Robust Design-for-Security Architecture for Enabling Trust in IC Manufacturing and Test

Ujjwal Guin, *Member, IEEE*, Ziqi Zhou, *Student Member, IEEE*, and Adit Singh, *Fellow, IEEE*

***Abstract*— Due to the prohibitive costs of semiconductor manufacturing, most system-on-chip design companies outsource their production to offshore foundries. As most of these devices are manufactured in environments of limited trust that often lack appropriate oversight, a number of different threats have emerged. These include unauthorized overproduction of the integrated circuits (ICs), sale of out-of-specification/rejected ICs discarded by manufacturing tests, piracy of intellectual property, and reverse engineering of the designs. Over the years, researchers have proposed different metering and obfuscation techniques to enable trust in outsourced IC manufacturing, where the design is obfuscated by modifying the underlying functionality and only activated by using a secure obfuscation key. However, Boolean satisfiability-based algorithms have been shown to efficiently break key-based obfuscation methods, and thus circumvent the primary objectives of metering and obfuscation. In this paper, we present a novel secure cell design for implementing the design-for-security infrastructure to prevent leaking the key to an adversary under any circumstances. Importantly, our design does not limit the testability of the chip during the normal manufacturing flow in any way, including postsilicon validation and debug. Our proposed design is resistant to various known attacks at the cost of a very little ($< 1\%$) area overhead.**

***Index Terms*— Design for security (DFS), integrated circuit (IC) overproduction, obfuscation, piracy, reverse engineering (RE).**

I. Introduction

COUNTERFEITING and piracy have become major problems in the twenty-first century due to the globalization of the semiconductor industry [2]–[4]. Because of the persistent trend of device scaling and the resulting increase in the complexity of the fabrication process, most companies designing system-on-chips (SoCs) no longer maintain a fabrication unit (foundry or fab) of their own. Costs for building and maintaining such foundries are reported to be more than several billions of dollars [5]. This leads to the adaptation of horizontal integration in the semiconductor industry where the SoC designers contract foundries and assemblies for production. In parallel, the continuous trend of device scaling has enabled designers to fit more and more functionality on an SoC to reduce overall area and cost of a system. As the complexity of modern SoCs grows exponentially, it is virtually impossible to design a complete system by an SoC designer alone. Therefore, the semiconductor industry has shifted gears to the concept of design reuse rather than designing the whole SoC from scratch. Due to the lack of transparency and the resulting lack of trust may lead to the following vulnerabilities.

1) *Integrated circuit (IC) overproduction:* An untrusted foundry/assembly can produce more number of unauthorized chips [6]–[14] and can make illegitimately larger profits by selling them in the market as no research and development cost is incurred during production. Moreover, they can also practically overbuild chips at zero cost by manipulating the yield information [14]–[17].
2) *Out-of-specification/defective ICs from manufacturing:* Due to the imperfect manufacturing and assembly processes, foundry/assembly discards defective chips and sends defect free chips to the market. In a trusted environment, these defective chips are always scrapped. However, an untrusted entity in the production process (a rogue employee) can source these rejected defective chips to the gray market [14]. The application of these chips in a critical infrastructure can cause significant damage.
3) *Intellectual property (IP) piracy and reverse engineering (RE):* An untrusted foundry or its rogue employee can pirate the details of an SoC (e.g., test patterns and mask information) to a competitor company or make one or more illegitimate copies of the original IPs [13], [18]–[20]. The design details of an SoC can be reconstructed from the musk information by RE, which ultimately help to make cloned ICs [21], [22]. An untrusted foundry can also add some extra features to the SoC to introduce a backdoor or a hardware Trojan into these clone chips.

In this paper, we present a novel design-for-security (DFS) architecture to prevent the aforementioned attacks by obfuscating a netlist. The chips must be activated to unlock their full functionality before shipped them to the market. We insert locks in the netlist in such a way that the commercial automatic test pattern generation (ATPG) tools can generate test patterns without having the obfuscation key. This will provide support for performing manufacturing tests before the activation of the chips. We have added a scan flip-flop (FF) to drive a key bit such that an ATPG tool can reach to the obfuscated portion of

Manuscript received September 5, 2017; revised December 15, 2017; accepted January 19, 2018. This work was supported in part by the Intramural Grant Program from Auburn University and in part by the National Science Foundation under Grant NSF CCF 1527049. This paper was presented at the 2017 IEEE VLSI Test Symposium. *(Corresponding author: Ujjwal Guin.)*

The authors are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: ujjwal.guin@auburn.edu; zzz0044@auburn.edu; singhad@auburn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2797019

the circuit. We have provided the support such that an unlocked circuit (fully functional) blocks the scan out capability when an adversary attempts to dump the functional responses captured in the FFs through the scan chains. Due to the unavailability of scan data that contains the obfuscation key, existing attacks become unfeasible (see Section VI-A1). Note that the chips can be fully functional and structural tests can be carried out with the help of the key only in a secure environment, which can provide postsilicon debug and diagnosis support.

A. Contributions

The key contributions of this paper are as follows.

1) *Support for test before activation:* Our locked design does not require the obfuscation key during manufacturing tests and allows full scan-based structural manufacturing tests at the potentially untrusted foundry on the obfuscated design. Such structural tests can comprehensively test the circuit for virtually all faults (e.g., stuck-at, transition, and path delay faults [23]), even though the circuit is still locked. A foundry can perform the complete range of manufacturing tests on the locked chips without the need for any change in the normal IC fabrication and test flow.
2) *No capture of keys during scan tests:* The chip can also be tested, using both structural and functional tests, by untrusted end users, after the IC has been unlocked by programming the obfuscation key into the chip in a secure environment. This is allowable because our proposed solution prevents the capture of any information related to the keys during scan testing performed on even an unlocked chip. Any structural tests applied at this stage still operate on the locked obfuscated design. Basically, the programmed keys are disabled during scan-based tests. However, normal functional tests can obviously be performed on an unlocked chip configured for full functionality. The lack of access to a scan shift capability in conjunction with unlocked full functionality is a design feature that prevents an adversary from using scan to perform satisfiability (SAT)-based attacks on an unlocked IC to recover the key.
3) *Disabling scan dump after functional mode:* Furthermore, our DFS design blocks any direct transition from functional mode to scan mode. This is also a necessary feature to achieve complete protection against SAT-based attacks. Note that blocking the possibility of a scan dump in the midst of functional operation eliminates the availability of a "golden" functional circuit or "oracle" (an unlocked functional IC) with internal state visibility. In SAT-based attacks, a small set of distinguishing input scan test patterns (DIPs) are obtained from the locked circuit and incorrect keys are ruled out by any observed mismatches when the responses using candidate keys are compared with those from an unlocked functional IC or oracle. However, in the absence of visibility into the many internal FF states of a sequential circuit, any comparison of just the observable output signals provides very minimal information for each applied input pattern. This dramatically increases the complexity of any SAT-based attack, making it virtually impossible to apply against a large sequential design.
4) *Post-Si validation and debug support:* However, blocking any scan dump in the middle of functional operation can greatly complicate design error diagnosis and debug. Observe that logic design bugs and the obfuscation keys have a similar impact on the circuit; they both transform a good functional design into a faulty one. Consequently, preventing discovery of the obfuscation keys while at the same time providing support for logic error discovery and debug is inherently contradictory goals. Our proposed DFS architecture overcomes this problem with a novel design feature that necessitates availability of the actual obfuscation key for scan dump activation in the functional mode; having an unlocked chip is not sufficient. Recall that the key cannot be recovered from an unlocked chip. Thus, design debug, as well as key discovery using SAT or other formal methods, cannot be performed by an untrusted user. However, as described later in this paper, a full design debug capability is supported at a trusted site where the actual obfuscation key is available.

In summary, our DFS architecture is the first secure design that provides complete support for structural manufacturing tests, postsilicon validation and debug, and full in-system test capability, all with a very small area overhead. With respect to pinout, our proposed architecture requires only one additional global signal pin (Test).

While studying the robustness of the DFS architecture against SAT-based attacks, we have also developed a novel new attack that can discover the obfuscation keys for a sequential circuit in an efficient manner. This is a second significant contribution of this paper. We call this as the greedy attack. In this attack, an adversary simulates an obfuscated logic cone with just a few random patterns to rule out a hypothesis key. Note that the number of key bits can be very small for a logic cone when the key gates are uniformly distributed though out a sequential circuit, which consists of thousands of cones for a modern design. This is a probabilistic attack, and it cannot guarantee the elimination all possible incorrect key combinations. However, our experimental results show that we can eliminate a hypothesis key with few random patterns in most cases. Furthermore, greedy attack can be performed in combination with SAT-based attacks to efficiently find the key. Fortunately, this attack can also be prevented completely by our proposed DFS architecture.

The rest of this paper is organized as follows: In Section II, we present the background and related works. In Section III, we introduce the brute force attack based on logic cones. We also present a novel greedy attack in this section to efficiently estimate the secret key. We present our proposed DFS architecture in Section IV. In Section V, we describe our proposed approach to prevent IC overproduction and piracy and manufacturing rejection. We perform a detail evaluation of our proposed implementation in Section VI. We conclude this paper in Section VII.

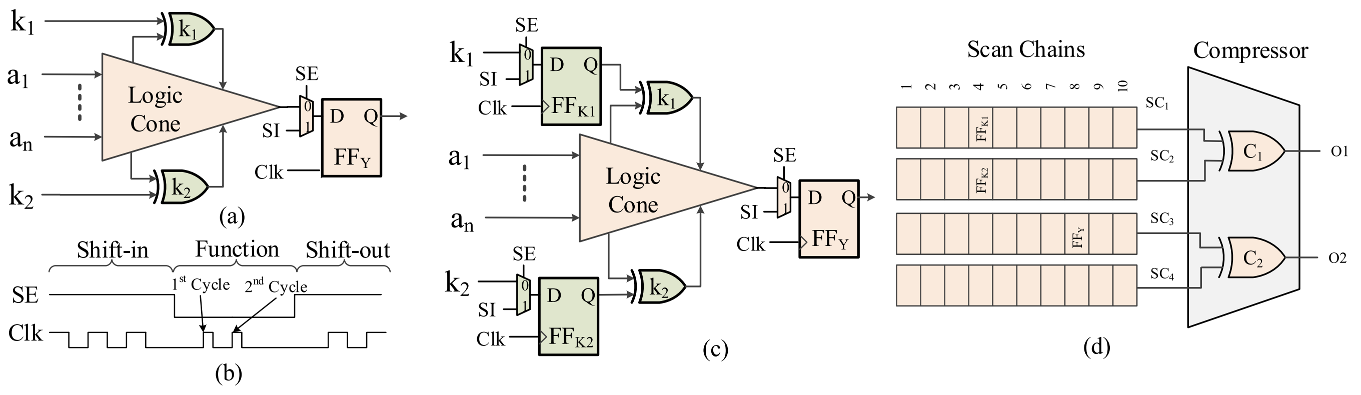


Fig. 1. Prior obfuscation approaches and their vulnerabilities. (a) Techniques proposed in [8] and [24]. (b) Timing diagram for manufacturing tests. (c) Technique proposed in [14]. (d) Attacks on [14].

II. Background and Related Work

Various countermeasures have been proposed in recent years to address the aforementioned threats. These solutions can be broadly classified into—logic obfuscation [8], [12], [24]–[26], IC metering [6]–[12], [14], [27], hardware watermarking [19], [28], [30], [31], and split manufacturing [32], [33]. Logic obfuscation and IC metering can potentially address all these attacks simultaneously. Moreover, IC metering can also be achieved through appropriate key management during logic obfuscation. IC metering aims to prevent all the aforementioned attacks by attempting to give the control over the IC manufacturing to the SoC designer [6]–[12], [14], [27]. These approaches can be either passive or active. Passive approaches register all new authorized ICs by incorporating physically unclonable functions [34]–[38] in each copy and then storing their challenge–response pairs in a secure database. Later, any suspect ICs taken from the market can be checked for proper registration. Active metering approaches are designed to automatically lock each new IC that is manufactured by a foundry until it is unlocked (activated) by the authorized SoC designers. Active metering can be efficiently implemented through logic obfuscation. This is a technique where a design is transformed to a different one to obfuscate the inner details of the original design [8], [12], [20], [24], [26], [39]. Only on the application of a programmed secret key can make the transformation reversed, thus preserving the original functionality. Roy *et al.* [8] first proposed to obfuscate a netlist by using a set of XOR/XNOR gates which can only be unlocked by using a key. Unfortunately, this design is not resistant to RE as the key controlled gates are directly related to their key bits (XOR and XNOR gates indicate 0 and 1 at the key location, respectively) and vulnerable to key sensitization attacks [24].

The solutions to prevent key discovery proposed by Rajendran *et al.* [24] appear to adequately address the above issues. However, Subramanyan *et al.* [40] have shown that the key in an activated circuit can always be exposed using scan-based manufacturing tests through SAT-based analysis. The SAT-based analysis algorithm [40] finds the correct key by ruling out incorrect ones iteratively, by using DIPs. For simplicity, the logic cone schematic is shown in Fig. 1(a) is obfuscated by two key bits, k_1 and k_2. Here, a logic cone is a combinational logic unit that represents a Boolean function, and generally bordered by FFs and input/output ports. Assume that this cone produces different outputs for $k_1 = 0$ and $k_1 = 1$ for some input pattern $[a_1\ a_2 \ldots a_n]$. Then, by observing the correct response from an activated working chip, the correct key ($k_1 = 0$ or 1) can be determined. Guin *et al.* [14], [17] proposed placing multiple FFs capturing signals controlled by different key bits [shown in Fig. 1(d)] at the same level of the parallel scan chains used in current test compression methodologies [41], [42], thereby exploiting the output compression architecture to address SAT-based attacks. Fig. 1(d) shows the architecture, where the keys (k_1 and k_2) are placed at the same level (location 4) in scan chains 1 and 2 (SC_1 and SC_2). It appears impossible to perform SAT-based attacks that discover both k_1 and k_2, as an adversary cannot access individual scan cells from the compressed output O_1. One cannot determine the key bits k_1 and k_2, as they are equally likely in the key.

A vulnerability still remains with this design in view of advances in SAT-based formal tools that can support analysis over multiple sequential clock cycles. The key may be exposed to the adversary through multicycle tests, such as delay tests to detect transition delay faults and path delay faults [23]. During these tests, the circuit response is captured multiple times (typically 2 for timing tests), which is shown in Fig. 1(b). In the first clock cycle, the key bits k_1 and k_2 are captured at FF_{k1} and FF_{k2}. Now, this key information is captured in the second clock cycle at FF_Y [see Fig. 1(c)] which can be located in a different scan chain (location 8 in SC_3). Thus, an adversary can perform a multicycle attack using a SAT-based approach. So even if the designer attempts to obscure the capture of key FF_{k1} and FF_{k2} at the end of the first cycle, an attacker can capture two or more clock cycles (FF_Y) to perform a SAT-based analysis.

Various other countermeasures have been proposed in recent years toward preventing SAT-based attacks. Yasin *et al.* [43] proposed SARLock, where a small comparator circuit is added in the design to ensure the exponential complexity of key bits. The main idea of SARLock is to make sure that each DIP can determine only one wrong key. In addition, they propose a two-layer or hybrid obfuscation mechanism, which consists of SARLock and strong logic locking (SLL) [44]. However, SARLock scheme can be broken by double DIP attack [45]. Moreover, SARLock and SLL cannot provide adequate attack resistance [46]. In another approach, Xie and Srivastava [47] proposed to add an anti-SAT block such that the exponential complexity of key search space can be preserved to protect a netlist from SAT-based attacks. Unfortunately, this scheme can also be broken by signal probability skew attack [48]. Recently, Xu *et al.* [46] proposed bypass attack to bypass SAT-resistant schemes. Moreover, Yasin *e*t al. [49] proposed a solution where the testing can be performed without activating the key. The key can be embedded in the manufacturing test patterns, or a dummy key can be applied during the test. As the functional response contains the key information, the solution can be vulnerable to the attack based on scan data [24], [50]. A different set of decamouflaging attacks have been proposed to break the security of the camouflaging circuits [51], [52]. Keshavarz *et al.* [53] proposed a solution that can circumvent this attack. Recently, Yasin *et al.* [54] proposed stripped-functionality logic locking to prevent existing attacks. No attacks have been reported so far. However, the implementation overhead can be large (e.g., 8% area overhead) for this secure implementation. Our objective is thus to design a low-cost secure solution that prevents SAT-based attacks and satisfies all the key requirements (described in Section IV-A).

We believe that SAT-based attacks are only possible due to availability of an oracle (an unlocked functional chip), that exposes the correct functionality (responses captured in the internal FFs) through scan chains. SAT-based attacks can be prevented completely, as it compares the response from an obfuscated circuit to the oracle to find DIPs, if we restrict the access of functional response through scan chains. However, a greater challenge lies when the designer places the key gates uniformly in an SoC. This is often necessary to obfuscate the netlist to hide most of its functionality. An attacker does not necessarily perform the SAT-based attacks to extract the key when they are distributed throughout the netlist. An adversary can simply search the entire key space (brute force) to find out the key. In Section III-A, we will present brute force attacks to find the key. However, brute force attacks can be unfeasible when the keys are placed in larger cones (e.g., 128 inputs). An improved version of brute force attack (we call as greedy attack) can help an adversary to find the key by using a small number of random patterns (see Section III-B for details). Toward addressing these vulnerabilities, this project focuses on designing an obfuscated circuit such that it can withstand SAT-based, brute force, and greedy attacks.

III. Attacks on Existing Logic Obfuscation Techniques

Modern electronic designs are sequential in nature and consist of combinational logic and memory elements. The outputs of a sequential circuit depend both on the inputs and its internal state. Generating test vectors to test a sequential circuit is extremely challenging as it is required to initialize the internal state before applying a pattern and then carry the response to the primary output (PO) [23]. This leads to adopt scan design, where controllability and observability are provided for the memory elements (FFs). The basic idea of scan is to convert the sequential circuit to its combinational equivalent. Each combinational block can be tested simultaneously through the scan chains. It is now very relevant to analyze the security of the obfuscated sequential circuits. In this section, we present two different attacks that can partially (full) recover the obfuscation key for sequential circuits.

A. Brute Force Attack Based on Logic Cones

For the uniform obfuscation of a netlist, it is required to distribute the key throughout the netlist such that the circuit produces incorrect result most of the time. This can create a new vulnerability that an adversary can estimate the key by using exhaustive search when a key gate is placed in a smaller cone. We call this attack as brute force attack based on logic cones, which was first introduced by Lee and Touba [55]. Brute force attack is very important to evaluate the security strength of an obfuscated design.

Brute force attacks can be performed through the scan chains, which are inserted into a design to provide manufacturing test support [23]. This insertion of scan chains converts a sequential circuit to its combinational equivalent and contains hundreds/thousands of cones with varying input sizes. If a key gate is placed in a cone with smaller number of inputs, an adversary can perform an exhaustive search to estimate the key value. In order to get a better understanding of brute force attack, it is necessary to analyze attacker's effort (AE), which can be defined as the total number of trials to estimate the key. In this attack scenario, an adversary tries all possible combinations of key and input values of a cone and observes the output of the locked circuit. For a correct key, the output must be equal to the output of that cone of an unlocked functional IC (oracle).

Let us assume a cone with n logic inputs and m key inputs. Here, $X = \{x_1, x_2, \ldots, x_{2^n}\} \in \{\{0,1\}^n\}$ represents all inputs patterns, and $K = \{k_1, k_2, \ldots, k_{2^m}\} \in \{\{0,1\}^m\}$ denotes all possible keys. Now, the input/output relations of the cone are represented by a function F, such that $Y = F(X)$. Similarly, for an obfuscated cone, it becomes $Y = F(X, K)$. For an unlocked circuit $F(x) = F(x, k_O)\ \forall x \in X$, where k_O is the obfuscation key. A brute force attack verifies for every $k_j \in K$ if

$$F(x, k_O) \overset{?}{=} F(x, k_j) \quad \forall x \in X. \tag{1}$$

The hypothesis key k_j becomes the obfuscation key, k_O if 1 holds.

Here, the AE becomes $O(2^{n+m})$ for a logic cone. Let us now study the case, where the keys are uniformly distributed across the design. The m-bit obfuscation key is distributed into r cones, where ith cone receives m_i-bit key, and $\sum_{i=1}^{r}(m_i) = m$. Here, we assume that a key bit is routed to only one cone. The (AE$_i$) for cone i becomes $O(2^{n_i+m_i})$. The overall AE will be AE $= \max(\text{AE}_i)$ as all the cones

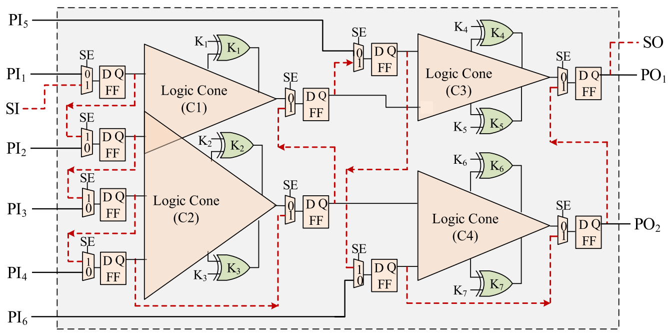


Fig. 2. Example of a scan-inserted sequential circuit.

TABLE I

PC in IWLS BENCHMARKS

Bench-mark	# Gates	# Cones	PC $\leq$ 16	16<PC $\leq$ 32	32<PC $\leq$ 64	64<PC $\leq$ 128	PC >128
S35932	16,065	2066	100.00%	0.00%	0.00%	0.00%	0.00%
S38584	19,253	1332	78.49%	19.88%	1.64%	0.00%	0.00%
S38417	22.179	1559	58.33%	16.25%	9.42%	16.00%	0.00%
b17	37,117	1311	11.06%	4.36%	11.76%	22.43%	50.39%
b18	92,048	3075	6.97%	5.04%	12.91%	14.60%	60.47%
b19	174,157	6074	6.80%	5.14%	12.59%	14.55%	60.92%

can be tested simultaneously through the scan chains (see details in McCluskey's verification test paper [56]). We call this complexity as "*best-case*" where an adversary can perform the attack simultaneously.

We will now present a short example to describe the complexity of this attack. Fig. 2 shows a sequential circuit, where seven key gates are placed. We assume that the circuit contains four logic cones, namely, C1, C2, C3, and C4 where, C1 and C2 have one overlapping input. The circuit has six inputs and two outputs. For simplicity, we assume that the circuit contains one scan chain (highlighted in dotted red). To find the correct key, an adversary will try all possible combinations. Thus, the AE for C1 (AE_1) will be 2^3. Similarly, AE for cones C2, C3, and C4 will be $AE_2 = 2^5$, $AE_3 = 2^4$, and $AE_4 = 2^4$, respectively. It is interesting to note that an adversary can brute force all the cones simultaneously by shifting the appropriate patterns through the scan chain. The number of such scan shift operations (the overall AE) is the $\max(A_i) = 2^5$, which is much smaller than the exhaustive key search (2^{6+7}) to find 7-bit obfuscation key. However, an adversary can find some key bit much quickly if they are placed in a smaller cone (e.g., C1).

However, a designer can route one key to multiple cones. For example, the m-bit obfuscation key can be distributed into r cones, where all cones receive m-bit key. In this case, the AE becomes $O(2^{n+m})$. We call this complexity as "worst case" as an adversary cannot perform brute force attacks like previously. Note that the routing congestion will increase significantly if we want to route all the key bits to different targeted cones. A compromise can be made, where a key be can be routed to few cones without increasing the routing congestion. However, due to the large number of cones (more than 6000 cones for a moderate circuit like *b19*, see Table I), a designer can only obfuscate a very small portion of the circuit. Thus, it may be very tempting for him/her to distribute the key in different cones to maximize the effect of obfuscation.

In summary, an adversary can perform brute force attacks to all the cones simultaneously through scan chains to estimate the complete m-bit key, when the key bits distributed across the circuit. He/she can find a part of key if those keys are placed in a small cone. The strength of the obfuscation depends only on the cone size, rather than the total number of bits in the obfuscation key and the primary inputs (PIs) of a complete netlist.

We have performed a simulation on IWLS benchmarks [57] to analyze the number of cones that can be targeted for brute force attacks. Table I shows the cone analysis for six different benchmarks. Based on Table I, we can find that the cone size in the netlist varies from a few inputs up to hundreds of inputs. For a small benchmark (e.g., S35932) all the cones have less that 16 inputs. For benchmark S38584, percentage of cones (PC) with less than 16 inputs is 78.49% and PC with less than 64 inputs is 100%. For these smaller benchmarks, an adversary can simulate all input and key combinations to find out the obfuscation key. As we mentioned before that an SoC designer's objective is to place the key gates uniformly to have an higher obfuscation impact on the circuit. Each cone may have very few key gates. For larger benchmarks (e.g., b19), PC with less than 16 inputs is 6.8%, whereas PC with less than 32 inputs, and greater than 16 inputs is 5.14%. Thus, an adversary can find few key bits if the keys are uniformly distributed across the circuit. However, an SoC designer can place the keys in larger cones to prevent this attack.

B. Greedy Attacks on Logic Cones

Brute force is an efficient approach to obtain the key value especially when the cone size is small. However, when the size of the cone becomes larger, the brute force attack may not be feasible as the AE remains exponential complexity with the number of inputs. In this section, we present a novel attack that greatly reduces AE for a circuit. We refer to this attack as greedy attack. Instead of applying every input combinations, an adversary greedily selects a few patterns to recover the secret key.

In greedy attack, an adversary simulates a cone with few random patterns. Then, the same patterns are applied to the same cone of an unlocked chip (oracle) to receive the correct response. If the comparison fails, it is guaranteed that the hypothesis key used during simulation is not the obfuscation key. The greedy attacks iterate all possible key combinations to rule out all hypothesis keys. Note that the number of key bits is very small for uniform obfuscation. This is a probabilistic attack, and it cannot guarantee to find the obfuscation key. However, our experimental results show that we can rule out a hypothesis key with few random patterns in most of the cases.

Greedy Attack: The hypothesis key, k_j is not the obfuscation key if

$$\exists x \in X^P : \; F(x, k_j) \neq F(x, k_O). \tag{2}$$

X^P is the set of p randomly selected patterns. The complexity of greedy attack is $O(p \times 2^{m_i}) \approx O(p)$, where m_i (can be

TABLE II

GREEDY ATTACKS on SMALL CONES

Key Size	Cone Size	The Number of Get 1 in 10000 Patterns					
		C1	C2	C3	C4	C5	C6
1	44-56	208	11	272	2031	898	87
	94-106	101	57	1742	3	28	15
2	44-56	886	321	461	1760	329	98
	94-106	633	391	29	65	182	59
4	44-56	2715	327	402	911	946	5
	94-106	236	417	62	106	31	169
8	44-56	1958	293	2162	414	1391	1269
	94-106	520	377	3354	378	64	292

TABLE III

GREEDY ATTACKS on LARGE CONES

Key Size	Cone Size	The Number of Getting 1 in 200000 Patterns					
		C1	C2	C3	C4	C5	C6
1	144-156	2	3	412	3	3	6
	194-206	6	14	19	7096	7	9
2	144-156	5	3	5	25	1	6
	194-206	2	1686	6	131	17	13
4	144-156	13	2	6	12623	5	3
	194-206	2	4	11	30	221	906
8	144-156	13	4	1511	11	3	13
	194-206	7	632	12	2726	1573	3408

very small, e.g., 1) is the key size of the ith cone. This attack is feasible when a designer uniformly distributes the keys in their design to have a greater impact of obfuscation.

To validate this attack, we perform an experiment by using Synopsys Design Compiler [58] and VCS [59] on few IWLS 2005 [57] benchmarks. We have found out that we can find a wrong key of different sizes (1-bit, 2-bit, 4-bit, and 8-bit) by using only 10K random input patters for a small cone size (see Table II) and 200K random patterns for large cones (see Table III). We use a Perl script to extract few cones from benchmarks b17, b18 and b19, and use VCS to perform the simulation.

Table II shows the simulation results for few small cones (less than 128 bits) from the ITC 99 benchmarks. We apply 10K random patterns and observe the responses. Six different cones (C1–C6) are randomly selected to perform the greedy attack. Column 1 represents the number of key gates that are placed in these cones. Column 2 represents cone size. We select a cone that is mentioned in the range. Rest of the columns show the number of times Equation 2 are satisfied. From Table II, it is clear that almost all the cones produce incorrect results most of the cases.

Table III shows the greedy attack on large cones. The larger cones require more random patterns to find a mismatch that satisfies 2. We apply 200k randomly patterns. It generally takes less than a minute to apply all these patterns to perform this attack. The simulation is performed in HP Z840 Workstation with Intel Xeon E5-2620 v3 (2.4 GHz/6 cores) processor and 64 GB of RAM. The majority of the cases, the adversary finds an incorrect key effectively in few minutes.

In summary, existing logic obfuscation techniques suffer from three different attacks—brute force, greedy, and SAT-based attacks. Our objective is to design an obfuscation technique that can effectively circumvent all these different attacks. Alternatively, we can state that we require a design solution that prevents access to the response of a logic cone through scan chains. Without an oracle, an adversary cannot compare the simulation results with the oracle and perform such attacks.

IV. PROPOSED DESIGN-FOR-SECURITY IMPLEMENTATION

A. Requirements of DFS Implementation

This section provides an in-depth analysis for all the requirements for successfully preventing IC overproduction, manufacturing rejection, and IP piracy.

1) Attack Resistance: The netlist must be designed in such a way that the chip never leaks the key (during either tests or normal functions), which makes the design resistant to various known attacks [1], [24], [40], [45], [48]. Finding of a key must satisfy NP completeness, and the key must be kept long enough such that brute force attacks become impractical. In addition, the key must be resistant to RE attack, where an attacker must not find the key by looking at the circuit netlist. Direct mapping of the key bits to XOR or XNOR gates are prohibited.

2) Uniform Distribution of the Key: The key gates need to be placed uniformly to a design to obfuscate its significant part. As the modern designs are sequential in nature, care needs to be taken to place a key gate. It can be subjected to brute force attacks (see Section III-A). It can also be vulnerable to greedy attacks (see Section III-B) irrespective of the size of the cone. In addition, any cones are subjected to SAT-based attacks. The obfuscation scheme must address all these attacks.

3) Structural Test Capability Without the Key: Allowing structural tests before the activation is one of the key requirements for preventing the overproduction of chips. It is necessary to add capability which permits a foundry or assembly to perform structural tests right after manufacturing and discard the defective chips. One can argue that tests can be performed at the SoC designer's site. However, it requires additional test setup for the SoC designers, which they may not have. In addition, it is not wise to send chips to the SoC designers without tests which require addition transportation. However, the greater challenge is that the foundry cannot stabilize the process unless they monitor the outcome. Thus, it is absolutely required that the tests have to be performed at the manufacturing site.

4) Postsilicon Validation and Debug Capability: The circuits must be modified in such a way that it does not impact the postsilicon validation and debug, where the chips generally run at speed and scan dumps may be required to obtain high observability of internal nodes.

5) Full In-System Test Capability: The obfuscated circuit must support in-system test capability. It is absolutely necessary that a chip does not leak key information to its POs while it is in functional mode. In this mode, a set of functional test vectors is required to test a design. While testing, it is required that each module (IPs) to be initialized to the desired state. Setting that state of a complex industrial circuit through PIs becomes a major challenge and could potentially take millions of clock cycles [60]. Thus, test engineers often shift the state

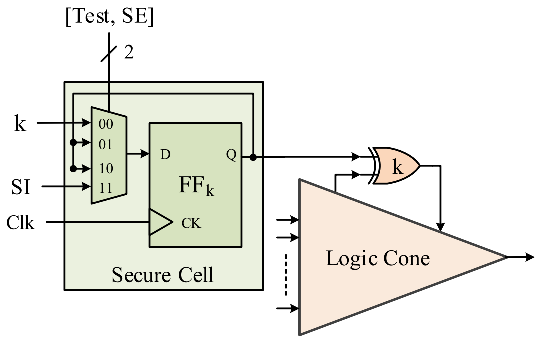


Fig. 3. Proposed SC architecture.

TABLE IV
MODES OF OPERATION

Test	SE	Mode	Description
0	0	M0	The chip is in functional mode. The secure cell applies key to the logic.
0	1	M1	The secure cell holds its previous value. The rest of the circuit is in functional/shift mode depending on the *SE*.
1	0		
1	1	M2	The SC becomes scan cell and it becomes a part of the scan chain.

through existing design-for-test (DFT) structure [61]. It is thus required that keys do not impose any limitation to this hybrid testing.

B. Proposed Design-for-Security Architecture

The objective in designing the new DFS architecture is to prevent the key getting exposed during manufacturing tests. We have mentioned in Section IV-A that if the key information is captured during a test, it will eventually be exposed to the POs of a working (unlocked) chip and an adversary can effectively retrieve the key.

Fig. 3 shows our proposed SC architecture used for design for security. We modify the scan cell in such a way that it can hold its previous state. The output of FF$_k$ is fed back to the its input through a multiplexer (MUX). As the MUX has four inputs, we need one additional *Test* pin for the MUX control. Depending on the value of Test and SE pins, a particular input is selected. The key bit (k) and scan in (SI) are connected to the first and fourth inputs of the MUX, respectively. The output of FF$_k$ is connected to the second and third inputs, which provides the capability to hold its previous state.

The SC operates in three different modes based on Test and SE, which is shown in Table IV. In mode $M0$, FF$_k$ captures the key k, which represents the normal functionality of the unlocked chip. The chip will be operated in this mode while it is in the field. In mode $M1$, the SC continues to hold its previous state. This mode provides test and debug capability without letting the key to be exposed as FF$_k$ continues to hold its previous state. Thus, no key information is captured in $M1$. Note that the rest of the circuit becomes functional mode when SE $=$ 0 and scan mode (shift in or shift out) when SE $=$ 1. Finally, SC becomes the scan cell at mode $M2$ and FF$_k$ becomes a part of a scan chain.

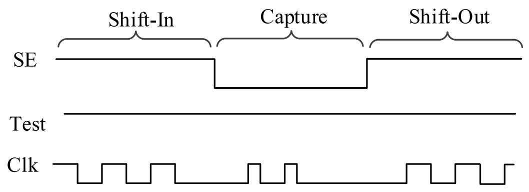


Fig. 4. Timing diagram for manufacturing tests (delay tests).

1) Manufacturing Test: The implementation of manufacturing tests using our proposed SC does not require any additional modifications in the existing test infrastructure. Note that the key is not programmed at this stage (see Fig. 7). It is required to keep Test pin active high (logic 1) during the test. During the scan shift-in phase, the SCs become a part of a scan chain ($\{Test, \text{SE}\} = \{1, 1\} = M2$) and receive values generated by the ATPG tool. Note that the key gate (k) (see Fig. 3) is directly reachable from the SI.

During the test response capture, the rest of the circuit becomes functional while the SCs hold their current state ($\{Test, \text{SE}\} = \{1, 0\} = M1$). No key bits are captured in the SCs as they continue to hold the states received during scan shift-in phase. This helps us to eliminate all the attacks completely. Finally, the captured functional responses are shifted out through the scan chain ($\{Test, \text{SE}\} = \{1, 1\} = M2$).

2) Postsilicon Debug and Validation: Complex modern designs can suffer from subtle logic and electrical design bugs that escape design verification and are only discovered in first silicon. This necessitates support for postsilicon validation, and if a bug is discovered, its diagnosis followed by design changes to correct the problem. Postsilicon debug is extremely challenging, and at a minimum requires both a fully functional test (on the activated design) as well as extensive scan test support. This extent of intrusive testing of the fully functional circuit can make it vulnerable to key discovery through SAT-based attacks or other formal tools. We therefore allow such full testing only in a secure design environment, with the key again applied through the scan chain, even if it is already programmed.

Full scan tests on the fully functional circuit are performed in mode $M1$ (Table IV). Recall that in this mode, with the scan enable low (functional mode), the programmed key bits are not captured in the SCs from where they are presented to the logic; instead, the SCs are designed to hold and retain their current value. Thus, if the key bits are shifted into the SCs during the scan shift in $M2$, and the scan enable (SE) is then lowered to the functional mode ($M1$), the scanned in key bits will be retained in the SCs ensuring unlocked functional operation as long as the scan enable stays low. Single or multicycle tests can be performed and the results shifted out ($M2$).

3) Functional Tests: The functional test can only be performed after the activation of the chips. Mode $M0$ supports functional tests. Functional patterns are applied to the PIs of a chip, and the responses are collected at the POs. It is required to initialize the finite state machine of a design before actual tests are performed, and sometimes could lead to millions of clock cycles [60]. Test engineers often shift this initialization

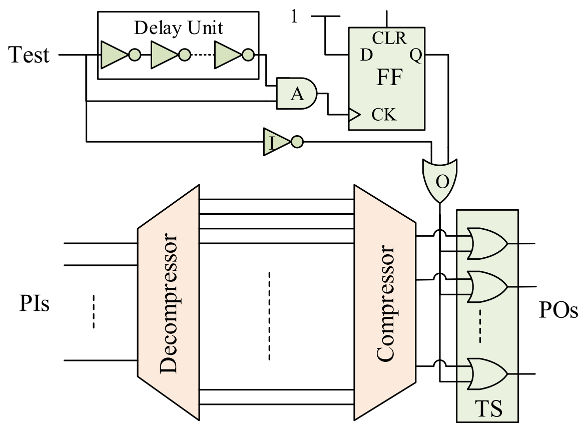


Fig. 5. Proposed architecture to restrict scan data access.

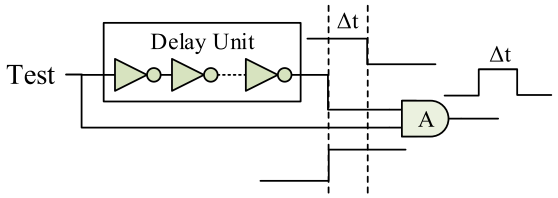


Fig. 6. Pulse generator module for detecting a positive transition at Test pin.

state through existing scan architecture. Mode $M2$ can be used to shift this state to the design, and then, it is switched to mode $M0$.

4) Mode Control: An important restriction on switching between different operation modes for the SC is absolutely necessary for maintaining security. Switching from $M0$ to $M2$ ($M0 \rightarrow M2$ or $M0 \rightarrow M1(\text{Test} = 1) \rightarrow M2$) cannot be permissible. To be specific, any positive transition at the Test pin will not be permitted. The key will be captured in $M0$ and be shifted out while the cell is in $M2$, if we allow this to happen. In addition, we will not allow shift out when Test is not asserted (i.e., Test $= 0$), which will prevent an adversary getting scan data (from SC to end of the scan chain may be shifted out while setting SE $= 1$) during Test $= 0$.

Fig. 5 shows our proposed architecture to restrict scan data access. We have added a series of OR gates at the output of the compressor (Test data compression is widely accepted by the industry [41], [42]), which is highlighted in green. The output of the test suppressor (TS) block becomes always 1 when the output of the OR gate (denoted by O) is asserted. One can place TS block before the compressor; however, the number of OR gates will be increased significantly.

The output of the OR gate O becomes 1 when one or both inputs become 1. This ensures that an adversary cannot access scan data while Test $= 0$, which is one of the requirements for protecting the key. Now, we need to make sure that there is no positive transition on the Test pin.

Fig. 6 shows a pulse generator, when it experiences a positive transition of the Test pin. The delay unit consists of odd number of inverters and is fed to an AND gate A. A pulse with duration Δt is generated at the output of gate A. The width of this pulse Δt can be controlled by manipulating the number of inverters.

Algorithm 1 Place All Secure Cells in a Circuit

```
input : Scan inserted netlist, key size ($|K|$)
output: Obfuscated and locked netlist
1  Read scan inserted netlist ;
2  Fault simulation to find all untestable faults ;
3  $UF \leftarrow$ Sort the nets based on the number of untestable
     faults in the descending order ;
4  $L \leftarrow UF.size()$ ;
5  if $N \geq |K|$ then
6      for $i \leftarrow 0$ to $|K-1|$ do
7          Insert a key gate at $net_i$ ;
8          Place a secure cell ;
9      end
10 else
11     for $i \leftarrow 0$ to $|L-1|$ do
12         Insert a key gate at $net_i$ ;
13         Place a secure cell ;
14     end
15     for $i \leftarrow |L|$ to $|K-1|$ do
16         Insert a key gate at the randomly location of the
             netlist ;
17         Place a secure cell ;
18     end
19 end
```

The output of the AND gate A is fed to the clock input of the FF shown in Fig. 5. When the FF detects a pulse, logic 1 will be captured and its output becomes 1 permanently. This FF can be cleared once during power up or after certain clock cycles (length of the scan chain) depending on one's choice. It is worth mentioning here that the Test pin can also be fed to the clock input of FF.

5) Secure Cell Placement: Higher fault coverage (FC) is often required to ensure the high yield of good chips [23]. In a circuit, there are many untestable faults due to the controllability and/or observability issues. Test point insertion is widely used to detect many of these untestable faults and thus increase the FC of a circuit. Our proposed SC can be used as a test point. Thus, the objective of placing a SC (e.g., one key bit) in the netlist in such a way that it provides the detection capability of untestable faults.

Algorithm 1 determines the key location such that FC can be increased by sorting the nets (fault locations) based on the number of faults present in them. The two scenarios may arise. First, the number of such nets (L) is greater than key size, $|K|$ (lines 5–9). This may arise when the design has many untestable faults and we have enough nets to place the key gates. Second, the number of such nets, L is less than key size, $|K|$ (lines 11–18). L key gates are placed first (lines 11–14), and the remaining key gates are placed randomly (lines 15–18). Note that the SC introduces few new faults to the design that can reduce the overall FC.

V. Proposed Flow for Enabling Trust in IC Manufacturing and Test

The primary requirement for preventing IC overproduction and IP piracy is to obfuscate a design with a key which is

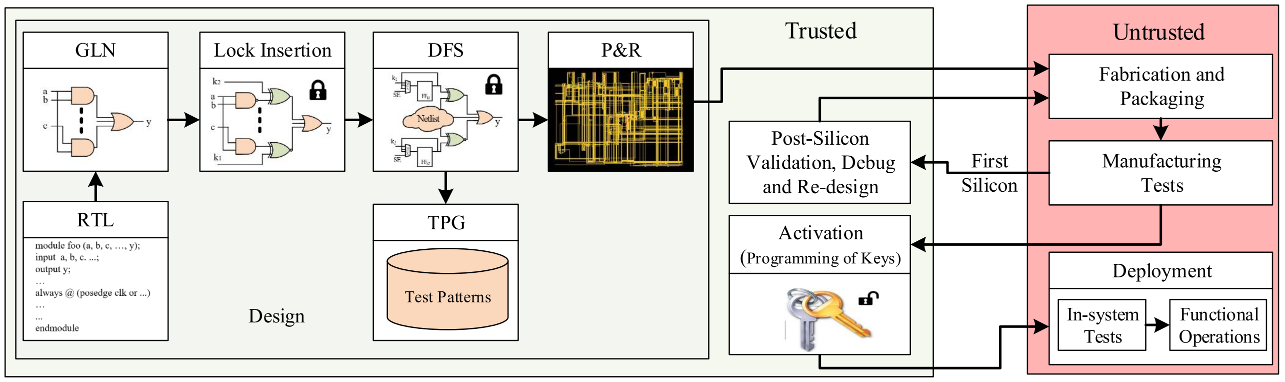


Fig. 7. Proposed flow for enabling trust in IC manufacturing and test.

resistant to all known lines of attack. The design must support all the requirements (mentioned in Section IV-A) for the obfuscation key. The manufacturing tests can be performed at the foundry and/or package assembly as these tests do not require any key. It is also important to implement manufacturing tests before the activation of chips as an untrusted foundry can manipulate the yield information (the ratio of the defect free chips to the total number of chips) with the SoC designer and stockpile a large number of chips without contributing any design costs. In addition, it can source defective or out-of-specification chips to the market if the chips are activated before the manufacturing tests. Thus, it is necessary to activate the chips in a trusted environment such that an untrusted entity cannot get any undue advantages. In summary, manufacturing tests can be performed at any untrusted site; however, activation must be executed at a trusted site. It is also important to note that postsilicon validation and debug require the chip to be fully functional (activated) with full structural test capability enabled.

Fig. 7 shows the overview of our proposed flow for enabling trust in IC manufacturing and test. This flow is exactly the same as existing IC design and fabrication process, except for the lock insertion, DFS, and activation stages. The process starts with the RTL design phase, and then, it goes through synthesis to obtain gate level netlist (GLN). A set of key gates are now inserted to lock the GLN, which can only be unlocked through a proper key. We recommend adopting one existing RE-resistant lock insertion technique [24] such that an adversary cannot find the key by simply observing the key gates. Note that the left half of Fig. 7 highlighted in green (design phase, activation, and postsilicon validation and debug) is under designer's control and is trusted. On the other hand, the right half of Fig. 7 highlighted in red (fabrication and packaging, manufacturing tests, and deployment) is untrusted.

Even if the keys are resistant to RE, an adversary can still discover the key by using brute force, greedy, and SAT-based attacks. To avoid the key being exposed to these attacks, we propose to insert novel secure scan cells (see Section IV-B for details) to the key gates. This makes the keys resistant to known manufacturing test-related attacks. As all the key gates are reachable through these SCs, it is not required to provide key information to the ATPG tool for generating test patterns. It is worth mentioning that the keys now satisfy all the requirements mentioned in Section IV-A. After DFS stage, the design is moved to the place and route (P&R) stage, and then Graphic Database System II files are created. Finally, they are sent to foundry for fabrication and packaging.

After manufacturing the first batch of chips, the foundry performs manufacturing tests and sends to the SoC designer for postsilicon validation and debug, where it validates correct behavior in actual application environments. Any bugs may have been undetected previously during presilicon verification. During this stage, the SoC designer performs many different tests (combination of structural and functional) and observes the internal states to detect and diagnose any bugs. Our proposed DFS architecture provides the postsilicon validation and debug support which is absolutely required for SoC design and fabrication process.

Once the postsilicon validation and debug is complete, the SoC designer provides the contract to a foundry to fabricate a certain number of chips. After fabrication, the foundry performs manufacturing tests and sends the defect-free dies to the assembly for packaging. The assembly performs final tests and sends back the chips to the SoC designer for activation. Finally, SoC designer activates the chips and sends them to the market for deployment. In-system functional tests can be performed on these activated chips in the field to test for their correct functionality.

Note that the left half of Fig. 7 highlighted in green (design phase, activation and post-silicon validation and debug) is under designer's control and is trusted. On the other hand, the right half of Fig. 7 highlighted in red (fabrication and packaging, manufacturing tests, and deployment) is untrusted and the keys may get compromised due to various attacks (SAT, RE, etc.). As the key information is not leaked (see Section IV-B) for our proposed design during any tests, we can safely say that these attacks are ineffective in extracting the key.

VI. Results and Analysis

A. Security Analysis

Ensuring security by protecting the key being exposed to an adversary is our prime objective. In this section, we will present different known attacks for security evaluation.

1) Attack Resistance: All different attacks (e.g., brute force, greedy, and SAT-based attacks) are primarily based on the actual observation of the response of a logic cone through the scan chains of a circuit. As long as the key information is captured during functional mode and then dumping the responses through scan chains, the key will be exposed to the aforementioned attacks. Our proposed design is resistant to these attacks as the SC prevents an adversary to capture key information while testing. SC is designed in such a way that it holds its previous state when the chip experiences tests. In addition, we impose restrictions for mode switching (mode $M0$ to mode $M2$) to access scan data. An adversary cannot extract functional response through the scan chains. He/She can only observe all 1s, when he/she tries to dump the scan data which contain the key information.

Now one may argue that an adversary can perform SAT-based attacks by observing the functional responses. The inability of the attacker in our approach to use the blocked scan chains in a legally acquired unlocked chip limits the attacker's ability to obtain the complete set of internal FF states from an available functional part. Only the small number of input/output signals, along with perhaps a few additional signals captured in accessible internal memory, is available as input data for the SAT solver in its attempt to evaluate the obfuscation key. The logic states of the much large number of internal FFs (in each execution state) are no longer available as traces and must be computed by the SAT solver through extensive analysis over many sequential time frames. Consequently, the number of unassigned variables explodes dramatically for the SAT solver. Informally, this increase in computational complexity can loosely be compared to the increase in complexity of sequential ATPG over that of single time frame scan-based combinational ATPG. Practical state-of-the-art SAT solvers today are unable to handle sequential analysis for large designs beyond a few dozen time frames due to the explosion in problem size [62], [63]. Thus, our approach clearly makes mounting a SAT attack extremely difficult, if not impossible. Earlier proposals that allow scan tests in the unlocked mode can be attacked by repeatedly analyzing a single time frame using SAT for multiple test inputs.

Our proposed solution is also resistant to the attack proposed in [50]. This attack is based on an optimization problem, where the correct key assignment is found by maximizing FC with M test stimuli (S) and responses (R). The attack can be described as follows:

$$\begin{aligned}
&\max \text{FC}\\
&\text{s.t. } CamoCkt(S_1, A, \cdot) = R_1\\
&\qquad CamoCkt(S_2, A, \cdot) = R_2\\
&\qquad\quad \cdot\\
&\qquad CamoCkt(S_M, A, \cdot) = R_M\\
&\qquad Solve for A
\end{aligned}$$

where *CamoCkt()* is the locked circuit. S_i and R_i are the ith stimulus and response, respectively. A is the correct key assignment.

Note that the attack is feasible when the obfuscation key impacts the FC. In our proposed approach, the key has no impact on the FC. Preciously, each key gate is directly reachable to the ATPG tool through the scan chains (see Fig. 3). There are two possible scenarios that can happen: 1) ATPG tool assigns the same A during each test pattern generation. In this case, the attack will result A. Note that A is not the obfuscation key. 2) The ATPG tool assigns random A ($A_i \neq A_{i-1}$). The attack may find one of the A_i. Again, none of the A_is are the key. Thus, Oracle-Less Attack proposed by Yasin *et al.* is not feasible to our proposed solution.

The security of our proposed approach lies on the length of the key. A key must be long enough such that it can withstand exhaustive key search, as our proposed design is resistant to brute force, greedy, and SAT-based attacks, and maintains NP completeness. As no key information is captured during the test, an attacker must try at least $2^{|K|}$ combinations to make the circuit completely functional. Here, $|K|$ is the length of the key. It is computationally unfeasible to find a correct key when $|K|$ is greater than 128 considering current computing resources. However, one can use 256 or higher bit keys for obfuscating a netlist considering future computing resources.

2) Tampering: An untrusted foundry can modify the masks to bypass the mode control logic (see Fig. 5 and write a permanent "zero" value at the output of the OR gate, O. In this case, an adversary has the full control of changing the modes ($M0$ to $M2$) and perform SAT-based attacks to find the key. Fortunately, this attack can easily be detected by the SoC designers and can be prevented. If the foundry manufacture chips with the tampered masks and send chips to the SoC designer for activation, he can easily detect the tampering by switching the modes and observe data. The scan data will be all 1s if it not tampered.

Now an untrusted foundry can maintain two (one tampered and one genuine) sets of masks, and send chips to the SoC designer those are manufactured with genuine masks. For the worst case, it can send one tampered chip (fabricated with the tampered masks) along all genuine chips hoping that the SoC designer will burn the key and thus can get hold of the scan data (key) from this working chip (bypassed our security measures). To circumvent this attack, the SoC designer needs to verify the chip before activating. It is important to note that the reputation of a foundry will be demolished if the SoC designer detects tampering. Moreover, it is extremely expensive to design a new set of masks, we believe that there is little economic incentive for an untrusted foundry to maintain two different sets of masks.

B. Area Overhead Analysis

The area overhead for our proposed approach is primarily resulted from four parts.

1) *SC module:* This SC contents two parts: a 4-to-1 MUX and a scan FF. The SC can switch among three modes: functional mode, hold mode, and scan mode. Based on our proposed structure, this will not disclose the key during any time. For a single SC (a 4-to-1 MUX and a scan FF), it usually contents 20 gates. The number of SCs are equal to the key length $|K|$, as each key bit is fed to a different SC. We require 256 (128) SCs when $|K|$ is 256 (128) to maintain long-term security. The approximate gate count for an SC is around 20.

TABLE V
TEST METRIC COMPARISON

Benchmark	Key Bits	Test Coverage				Pattern Count				Area Overhead
		ORG	KEY	DFS	Change(%)	ORG	KEY	DFS	Change(%)	
s35932	128	100.00%	100.00%	100.00%	0.00%	56	55	65	18.18%	6.14%
s38584	128	100.00%	100.00%	100.00%	0.00%	536	549	565	2.91%	7.84%
s38417	128	100.00%	100.00%	100.00%	0.00%	1,133	1,124	1,115	-0.80%	6.75%
b17	128	99.92%	99.91%	99.80%	-0.11%	2,542	2,516	2,559	1.71%	4.23%
b18	128	99.54%	99.60%	99.58%	-0.02%	5,086	5,112	5,116	0.08%	1.51%
b19	128	99.65%	99.65%	99.64%	-0.01%	9,395	9,387	9,398	0.12%	0.80%

2) *Keys gates:* The size due to keys also depends on the length of chip unlock key. To implement one key bit, we need one XOR/XNOR gate.
3) *Test suppression:* The number of OR gates is equal to the compressor output. We can safely assume that we require 100 OR gates for Test Suppression.
4) *Mode control:* We need approximately 20 gates to implement this module.

From the above analysis, we conclude that the majority of the overhead results from the SCs (number of key bits). The total gate count for our proposed approach is approximately 5200 when we consider 256-bit key. This can be reduced significantly to 2700 when the key is 128 bit long. For a large benchmark (e.g., b19), the area overhead is less than 1% (see Column 11 of Table V). However, it can be very less ($\ll 1\%$) for a modern industrial design with millions of gates. *Note that we need one additional pin (Test) to provide DFS support.*

C. Simulation Results

To evaluate the effectiveness of our proposed DFS architecture, we use Synopsys tools (Design Compiler [58], TetraMax [64], and VCS [59]) to perform the simulation by using Synopsys 32-nm SAED32 EDK Generic Library [65] on IWLS 2005 [57] benchmark circuits. Table V shows the test metrics comparison between different methods. We compare the test coverage and pattern counts among the original netlist with no locks (denoted as ORG), key gate-inserted netlist (KEY) and our proposed DFS-inserted netlist (DFS). Column 2 shows the size of the obfuscation key, which represents the number of key gates to be insert into the netlist. Columns 3–5 represent the test coverage. The test coverage for KEY is computed by applying a preset key (all 0s) during test pattern generation; on the other hand, we do not need any key information for DFS. Column 6 represents the percentage change of the test coverage from KEY to DFS. We do not expect any significant change in the test coverage. However, we loose a small percentage for some benchmarks. This can be due to the added faults from the MUX of the DFS architecture. Similar analysis can be performed for the test pattern counts (shown in Columns 7–9). We see a minor increase in the pattern counts for moderate and large size benchmarks.

VII. CONCLUSION

In this paper, we have proposed a novel SC design for implementing DFS infrastructure that successfully prevents the leaking of obfuscation key to an adversary, and thus establishes trust in semiconductor manufacturing. First, our proposed SC disables scan dump after functional mode. This provides a complete protection against all different attacks that require an oracle (an unlocked functional chip) to compare the simulation responses. As scan dump is not allowed during normal operations using scan chains, an adversary cannot have access the functional responses of an unlocked chip. Second, our proposed SC enables manufacturing tests before the activation of chips, which is absolutely necessary to prevent unauthorized overproduction of chips. This allows the fully scan-based tests at the foundry on the obfuscated design. Finally, the DFS architecture provides complete support for post-Si validation and debug to ensure perfect compliance to its specifications by correcting subtle logic and electrical design bugs that escape design verification and are only discovered in first silicon. In summary, our proposed solution provides a secure metering and obfuscation technique without modifying the existing IC manufacturing and test flow with the cost of a small area overhead. However, we need one additional pin (Test) to provide DFS support.

REFERENCES

[1] U. Guin, Z. Zhou, and A. Singh, “A novel design-for-security (DFS) architecture to prevent unauthorized IC overproduction,” in *Proc. IEEE VLSI Test Symp. (VTS)*, Apr. 2017, pp. 1–6.

[2] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, “Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain,” *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[3] U. Guin, D. DiMase, and M. Tehranipoor, “Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead,” *J. Electron. Test.*, vol. 30, no. 1, pp. 9–23, 2014.

[4] M. M. Tehranipoor, U. Guin, and D. Forte, *Counterfeit Integrated Circuits: Detection and Avoidance*. New York, NY, USA: Springer-Verlag, 2015.

[5] A. Yeh, “Trends in the global IC design service market,” DIGITIMES Res., Tech. Rep., Mar. 2012. [Online]. Available: http://www.digitimes.com/news/a20120313RS400.html?chid=2

[6] F. Koushanfar and G. Qu, “Hardware metering,” in *Proc. IEEE-ACM Design Autom. Conf.*, Jun. 2001, pp. 490–493.

[7] Y. M. Alkabani and F. Koushanfar, “Active hardware metering for intellectual property protection and security,” in *Proc. USENIX Secur. Symp.*, 2007, pp. 20:1–20:16.

[8] J. A. Roy, F. Koushanfar, and I. L. Markov, “EPIC: Ending piracy of integrated circuits,” in *Proc. Design, Autom. Test Europe*, Mar. 2008, pp. 1069–1074.

[9] R. Chakraborty and S. Bhunia, “Hardware protection and authentication through netlist level obfuscation,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 674–677.

[10] Y. Alkabani, F. Koushanfar, and M. Potkonjak, “Remote activation of ICs for piracy prevention and digital right management,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 674–677.

[11] J. Huang and J. Lach, “IC activation and user authentication for security-sensitive systems,” in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust*, Jun. 2008, pp. 76–80.

[12] A. Baumgarten, A. Tyagi, and J. Zambreno, “Preventing IC piracy using reconfigurable logic barriers,” *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 66–75, Feb. 2010.

[13] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer-Verlag, 2012.

[14] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, “FORTIS: A comprehensive solution for establishing forward trust for protecting IPs and ICs,” in *Proc. ACM Trans. Design Autom. Electron. Syst. (TODAES)*, 2016, p. 63.

[15] G. K. Contreras, M. T. Rahman, and M. Tehranipoor, "Secure split-test for preventing ic piracy by untrusted foundry and assembly," in *Proc. Int. Symp. Fault Defect Tolerance VLSI Syst.*, Oct. 2013, pp. 196–203.

[16] M. T. Rahman, D. Forte, Q. Shi, G. K. Contreras, and M. Tehranipoor, "CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2014, pp. 46–51.

[17] U. Guin and M. M. Tehranipoor, "Obfuscation and encryption for securing semiconductor supply chain," in *Hardware Protection Through Obfuscation*. New York, NY, USA: Springer-Verlag, 2017, pp. 317–346.

[18] E. Castillo, U. Meyer-Baese, A. García, L. Parrilla, and A. Lloris, "IPP@HDL: Efficient intellectual property protection scheme for IP cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 5, pp. 578–591, May 2007.

[19] A. B. Kahng *et al.*, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2001.

[20] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[21] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *Proc. 11th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, 2009, pp. 363–381.

[22] M. M. Tehranipoor, U. Guin, and S. Bhunia, "Invasion of the hardware snatchers," *IEEE Spectr.*, vol. 54, no. 5, pp. 36–41, May 2017.

[23] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, Nov. 2000. [Online]. Available: http://www.springer.com/us/book/9780792379911

[24] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2012, pp. 83–89.

[25] J. Rajendran *et al.*, "Fault analysis-based logic encryption," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 410–424, Feb. 2015.

[26] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in IC piracy with test-aware logic locking," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 6, pp. 961–971, Jun. 2015.

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

[28] E. Charbon, "Hierarchical watermarking in IC design," in *Proc. Custom Integr. Circuits Conf.*, May 1998, pp. 295–298.

[29] A. B. Kahng *et al.*, "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2006.

[30] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Designs: Theory and Practice*. New York, NY, USA: Springer-Verlag, 2003.

[31] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 20, no. 10, pp. 1253–1261, Oct. 2001.

[32] R. W. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," U.S. Patent 7 195 931 B2, Mar. 27, 2007.

[33] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and secure intellectual property (IP) design with split fabrication," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2014, pp. 13–18.

[34] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2002, pp. 148–160.

[35] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 9–14.

[36] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for ip protection," in *Proc. Int. Workshop Cryptogr. Hardw. Embedded Syst.*, 2007, pp. 63–80.

[37] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," *Towards Hardw.-Intrinsic Secur.*, pp. 3–37, Oct. 2010. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-642-14452-3_1

[38] A. Maiti and P. Schaumont, "The impact of aging on a physical unclonable function," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 9, pp. 1854–1864, Sep. 2014.

[39] X. Zhuang, T. Zhang, H.-H. S. Lee, and S. Pande, "Hardware assisted control flow obfuscation for embedded processors," in *Proc. Int. Conf. Compil., Archit., Synthesis Embedded Syst.*, 2004, pp. 292–302.

[40] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 137–143.

[41] *Compression for Highest Test Quality and Lowest Test Cost*, Synopsys, Mountain View, CA, USA, 2015.

[42] P. Nagaraj, "Choosing the right scan compression architecture for your design," Cadence Design Syst., 2015. [Online]. Available: https://www.cadence.com/rl/Resources/white_papers/Test_Compression_wp.pdf

[43] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 236–241.

[44] M. Yasin, J. J. V. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016.

[45] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. ACM Great Lakes Symp. VLSI*, 2017, pp. 179–184.

[46] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Proc. Int. Conf. Cryptogr. Hardw. Embedded Syst. (CHES)*, 2017, pp. 1–21.

[47] Y. Xie and A. Srivastava, "Mitigating sat attack on logic locking," in *Proc. Int. Conf. Cryptogr. Hardw. Embedded Syst.*, 2016, pp. 127–146.

[48] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-SAT," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 342–347.

[49] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of logic encrypted chips: Pre-test or post-test?" in *Proc. Conf. Design, Autom. Test Europe (EDA)*, 2016, pp. 139–144.

[50] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2668–2682, Nov. 2017.

[51] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. NDSS*, 2015, pp. 1–14.

[52] D. Liu, C. Yu, X. Zhang, and D. Holcomb, "Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits," in *Proc. Conf. Design, Autom. Test Europe (EDA)*, Mar. 2016, pp. 433–438.

[53] S. Keshavarz, C. Paar, and D. Holcomb, "Design automation for obfuscated circuits with multiple viable functions," in *Proc. IEEE Design, Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2017, pp. 886–889.

[54] M. Yasin, A. Sengupta, M. Ashraf, M. Nabeel, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. ACM/SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1601–1618.

[55] Y.-W. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Proc. IEEE 16th Latin-Amer. Test Symp. (LATS)*, Mar. 2015, pp. 1–6.

[56] E. J. McCluskey, "Verification testing—A pseudoexhaustive test technique," *IEEE Trans. Comput.*, vol. C-33, no. 6, pp. 541–546, Jun. 1984.

[57] C. Albrecht. (2005). *IWLS 2005 Benchmarks*. [Online]. Available: http://iwls.org/iwls2005/benchmarks.html

[58] *DC Ultra: Concurrent Timing, Area, Power, and Test Optimization*, Synopsys, Mountain View, CA, USA, 2017.

[59] *VCS: Industry Highest Performance Simulation Solution*, Synopsys, Mountain View, CA, USA, 2017.

[60] R. Kuppuswamy, P. DesRosier, D. Feltham, R. Sheikh, and P. Thadikaran, "Full hold-scan systems in microprocessors: Cost/benefit analysis," *Int. Technol. J.*, vol. 8, no. 1, pp. 63–72, Feb. 2004.

[61] U. Guin, T. Chakraborty, and M. Tehranipoor, "Functional $F_{\max}$ test-time reduction using novel DFTs for circuit initialization," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 1–6.

[62] E. Singh, C. Barrett, and S. Mitra, "E-QED: Electrical bug localization during post-silicon validation enabled by quick error detection and formal methods," in *Proc. Int. Conf. Comput.-Aided Verification*, 2017, pp. 104–125.

[63] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "Complexity of sequential ATPG," in *Proc. IEEE Eur. Design Test Conf. (ED & TC)*, Mar. 1995, pp. 252–261.

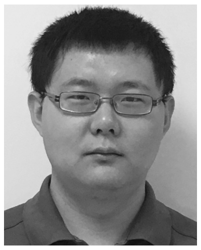
[64] *TetraMAX ATPG: Automatic Test Pattern Generation*, Synopsys, Mountain View, CA, USA, 2017.

[65] *Synopsys 32/28 nm Generic Library for Teaching IC Design*. Accessed: 2017. [Online]. Available: https://www.synopsys.com/community/university-program/teaching-resources.html

Ujjwal Guin (S'10–M'16) received the B.E. degree from the Department of Electronics and Telecommunication Engineering, Bengal Engineering and Science University, Howrah, India, in 2004, the M.S. degree from the Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA, USA, in 2010, and the Ph.D. degree from the Electrical and Computer Engineering Department, University of Connecticut, Storrs, CT, USA, in 2016.

He is currently an Assistant Professor at the Electrical and Computer Engineering Department, Auburn University, Auburn, AL, USA. He has developed several on-chip structures and techniques to improve the security, trustworthiness, and reliability of integrated circuits. He has coauthored the book *Counterfeit Integrated Circuits: Detection and Avoidance*. He has authored several journal articles and refereed conference papers. His current research interests include hardware security and trust, supply chain security, cybersecurity, and VLSI design and test.

Dr. Guin is an active participant in the SAE International's G-19A Test Laboratory Standards Development Committee.



Ziqi Zhou (S'17) received the B.E. degree from the College of Mechanical and Electrical and Engineering, North China University of Technology, Beijing, China, in 2012. He is working toward the Ph.D. degree at the Electrical and Computer Engineering Department, Auburn University, Auburn, AL, USA.

His current research interests include hardware security, VLSI design and test, and computer-aided design of digital systems.

Adit Singh (S'81–M'83–SM'00–F'02) received the B.Tech. degree from IIT Kanpur, Kanpur, India, and the M.S. and Ph.D. degrees from Virginia Tech, Blacksburg, VA, USA, all in electrical engineering.

He was on the faculty at the University of Massachusetts Amherst, Amherst, MA, USA, and the Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, VA, USA. He has also held visiting positions during sabbaticals, most recently as a "Guest Professor" at the University of Freiburg, Freiburg im Breisgau, Germany, in 2012. He is currently a James B. Davis Professor of Electrical and Computer Engineering at Auburn University, Auburn, AL, USA. He has served as a Consultant to many major semiconductor, test, and EDA companies around the world, including as an Expert Witness on patent litigation cases. He has authored over 250 research papers and holds international patents that have been licensed to industry. He is particularly recognized for his pioneering contributions to statistical methods in test and adaptive testing. His current research interests include VLSI technology, in particular integrated circuit test and reliability.

Dr. Singh has had leadership roles as a General Chair/Co-Chair/Program Chair for dozens of international VLSI design and test conferences. He was a Program Co-Chair for the 2014 International Conference on VLSI Design, and a Program Chair of the 2015 Asian Test Symposium. He also served on the editorial boards of several journals, including the IEEE Design and Test, and on the Steering and Program Committees of many of the major IEEE international test and design automation conferences. He served two elected terms as a Chair for the IEEE Test Technology Technical Council (2007–2011), and on the Board of Governors of the IEEE Council on Design Automation (2011–2015).