have a unified dimension for the machine learning technique. This task cannot be effectively handled by existing graph classification or regression models because of the unique properties of both the input and output. Unlike the conventional graphs, the correlation among the neighboring logical nodes in the CNF bipartite graph does not indicate "proximity" or "similarity". Instead, it indicates the logical relation with signed edges in a variable-size bipartite graph. A novel graph encoder layer has been proposed by leveraging and extending the energy of the **Restricted Boltzmann Machine** (RBM) to address this unique issue.

By innovatively treating the literals and clauses as visible and hidden units, the CNF bipartite graph can be modeled by RBM. The energy of the original RBM is defined as:

$$E(\boldsymbol{v}, h) = -\underbrace{\boldsymbol{a}^\top \boldsymbol{v}}_{\text{visible}} - \underbrace{\boldsymbol{b}^\top \boldsymbol{h}}_{\text{hidden}} - \underbrace{\boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}}_{\text{interaction}}, \quad (1)$$

Where $\boldsymbol{v}$ and $\boldsymbol{h}$ are the values of visible and hidden nodes, respectively, and $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{W}$ are weights to learn. The first term in equation (1) is the energy of visible nodes, the second term is the energy of hidden nodes, and the last terms is the interaction energy between visible and hidden nodes. Inspired by the two group modeling, $\boldsymbol{v}$ and $\boldsymbol{h}$ are the representations of a literal and a clause in the CNF bipartite graph, respectively. Similarly, an energy form is defined for characterizing a CNF:
$E = -\alpha \cdot E_{literal} - \beta \cdot E_{clause} - \gamma \cdot E_{interaction}$, where $E_{literal}, E_{clause}$ and $E_{interaction}$ are the energies of literals, clauses, and their connections, while α, β, γ are the weights of them, respectively. Since SAT runtime estimation over CNF is a highly nonlinear process, the traditional linear function has been generalized into a new nonlinear version:

$$E = f_\Phi(E_{literal}, E_{clause}, E_{interaction}), \quad (2)$$

where f_Φ is a neural network function controlled by parameter Φ. In the following, we study bipartite connection energy $E_{interaction}$ first and then $E_{literal}, E_{clause}$ in turn. Based on RBM, $E_{interaction}$ is defined as linear function of literals:

$$E_{interaction} = \sum_m \sum_n v_m w_{m,n} h_n, \quad (3)$$

where v_m is a literal and h_n is a clause in one single CNF bipartite graph $\mathcal{G}_i$. However, $E_{interaction}$ is not necessarily a sum function. Therefore, we generalize $E_{interaction}$ by generalizing convolutional graph layers into bipartite graphs. However, most existing graph deep learning operators focus on graphs with fixed topology. However, in our case. the size and topologies of the CNF bipartite graph vary across different instances dramatically. To solve this problem, we design a kernel for aggregating interaction information in one graph. Specifically, a d-dimensional vector of pseudo-coordinates is associated with $[\boldsymbol{v}, \boldsymbol{h}]$. We also define a weighting kernel $Z_\Theta(\cdot, \cdot)$, so that for one CNF bipartite graph $\mathcal{G}_i$, we have:

$$E_{interaction} = \sum_m \sum_n Z_\Theta(\mathcal{E}(\boldsymbol{v}_m, \boldsymbol{h}_n)) \cdot \mathcal{E}(\boldsymbol{v}_m, \boldsymbol{h}_n), \quad (4)$$

where $Z_\Theta(\cdot)$ projects the $[\boldsymbol{v}, \boldsymbol{h}]$ into a new value as the weight of $[\boldsymbol{v}, \boldsymbol{h}]$, and $\mathcal{E}(\boldsymbol{v}_m, \boldsymbol{h}_n)$ represents the interaction or edge value between $\boldsymbol{v}_m$ and $\boldsymbol{h}_n$ which can be 1, -1 or 0. Similarly, we further generalize $E_{literal}, E_{clause}$ as:

$$E_{literal} = \sum_m Ent(\boldsymbol{v}_m) \cdot \boldsymbol{v}_m, \text{ and } E_{clause} = \sum_n Ent(\boldsymbol{h}_n) \cdot \boldsymbol{h}_n, \quad (5)$$

where $\boldsymbol{v}$ and $\boldsymbol{h}$ indicate attributes of literal and clause respectively. While *Ent* denotes entropy function. Therefore, the final model for CNF is:

$$E = f_{\Phi}(\sum_{m} Ent(\boldsymbol{v}_m) \cdot \boldsymbol{v}_m, \sum_{n} Ent(\boldsymbol{h}_n) \cdot \boldsymbol{h}_n, \sum_{m}\sum_{n} Ent(\mathcal{E}(\boldsymbol{v}_m, \boldsymbol{h}_n)) \cdot \mathcal{E}(\boldsymbol{v}_m, \boldsymbol{h}_n)), \tag{6}$$

Equation (6) above does not consider the sign of the edges between literals and clauses. Hence, positive and negative information is encoded separately. Further, as an additional feature for learning, we count positive and negative edges for each clause and normalize both positive and negative counts.

3.5 Evaluation of Design Security Analyzer

The input to the framework is the obfuscated netlist and label (de-obfuscation time using SAT-attack). In this work, we considered the SAT attack proposed in [12] for security validation. However, any other derivatives of SAT attack(s) for de-obfuscation runtime evaluation could be used. We collect the data for initial training of the model by collecting obfuscated netlist and time taken by the SAT-attack to reverse engineer them. For generating obfuscated netlist, we used various benchmarks shown in Table 2, and we obfuscated them with LUT-based obfuscation [6] with varying sizes of LUT and obfuscation coverage. The total dataset consists of roughly 21000 obfuscated instances. Before this data is fed to the model, it requires pre-processing, which involves transformation to CNF and creating the adjacency matrix. The data is sampled and divided into 80% training and 20% testing sets.

From the data, we represent the obfuscated circuit as an adjacency matrix. The intermediate output of this operation is the CNF bipartite graph. Following this, multiple order information is extracted from the CNF bipartite graph. The first two orders of the CNF bipartite graph are considered in this model, and it is easy to extend to any order. After extracting the raw features of the *CNF bipartite graph*, an energy-based [2] kernel is used to model the dynamic-size data. This new kernel calculates the energy, which identifies the complexity of the corresponding CNF bipartite graph. To handle the dynamic size of CNF, the energy concept from RBMs is employed to aggregate literal and clause distribution into the fixed dimension. The aggregation of energy is treated as a feature of the targeted obfuscated IC. For modeling the runtime variance for different instances, a distribution kernel is applied in the last layer. In our model, we use an exponential distribution and finally use Adam as an optimizer. The final model generated using this process can be deployed to predict the runtime for the newly generated netlist.

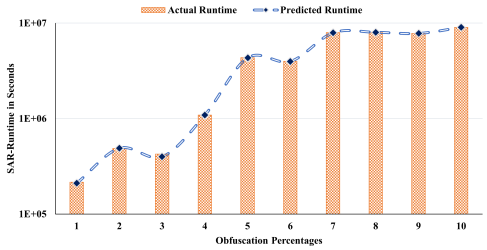


Figure 5: Prediction performance of Design Security Analyzer

To demonstrate the effectiveness of the Design Security Analyzer for security validation, we use the Pearson and Spearman coefficients. Positive scores of 1 indicate the capacity of the network in predicting the trend of runtime. While another metric, i.e., **Mean Squared Error** (MSE), shows the prediction error. For calculating the MSE scores, we take the log of the runtime. It is evident that the MSE doesn't go beyond ~3, meaning that the reported runtime has at max the delta of +/- 1000 seconds (15 minutes) while predicting for larger benchmarks. From table 1, it is evident that for all the benchmarks, MSE is low, and Pearson and Spearman have a positive correlation. Figure 5 shows the visual representation of runtime estimation. In this manner, Design Security Analyzer can be used to validate the security quickly. We used RISC-V as the benchmark for runtime prediction in Figure 5. For all the experiments, LUT of size 8 was used for obfuscation along with obfuscation methodology used in [6]. The X-axis denotes obfuscation percentage, and Y-axis denotes the de-obfuscation time.

Table 1: Performance of Design Security Analyzer

Benchmarks	MSE	Pearson	Spearman	Benchmarks	MSE	Pearson	Spearman
B01	1.241	0.95	0.97	DES Area	2.81	0.93	0.91
B02	1.26	0.91	0.89	DES Perf	2.38	0.92	0.92
B04	1.30	0.93	0.90	SHA-3	2.54	0.94	0.91
B12	1.8	0.95	0.93	8-bit CPU	3.1	0.91	0.90
AES	2.76	0.94	0.88	32-bit RISC-V	3.30	0.90	0.89

3.6 Configuring the LUT Unit Obfuscation Cell

The LUT unit obfuscation cell configuration depends on the logical function the cell is replacing and the input selection. Figure 2b shows a LUT unit obfuscation cell configured to perform a 2 to 1 AND function. In this example, both logical and input LUTs have identical topologies. Programming the configuration to the cell unlocks the desired cell function. To program the cell, two methods may be used. If CBs are to be loaded in externally, a dedicated scan chain is used. Alternatively, CBs may be driven internally from a non-volatile macro cell to bolster security. In this configuration, CBs may be directly driven to their respective LUTs, and programming is performed in the manner required by the non-volatile macro cell. A Python script converts the netlist to a graph of module objects, determines uncorrelated dummy input options, then randomly selects a number of them depending on the obfuscation needs. Then the script replaces the target cells with LUT unit obfuscation cells and connects them to the input list. Synopsys VCS logic simulator is used to generate logical CBs, and the Python script combines them with Python-generated input obfuscation CBs to create the configuration bit-stream.

3.7 Validating Obfuscated Designs

A target design is considered in a "locked" state so long as the proper configuration has not been applied. After configuration, the target design will return to its specified functionality. To validate that an obfuscated design is functional, a short programming task is completed, followed by the original design testbench. This task may be appended to the initial block of a **Hardware Verification Language** (HVL) testbench for pre-silicon validation. In the case of a scan chain, this task drives the input signals to the CB scan chain, as shown in Figure 6. If a non-volatile cell macro is used, the task drives the configuration logic of the macro cell, and upon configuration, the CB contents are driven directly to the CB input of the respective LUTs. To select the appropriate benchmark, *address_0* and *address_1* is set to 00, 01, 10 or 11 for selecting original,low, medium, and high obfuscated versions of the test chip.

4 DESIGN AND IMPLEMENTATION OF TEST CHIP

After validating both the security and the functionality of the obfuscated design, we must perform the post-fabrication functional verification. Through this experiment, we want to demonstrate the process of validating the functionality of the obfuscated design while verifying the claims of LUT-based obfuscation [6] on the data

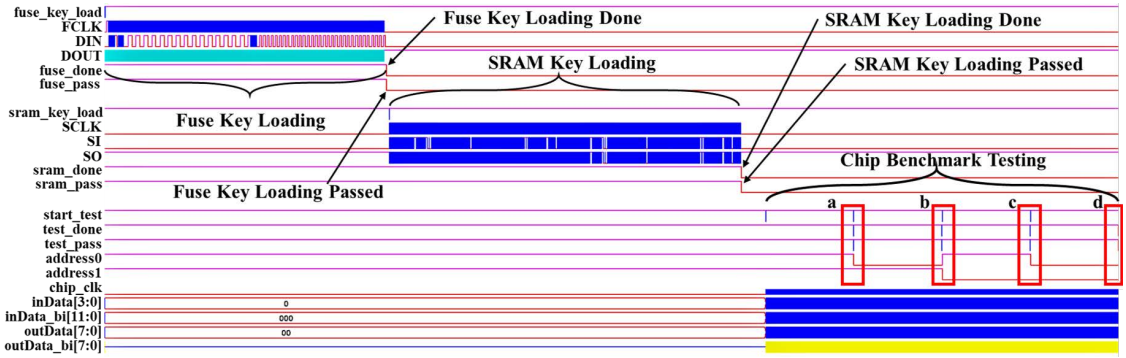


Figure 6: Synopsys VCS pre-silicon simulation for verification of obfuscated IP. The simulations include validation of a) original design, b) low, c) medium, and d) high obfuscated versions.

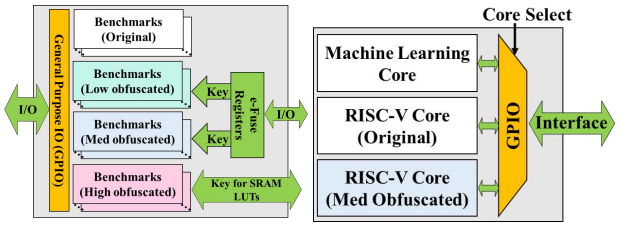
obtained post-fabrication rather than relying on the simulated results. The functionality test conducted here ensures that the design performs functional when the correct key is applied and enters obfuscated state when the wrong key is used.

To validate the LUT-based logic locking method, two test chips, as shown in Figure 8, were implemented and fabricated in TSMC 65nm technology. Chip1 contains three popular encryption engines (AES, DES, and SHA-3), a custom ALU, and other benchmarks. Chip2 contains three designs: a logic-locked 32-bit RISC-V microprocessor core, and its original counterpart. All benchmarks incorporated in both chips are shown in Table 2. We evaluated LUT-based obfuscation by implementing the LUTs in volatile and non-volatile form while sweeping the length of obfuscation key to provide readers with qualitative and quantitative results. Figure 7a shows the architecture of chip1, which contains 10 different benchmarks, each of which has four versions - original, low obfuscated, medium obfuscated, and high obfuscated. The key length in the low, medium, and high obfuscated designs is 288, 576, and 3168 bits, respectively. In low obfuscation we add single LUT_8 + 8×LUT_2, while medium obfuscation contains 2, and high obfuscation contains 11 LUT_8 + 8×LUT_2. LUT_8 + 8×LUT_2 refers to the Large LUT of size 8 whose 8 inputs are driven by the small LUT of size 2 as described in [6]. We evaluate these three levels to explore the solution's scalability and demonstrate the performance impact of obfuscation.

A **General-Purpose IO** (GPIO) block selects one of the designs connected to the top-level IO pins for testing. To demonstrate different key storage methods, the key bits for low and medium obfuscated designs are stored in non-volatile e-fuse registers, whereas those of the high obfuscated designs are stored in volatile SRAM registers. E-fuse was chosen as it was the only embedded non-volatile memory available in the target fabrication technology; however, non-volatile LUTs can be made of any embedded non-volatile technology and preferable with enhanced security against reverse engineering. In this study, we also studied SRAM storage as it is a low-cost option that is readily available in standard CMOS, but it requires an external and secure non-volatile key storage.

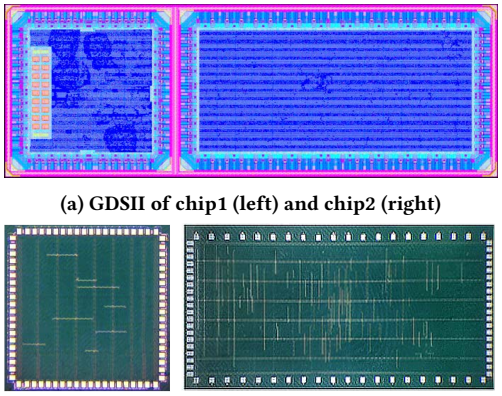
Table 2: Benchmarks included in test chip

Source	Benchmark	Description	Total # of cells
OpenCore	DES_area	DES optimized for area	2,085
	DES_perf	DES optimized for performance	15,851
	AES	AES cipher	10,787
	SHA-3	SHA-3 Encryption core (Keccak 512)	13,702
ITC'99	B01	FSM that compares serial flows	34
	B02	FSM that recognizes BCD numbers	26
	B04	Compute min and max	310
	B12	1-player game (guess a sequence)	2656
Custom	ALU	Multiplier/Adder/AND	136
OpenCores	CPU	8-bit microprocessor	1620
PicoRV32	CPU	32-bit RISC-V microprocessor	6892



(a) Chip1 architecture **(b) Chip2 architecture**

Figure 7: Overall Architecture for chip1 (a) and chip2 in (b)



(a) GDSII of chip1 (left) and chip2 (right)

(b) Die photo of chip1 (left) and chip2 (right)

Figure 8: (a) GDSII and (b) Die of chip1 and chip2

5 POST-SILICON VALIDATION OF LUT-BASED LOGIC-LOCKED CORES

5.1 Post-silicon Test Fixture

After fabrication, we perform post-silicon functional validation on each benchmark using an Intel Aria 10 development kit as a test instrument. We also develop a **Printed Circuit Board** (PCB) with a socket for the chip to allow simple interfacing between the **Field Programmable Gate Array** (FPGA) and the test IC. The test structure is configured in the manner shown in Figure 10. This configuration is deployed to hardware as shown in Figure 11.

For testing, the test vector input/response patterns are cached inside the FPGA and are selected via an address line from the controller. After a test vector is applied, the response is checked with the expected result. Test vectors are derived from the original test benches used to validate the individual benchmarks. Both SRAM and e-fuse-based LUTs are validated, and the controller module on the FPGA performs all configuration programming before applying test patterns in the same manner as pre-silicon validation. Through

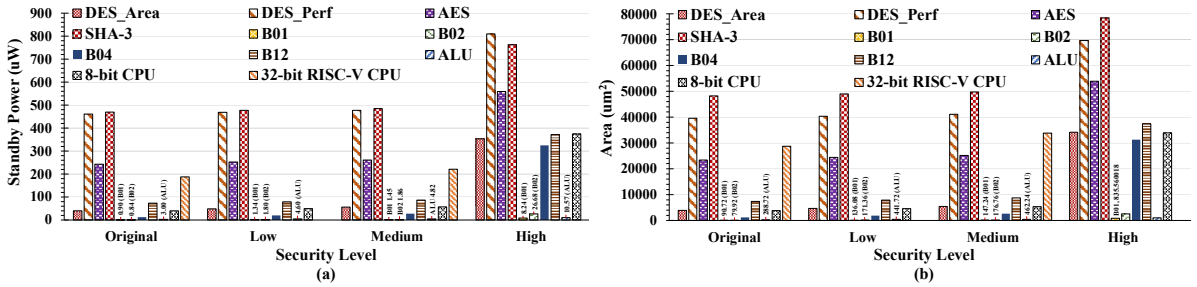


Figure 9: Study of (a) Standby power and (b) area for various obfuscation levels and benchmark for LUT-based obfuscation.

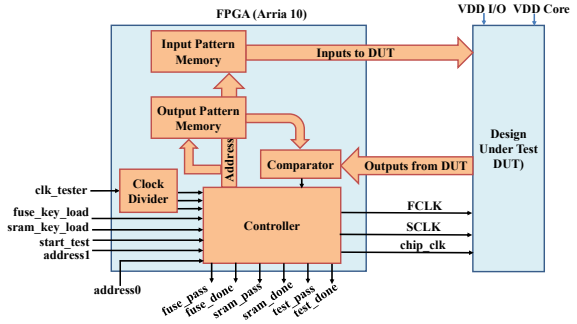


Figure 10: Diagram of FPGA-based Silicon validation setup for validating the functionality of obfuscated IP post fabrication.

this small-scale experiment, we show how post-silicon validation can be achieved.

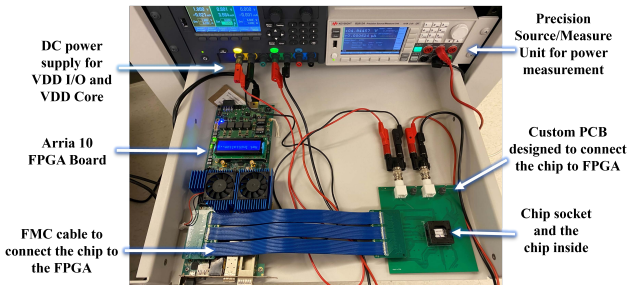


Figure 11: Test setup using the Intel Aria 10 development kit.

5.2 Post-silicon validation results

All benchmarks are tested using test vectors generated from the original test bench of each design. To select the benchmark variant, control signals are added. Keys are loaded upon pressing a push-button switch. The LEDs have been assigned on the FPGA test setup to identify failure or the success of the testing process. To show that the test bench properly stimulates the obfuscated cells, incorrect keys are also programmed for each benchmark, and failure is observed for improperly programmed design.

Following the functionality validation results, we also validated the claim of LUT-based obfuscation [6]. Figure 9 shows the standby power and area footprint of the fabricated designs. All 3 variants, i.e. (Low, Medium, and high), result in SAT-timeout. These SAT-resilient benchmarks have an average of 7% area overhead for low obfuscated configuration while 14% and 262% for medium and high obfuscation. Standby power scales dramatically as the security level is improved with 33.93%, 45.93%, and 903.92% in the low, medium, and high cases, respectively. On the other hand, the LUT-based obfuscation incurs only 0.03%, 3.53%, and 17.82% average active power overheads for low, medium, and high obfuscations, respectively. These results validate that standby power and area dominate PPA cost as LUT unit cell size is increased. We do not include timing results as all designs maintained their original target frequency of 200 MHz for the RISC-V core and 100MHz for all other benchmarks.

6 CONCLUSION

In this work, we show, for the first time, a low overhead digital IC design obfuscation flow that is compatible with existing EDA tools and proven in silicon. We present actual area, performance, and power overheads of LUT-based obfuscation from fabricated silicon. These measurements validate the claim that LUT-based obfuscation allows for low design overheads while maintaining unparalleled SAT resiliency. We propose and characterize a security analysis technique for rapidly evaluating SAT run-time of an obfuscated design pre-fabrication. We demonstrate both nonvolatile internal (e-fuse) and volatile external (SRAM) LUT key configuration to show the solution's flexibility. Additionally, we evaluate several levels of security (low, high medium) and show that the SAT attack may be thwarted in the low obfuscation case at minimum overhead, and in the medium obfuscation case, with a maximum of 14% overhead allowing for a large SAT run time margin. Moreover, we discuss The security oriented design flow and the methodology to perform the validation of the obfuscated designs pre- and post-silicon which requires no modifications beyond a simple configuration stage. In conclusion, this study successfully shows that LUT-based obfuscation methods can be easily verified for security and functionality and can protect the design against various attacks.

REFERENCES

[1] Christopher M Bishop and Tom M Mitchell. 2014. Pattern Recognition and Machine Learning. (2014).
[2] C. Poultney *et al.* 2007. Efficient learning of sparse representations with an energy-based model. In *Advances in NIPS*. 1137–1144.
[3] D. Hammond *et al.* 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
[4] D. Shuman *et al.* 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.
[5] Michaël et.al. Defferrard. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in NIPS 29*. 3844–3852.
[6] G. Kolhe *et al.* 2019. Security and Complexity Analysis of LUT-based Obfuscation: From Blueprint to Reality. In *2019 IEEE/ACM ICCAD*. 1–8.
[7] G. Kolhe *et al.* 2021. Securing Hardware via Dynamic Obfuscation Utilizing Reconfigurable Interconnect and Logic Blocks. In *2021 58th ACM/IEEE DAC*. 229–234.
[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[9] G. Kolhe and *et al.* 2021. Securing Hardware via Dynamic Obfuscation Utilizing Reconfigurable Interconnect and Logic Blocks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 229–234.
[10] L. Xu *et al.* 2012. SATzilla2012: Improved algorithm selection based on cost-sensitive classification models. *Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions* (01 2012), 55–58.
[11] M. Bronstein *et al.* 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
[12] P. Subramanyan *et al.* 2015. Evaluating the Security of Logic Encryption Algorithms. In *Int'l Symp. on Hardware Oriented Security and Trust (HOST)*.