On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic

Hadi M Kamali, Kimia Z Azar
George Mason University
Fairfax, VA, USA
{hmardani,kzamiria}@gmu.edu

Houman Homayoun
University of California Davis
Davis, CA, USA
hhomayoun@ucdavis.edu

Avesta Sasan
George Mason University
Fairfax, VA, USA
asasan@gmu.edu

ABSTRACT

In this paper, we assess the security and testability of the state-of-the-art design-for-security (DFS) architectures in the presence of scan-chain locking/obfuscation, a group of solution that has previously proposed to restrict unauthorized access to the scan chain. We discuss the key leakage vulnerability in the recently published prior-art DFS architectures. This leakage relies on the potential glitches in the DFS architecture that could lead the adversary to make a leakage condition in the circuit. Also, we demonstrate that the state-of-the-art DFS architectures impose some substantial architectural drawbacks that moderately affect both test flow and design constraints. We propose a new DFS architecture for building a secure scan chain architecture while addressing the potential of key leakage. The proposed architecture allows the designer to perform the structural test with no limitation, enabling an untrusted foundry to utilize the scan chain for manufacturing fault testing without having a need to access the scan chain. Our proposed solution poses negligible limitation/overhead on the test flow, as well as the design criteria.

KEYWORDS

Reverse Engineering, Logic Obfuscation, Scan Chain, Key Leakage

ACM Reference Format:
Hadi M Kamali, Kimia Z Azar, Houman Homayoun, and Avesta Sasan. 2020. On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic. In *Proceedings of the Great Lakes Sympo-sium on VLSI 2020 (GLSVLSI '20), September 7–9, 2020, Virtual Event, China. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386263.3407655*

1 INTRODUCTION

The large cost of building semiconductor fabrication foundries, the ever-increasing density of ICs using smaller technology nodes, and significant maintenance and operation costs of the semiconductor facilities, have forced many high-tech companies to outsource many design stages, including the fabrication [1], making IC supply chain a global one. This financially and economically sensible vertical supply-chain model, however, has raised many security and trust concerns including but not limited to IP piracy, reverse engineering, and IC overproduction [13].

To combat these threats, amongst many countermeasure solutions, logic obfuscation [5] introduces a form of post-manufacturing programming into the design, making the functionality of the circuit dependent to the programming values, referred to as the *key*. After fabrication, when the design house receives the fabricated ICs, the correct key is programmed into a tamper-proof non-volatile memory (tpNVM) [18], reducing the functionality of the IP to the correct functionality. If the inserted key is not the correct key, the obfuscated design implement a different bogus function, generating a different output response (output corruption) to the same input.

In 2015, the introduction of the Boolean satisfiability (SAT) attack [11, 17] challenged the validity/strength of all prior logic obfuscation solutions. The SAT attack was able to easily break logic obfuscation solutions in a matter of seconds to minutes, shattering the false sense of security of all prior-art logic obfuscation solutions. The original SAT attack is applicable to combinational circuits. However, the existence of the much-needed scan chain for functional and structural testing, makes the sequential circuits also vulnerable to this attack when an adversary gains access to the scan chain.

After the introduction of the SAT attack, a wide range of new logic obfuscation solutions have been introduced in the literature to combat/prevent the SAT attack. These countermeasures could be divided into logic obfuscation solutions aiming to (1) formulate and apply a SAT-resilient logic obfuscation solution, and those aiming to (2) restrict unauthorized access to the scan chain.

Regarding the former category, they could be further broken down into (a) logical SAT-hard solutions such as SFLL [15] and Full-lock [2], and (b) behavioural SAT-inapplicable obfuscation techniques, such as cyclic logic locking [6], or Delay Logic Locking (DLL) [24]. Logical SAT-hard solutions aim at increasing either the number of SAT attack iterations [15] or the number of recursive calls in each iteration of the SAT attack to a sufficiently large number [2]. However, the problem with this group is either the extremely low output corruption (in SFLL [15]) or a large area overhead (in Full-lock [2]). On the other hand, the behavioural SAT-inapplicable obfuscation techniques aim to build techniques in a way that cannot be modeled using the SAT attack, such as cycles that traps the SAT solver or delay locking that cannot be modeled by the SAT attack. However, this breed was later broken by SAT inspired attacks on cyclic obfuscation such as CycSAT [3] and satisfiability modulo theory (SMT) [9] attack.

Blocking unauthorized access to the scan chain as the latter category [8, 19, 21–23] limits the access of an adversary only to the primary inputs and primary outputs (PI/PO). Expanding on the SAT attack, it was later shown, that an adversary can still attack a sequential circuit with no access to the scan chain by using unrolling-based SAT attack [12] or a bounded-model-checking (BMC) attack [10]. However, these sequential attacks are far weaker than pure SAT and are mostly applicable to moderately small sequential circuits. Since the sequential attacks are not scalable, by blocking the scan chain, and applying many of the prior logic obfuscation techniques, a moderately size obfuscated netlist could easily resist such attacks.

GLSVLSI '20, September 7–9, 2020, Virtual Event, China

ACM ISBN 978-1-4503-7944-1/20/09...$15.00
https://doi.org/10.1145/3386263.3407655

Table 1: Comparison of the State-of-the-art DFS architectures with our proposed kt-DFS.

Defenses	Test Time	Test Complexity	Resilient against		
			ScanSAT [10]	Shift&leak [16]	Glitch&Leak
EFF + RLL [19]	low	None	✗	✓	✓
R-DFS + SLL [21]	low	None	✓	✗	✓
mR-DFS + SLL [16]	high	key reload per pattern	✓	✓	✗
kt-DFS + SLL	low	None	✓	✓	✓

Prior work on restricting unauthorized access to the scan chain could be divided into (1) scan chain obfuscation [19, 23] and (2) scan chain blocking [21, 22]. In the scan chain obfuscation techniques, such as encrypt flip-flop [19] or dynamically obfuscated scan (DOS) [23], the scan chain is statically or dynamically locked by inserting key gates. However, ScanSAT [10] could break both statically and dynamically scan chain obfuscation techniques by transforming the obfuscated scan chain into a combinational circuit and thereby launching the SAT attack on them (the unrolling-based SAT) [10].

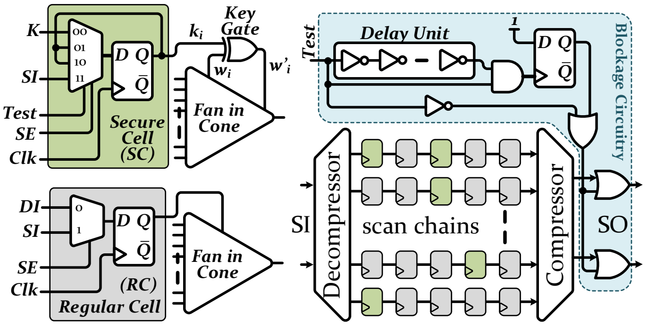
In the scan chain blocking techniques, after loading the obfuscation key (from tpNVM), the access to the scan-out(s) (SO) would be blocked [21]. By eliminating the access to the SO, an adversary's ability to monitor the behavior of the circuit will be limited only to the PO. This eliminates the possibility of the SAT attack as well as any attack that requires access to the scan chain, forcing an attacker to use the far weaker and non-scalable sequential attacks.

Scan chain blocking in the presence of logic obfuscation was first introduced in [21]. In the rest of this paper we refer to this solution as **r**obust **d**esign-**f**or-**s**ecurity (R-DFS). In addition to blocking the SO, the R-DFS also introduces a new storage element for holding the obfuscation key, denoted by secure cell (SC). However, the security of the R-DFS architecture was later challenged by the *shift-and-leak* attack [16]. To remedy the leakage issue, the authors proposed modification to the scan blocking architecture (we call it **mR-DFS**), equipping the SCs with a mode switch shift disable (MSSD) circuitry [16]. The mR-DFS blocks any shift operation after the obfuscation key is loaded from the tpNVM, removing the ability of an adversary to apply the shift-and-leak attack.

In this paper, by showing the architectural drawbacks of mR-DFS, we introduce our proposed DFS scan blocking architecture for protecting the logic obfuscation key. More precisely, the contributions of this work are as follows: (1) We illustrate how a glitch-based shift-and-leak attack allows an adversary to leak the logic obfuscation key even if the shift operation is disabled in mR-DFS, thereby, leaking the actual logic obfuscation key through the PO. (2) As a countermeasure, we propose a new key-trapped design-for-security (**kt-DFS**) architecture, where the scan chain that loads the logic obfuscation key is fully detached from regular scan chain(s). To fulfill this requirement, we propose a new secure cell design content of which cannot be shifted in the scan chain after a key registration event is observed. (3) We assess the security of proposed kt-DFS, and compare the proposed solution with R-DFS and mR-DFS. As shown in Table 1, we will illustrate how the kt-DFS can support both structural and functional testing while resisting all leaky-based and SAT-based attacks on logic locking.

2 BACKGROUND

Both R-DFS [21] and mR-DFS [16] block the SO pins after the obfuscation key is loaded into the design. The mR-DFS is built on top of R-DFS to fix the leakage issue. In the following section, we first describe how R-DFS works. Then, we explain the leakage issue identified in R-DFS, motivating the shift-and-leak attack. Then, we describe how mR-DFS remedies the problem with disabling shift operations after loading the obfuscation key.



(a) (b)

Figure 1: (a) Secure Cell (SC) vs. Regular Cell (RC), (b) Restricted Unauthorized Scan Access using Blockage Circuitry.

Table 2: Modes of Operation in Secure Cell (SC).

Test	SE	Mode	Description
0	0	M_0	The circuit is in functional mode. Actual keys from tpNVM applies to the Logic (Correct Functionality).
0 1	1 0	M_{1a} M_{1b}	The SCs hold their previous value. Based on the value of SE, RCs are in capture/shift mode.
1	1	M_2	The SCs become part of the scan chain. Actual/Dummy keys from SI for structural testing.

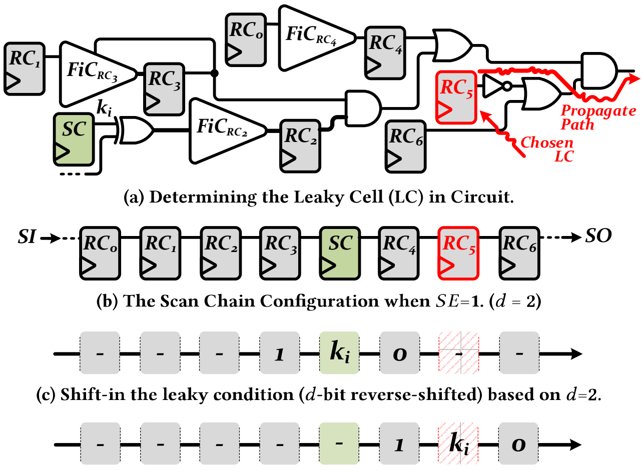
2.1 R-DFS: Restricting Scan Access

In R-DFS [21], the obfuscation key is stored in a custom-designed scan (storage) cell, denoted as secure cell (SC). As shown in Fig. 1(a), in R-DFS, each key value is stored in one SC. The R-DFS architecture, as indicated in Table 2, allows four types/modes of operation based on the *Test* and *SE* pins. The key values could be loaded into SCs either directly from tpNVM (actual key values in mode M_0) or the scan-in (dummy/actual key in mode M_2). The scan chains, as shown in Fig. 1(b), are constructed by stitching the SCs with regular scan Flip-Flops (SFF). The SFFs in this paper are denoted as Regular Cells (RC). The SCs keep their previous values in modes M_{1a} and M_{1b}. The only difference between the M_{1a} and M_{1b} mode is the value of the *SE* pin that determines the shift/capture mode in RCs. Both of the M_{1a} and M_{1b} modes allow the SCs to be bypassed (keeping their previous values) when the RCs are in shift/capture mode.

For the structural (a.k.a manufacturing fault) test, the *Test* pin must be 1, allowing the shift and capture operations to be carried in modes M_2 and M_{1b} respectively, giving unrestricted access to the scan. On the other hand, for a functional test, first, the correct key is loaded from tpNVM into SCs using the mode M_0. Then, the initial state is loaded into the RCs in mode M_{1a}, with no change on the key value in SCs. Finally, the response is observed at the PO in mode M_0. To block unauthorized access to the scan chain (when a valid key is loaded), as illustrated in Fig. 1(b), the R-DFS architecture utilizes a SO-blockage circuitry. This module blocks/masks the SOs upon a switch from functional mode (mode M_0 that loads the actual key into SCs) to test mode (mode M_2 that supports the shift operation). Hence, after loading the key in mode M_0, SO will no longer be accessible, removing the possibility of SAT attack, and limiting the adversary's attack option to the far weaker and non-scalable unrolling-based or BMC based attacks.

2.2 Shift-and-Leak Attack on R-DFS

Although R-DFS breaks the SAT attack by blocking the SO, the introduction of shift-and-leak attack [16] shows that there is a valid key leakage possibility in R-DFS that allows the adversary to observe and extract the logic obfuscation key using PO. This attack exploits (1) the availability of the shift-in process through *SI*, and (2) the capability of reading out the PO through chip pin-outs in



(a) Determining the Leaky Cell (LC) in Circuit.

(b) The Scan Chain Configuration when SE=1. (d = 2)

(c) Shift-in the leaky condition (d-bit reverse-shifted) based on d=2.

(d) shift all FFs, including SCs and RCs, in Mode M_2, to put the k_i into the LC.

Figure 2: Example of shift-and-leak attack on R-DFS.

the functional mode. Considering Fig. 2 as an illustrative example, the steps of a shift-and-leak attack are as follows:

(1) Identify leaky cells (*LC*s) that can leak info onto a PO.
(2) Insert a stuck-at-fault at the chosen *LC* candidate.
(3) Propagate the fault onto a PO (SCs set to unknown $X's$). If it fails to propagate, it rules out this *LC*, and repeats steps 1 and 2.
(4) Power up the chip in mode M_0 to load the correct key into *SC*s.
(5) Switch to mode M_{1a} (*SC*s hold value) and shift in d-bit reverse-shifted of the leak condition into the scan. The value of d is the scan distance between the targeted *SC* and the chosen *LC*.
(6) Switch to mode M_2 (*SC*s are in the scan), and perform d-bit shift to have the leak condition in place and the key in chosen *LC*.
(7) Clocklessly switch to mode M_0 and observe the PO, to leak the content of the *LC*, i.e., the target key bit.

The authors noted that when the number of *SC*s increases, ATPG may fail to find a leak condition for the chosen *LC*. To address this challenge, by exploiting the conventional SAT attack [17], a preprocessing step was added to the shift-and-leak attack, in which the logic cone was treated as a locked combinational circuit considering *RC*s as the primary inputs and *SC*s as the key inputs. The preprocessing phase (which resembles the steps of the conventional SAT attack) is launched as follows:

(1) Extract the combinational fan-in cones of the PO.
(2) Obtain a Discriminating Input (DIP) from the SAT tool on the extracted circuit.
(3) Power on the IC in mode M_0 (*SC*s capture the actual key).
(4) Switch to M_{1a} (*SC*s hold their values), and shift in the obtained DIP from the SAT tool to the *RC*s.
(5) Clocklessly switch to mode M_0 and observe the PO (*eval* of the SAT attack). Then, go to step 2 until no more DIP found.

2.3 mR-DFS: Resisting Shift-and-Leak

As a countermeasure to the shift-and-leak attack, the work in [16] proposes a modified version of robust design-for-security architecture (denoted as mR-DFS in this paper) with a slight modification to the R-DFS. Since mode M_{1a} is used in the shift-and-leak attack to shift-in the known patterns (leak condition or DIP) to *RC*s, in mR-DFS, this mode is blocked. Also, to avoid any other form of leakage, after switching to mode M_0, it is not possible to re-enable any shift mode in the scan chain. To do that, as shown in Fig. 3, they build a shift disable (*SD*) signal, such that when $Test$ = 1, *SD* follows *SE*. But, after the first capture of the actual key, i.e. when the $Test$ is low or when there is a positive transition on the $Test$, *SD* becomes *ALWAYS ZERO*, thereby blocking the shift operation. Hence, there is no longer a mode where *SC*s can be bypassed, retaining their values, while *RC*s can be loaded/shifted.

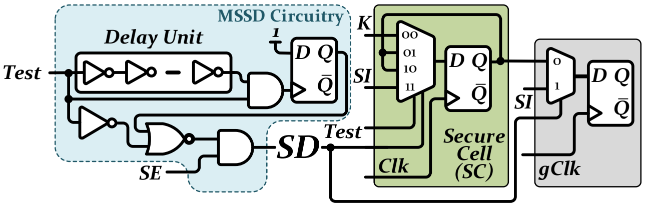


Figure 3: Mode Switch Shift Disable (MSSD) in mR-DFS.

3 mR-DFS ARCHITECTURAL DRAWBACKS

Although mR-DFS addresses the leakage problem in R-DFS using shift disable (*SD*), the introduction of this shift disable (*SD*) signal in mR-DFS poses some new challenges for design and implementation flow, as well as test and debug process. These challenges are discussed next:

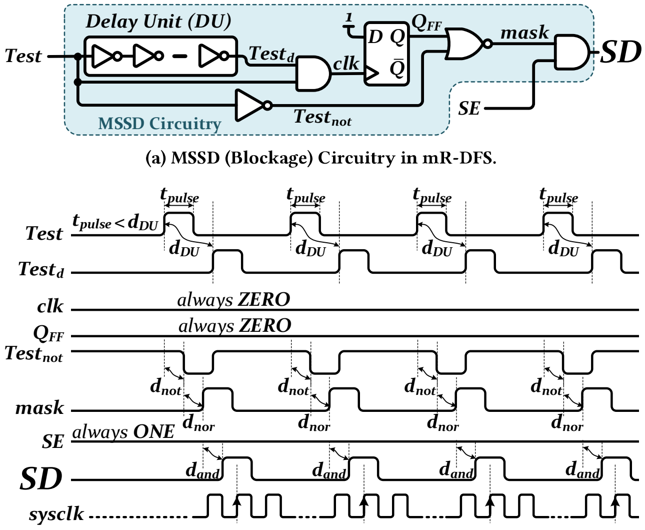
3.1 High Functional Test Time

Since there is no longer mode M_{1a} in mR-DFS architecture, the tester has to rely on mode M_2 to shift in and load the RCs. Also, since the shift is disabled when $Test$=0 or after the first positive transition on the $Test$, $Test$ must be high during power ON. Hence, the tester should use M_2 as the initial mode to shift in and load the initial state into the RCs. After loading the initial state, the tester switches the mode to M_0 to load the actual key. Since it is not possible to re-enable the shift process after switching to mode M_0, the tester has to rely on the responses on PO. For the next test pattern, the tester needs to switch back to the mode M_2 to shift in and re-load the initial state corresponded to the next pattern. However, due to the blockage of the shift operation after switching to mode M_0 ($Test$ = 0), the tester cannot use shift-in anymore for shifting in the next initial state. Hence, the tester has to reset the FF of MSSD circuitry to re-enable shift-in. This reset re-enables the *SD* to follow *SE*, thereby, the tester can shift in the next initial state. However this reset (*sys_rst*) will clear all storage elements, including SCs. So, it forces the tester to re-load the keys for the next test pattern. Hence, the actual key must be loaded again from tpNVM to accomplish the functional test, and this key reloading process (with each test pattern) significantly increases the functional test time. It should be noted that the initial state could be chosen to be used for a group of test patterns; however, choosing a specific initial state to be used for a group of patterns would increase the complexity of the functional test significantly. Besides, the designer cannot separate the reset pins for MSSD. Assuming that this reset pin is separated, the adversary can engage it to re-enable shift operation while the actual key is in place.

3.2 Necessity of Duplicating the SCs

In mR-DFS, after shifting in the initial state to the RCs using mode M_2, the tester switches to mode M_0 for only one cycle to load the actual key. However, during this one cycle, the RCs (loaded by initial state) would be updated. To avoid this problem, a clock gating circuitry has been introduced in mR-DFS to disable the clock for one clock cycle after switching from M_2 to M_0.

Without any consideration for this requirement in mR-DFS architecture, there are two possible methods to load the actual key from tpNVM in one clock cycle, however, both of them incurs considerable performance/area overhead: (1) engaging an ultra-wide memory that provides all bits of logic obfuscation key at once using only one read operation, (2) engaging temporary registers (FFs) to



(a) MSSD (Blockage) Circuitry in mR-DFS.

(b) Glitches in Post-Synthesis Timing Simulation of MSSD Circuitry.

Figure 4: Re-enabling Shift after Actual Key Load.

load the key into them at power ON, then connecting each SC to its corresponding temporary register to be loaded in one clock cycle.

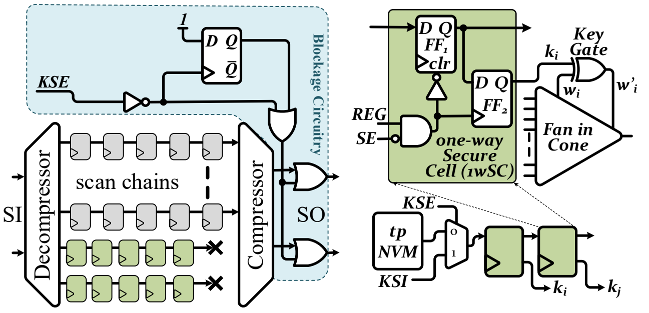
Regarding the former solution, it is required to have direct wiring from tpNVM to each SC (per each key gate). Hence, the ultra-wide memory must have an extremely high fanout to provide this direct connection. This ultra-high fanout wiring increases the complexity of placement and routing (PnR) process, and it would significantly decrease the performance of the design, and due to optimization constraints in each design, using this scheme is almost impractical.

By choosing the latter method, the incurred overhead is more reasonable. However, the required reset (*sys_rst*) for loading the next initial state will clear whole registers in the chip. So, a key re-loading from tpNVM to the temporary register is required for each (group of) test pattern. It raises two big problems in mR-DFS: (1) It significantly increases the required time for functional test, and (2) Since key re-loading takes more than one clock cycle, it violates the assumption of mR-DFS, where clock-gating disables the clock signal only for one clock cycle to preserve the value of the RCs. So, after only one clock cycle, during the key re-loading, the RCs would be updated, and the functional test will fail.

3.3 Re-enabling Shift using Leaky Glitches

In mR-DFS, as shown in Fig. 3, the selector of $MUX21$ in RCs is controlled by *SD*, which becomes *ALWAYS ZERO* immediately after the first attempt of switching back from mode M_0 to M_2 (re-enabling shift process). Switching from mode M_0 to M_2 means that there is a positive transition on the $Test$ pin, and this positive transition allows the FF in MSSD circuitry to capture its input (*CONSTANT ONE*). However, there is still a possibility to switch back from mode M_0 to M_2 (positive transition on the $Test$ pin) while the FF does not capture its input (*CONSTANT ONE*) to make *SD* to be *ALWAYS ZERO*. To show that, we draw a timing diagram of the post-synthesis timing simulation of all internal wires of MSSD circuitry.

As shown in Fig. 4(a), a delay unit (*DU*) has been used as a part of the fan-in-cone of the FF in MSSD circuitry, which is built using 10 inverters [16]. Assuming that the adversary is aware of timing information of the circuit, as shown in Fig. 4(b), she generates a stimuli for $Test$ pin in which the duration of high pulses is less than the delay of *DU* ($t_{pulse} < d_{DU}$). Hence, the inputs of the first *AND* gate, i.e. $Test$ and $Test_d$, have no overlap when both signals are high, and accordingly, DFF's *clk* would be *ALWAYS ZERO*. Since it is assumed that the DFF sets to 0 on reset, Q_{FF} would also be *ALWAYS ZERO*. So, the function of *NOR* gate is similar to *NOT* gate, whose input is $Test_{not}$. Consequently, $mask$ follows $Test$ with a delay of $d_{not} + d_{nor}$, and similarly, if we suppose that *SE* is *ALWAYS ONE*, *SD* follows $Test$ with a delay of $d_{not} + d_{nor} + d_{and}$. Since *SD* controls the shift operation in mR-DFS, using these potential glitches, the *SD* can re-enable the shift operation after mode M_0.



(a) Decoupling SCs from RCs

(b) One-way Secure Cell (1wSC)

Figure 5: (a) kt-DFS Architecture with New Re-designed Blockage Circuitry, (b) Using 1wSC for Logic Locking Key. *KSE* **determines the source of the key (tpNVM or** *KSI***).**

4 PROPOSED SOLUTION

When the logic obfuscation is in place, to introduce a secure and robust scan chain architecture, three requirements must be met:

(1) There must be no possibility of key leakage during the test.
(2) Both structural test and functional test must be carried out in a reasonable time (low test time overhead compared to the test time of the original design) without significant loss of coverage.
(3) The complexity of test flow (structural and functional) and the overhead of secure scan chain architecture must be minimized.

In our proposed key-trapped DFS (kt-DFS), the scan chain(s) of the SCs are completely decoupled from the scan chain(s) of the RCs. In fact, **there is no reason for stitching the RC and SC cells in one chain, which has been the source of vulnerability in both R-DFS and mR-DFS**. As illustrated in Fig. 5(a), there is no common path between RCs and SCs in our proposed kt-DFS architecture. Also, considering that the SCs are only responsible to store the key value, none of the internal operations/computations overwrites the content of the SCs. So, when the scan chain is in place for the SCs, only the shift-in through SI is available for them to load the keys, and the SO is permanently blocked for scan chain(s) of the SCs.

To guarantee the security of SCs against any form of leakage, we re-design and introduce a new secure cell, called 1-way secure cell (1wSC). Fig. 5(b) depicts the details of 1wSC. Each 1wSC has two internal storage elements: a scan-connected storage (denoted as FF_1), and a trap storage (denoted as FF_2). The scan-connected storage could be used to shift values in and out of the 1wSC or into the trap storage. However, the value of the trap storage cannot be shifted out, and is only connected to its corresponded key gate. The transfer of key value from FF_1 to FF_2 takes place after setting $REG = 1$ and $SE = 0$, which is called *register mode*. Registration of the key into trap storage takes place on the rising edge of the clock input of the FF_2, which is a function of *REG* and *SE*. Also, this condition is used as the *RESET* condition of all FF_1s to clear their values. Hence, $AND(Test, \overline{SE})$ is used as the clock source of FF_2s, and its toggled is used as the *RESET* for FF_1s.

Also, the trap storage does not have a reset, and upon power-up randomly initialized to 0 or 1. So, upon transition of the key from

Table 3: Modes of Operation in kt-DFS.

REG	SE	Mode	Description
0	0	M_0 (Functional Mode)	FF_2 must have the key*. FF_1 could capture the key*.
0	1	M_1 (Shift Mode)	FF_1 could capture the key*.
1	0	M_2 (Register Mode)	FF_2 are fed from FF_1. FF_1 will be reset to *ZERO*. Chain is erased.
1	1	M_3 (Shift Mode)	FF_1 could capture the key*.

* Based on KSE, actual/dummy key could be loaded from tpNVM/KSI

scan-connected storage to the trap storage, since the storage is initialized randomly, the adversary cannot determine the previous value of the trap storage. This prevents the back-side imaging attack based on the captured heat map as described in [14] (e.g. when the activity is observed on heat map for a specific storage element, the adversary cannot determine if the transition is {0 → 1} or {1 → 0}, and if *NO* activity is observed, the adversary cannot determine if the transition is {0 → 1} or {1 → 1}).

In our proposed kt-DFS, the keys could be loaded into 1wSC from either tpNVM or scan-in (SI). Hence, the tester would be able to carry out the structural test by loading the desired key using SI. But, since the scan chain(s) of 1wSCs are decoupled in kt-DFS, two dedicated scan-enable and scan-in are used for the scan chain(s) of the SCs, called *KSE* and *KSI* respectively.

The behavior of 1wSC is controlled using two pins, here called *REG* and *SE*. As captured in Table 3, based on these two pins, a 1wSC can be operated in three main modes:

(1) **Functional Mode (M_0)**: {*REG*, *SE*}={0,0}, and the RCs are in capture mode. Trap storage (FF$_2$s) must have the key. However, scan-connected storage (FF$_1$s) is able to capture a new key.
(2) **Shift Mode ($M_{1,3}$)**: {*REG*, *SE*}={*X*,1}, and the RCs are in shift mode. Scan-connected storage (FF$_1$s) is able to capture the key simultaneously, and there is no action on trap storage (FF$_2$s).
(3) **Register Mode (M_2)**: {*REG*, *SE*}={1,0}, and the pre-loaded key in scan-connected storage (FF$_1$s) would be written to trap storage (FF$_2$s), and scan-connected storage (FF$_1$s) will be cleared.

Similar to R-DFS and mR-DFS, a blockage circuitry is required to block the SO after the first attempt of key loading from the tpNVM. To support our proposed operational modes in the kt-DFS, a new blockage circuitry is designed. In kt-DFS, the SO must be blocked after loading the actual keys into FF_2s. When KSE is low, the FF_1 is fed using tpNVM. Hence, $\overline{KSE}$ is used to mask the SO. Note that the actual key would be loaded into FF_2 when REG = 1 and SE = 0 (register mode). However, before this condition, the tester has to load the actual key into FF_1s while the KSE is low. Hence, by only considering KSE = 0 as the blocking condition, we also cover the register-mode. Accordingly, the SO would be no longer available when KSE becomes low.

4.1 No Possibility of Key Leakage in kt-DFS

Considering that the leakage problem in R-DFS and mR-DFS is for unnecessary stitching of the RC and SC in the same scan chain, we fully decouple the SCs and RCs scan chains in kt-DFS, and the output of the scan chain(s) of SCs is permanently blocked. The values stored in the scan-connected storage (FF_1s) will be cleared with the transfer of the key to the trap storage (FF_2s). This guarantees that key values are trapped and no either regular or glitch-based shift can leak the key values to the SO.

4.2 Functional/Structural Test in kt-DFS

In kt-DFS architecture, the functional test and the structural test could be done without any significant limitation or any substantial overhead. For the structural test, since the SCs are equipped with new *KSE* and *KSI* pins, it could be accomplished using the following steps:

(1) Set $KSE \longrightarrow 1$ and mode to M_0. Shift in a dummy key via *KSI*.
(2) Switch to mode M_2 to write the key into FF_2, and to clear FF_1.
(3) Switch to mode M_1 to shift in the initial state into RCs.
(4) Switch to mode M_0 for one clock cycle for capturing new state.
(5) Switch again to mode M_1 to shift out the RCs to SO.

Unlike the structural test, the functional test requires the actual key. Hence, loading the key from tpNVM followed by register mode will block the SO. Considering the blockage of the SO, the steps of the functional test is as follows:

(1) Set $KSE \longrightarrow 0$ and mode to M_0. Shift in the actual key from tpNVM. (When KSE = 0, the SO is blocked.)
(2) Switch to mode M_2 to write the key into FF_2, and to clear FF_1. (Once KSE = 0 and mode is M_2, the SO will no longer available.)
(3) Switch to mode M_1 to shift in the initial state into the RCs.
(4) Switch to mode M_0 for one clock cycle for capturing new state, and clocklessly observe the PO.

It should be noted that similar to R-DFS and mR-DFS, the tester accomplishes the functional test through observing the PO with negligible loss of coverage.

4.3 Test Complexity and Scan Chain Overhead

Decoupling the scan chain(s) of SCs from that of RCs helps to facilitate the test flow for the tester compared to the test flow in mR-DFS. Despite mR-DFS with a mandatory *sys_rst* for each (group of) test pattern, no additional operation is required in kt-DFS for any form of the test. No *sys_rst* is required, and none of the operations is blocked after the first attempt of the actual key loading from tpNVM, and similar to R-DFS, only the SO is blocked to break the SAT attack. However, unlike R-DFS, it is fully secure against any form of leakage-based attacks, such as shift-and-leak.

The 1wSC in our proposed kt-DFS has two storage units and has a larger footprint compared to the SCs used in R-DFS and mR-DFS. However, the R-DFS and mR-DFS also need to transfer the key values from tpNVM to SCs. Using a very wide memory to derive thousands of keys is quite demanding in terms of area, and it imposes higher complexity during PnR. Hence, the R-DFS and mR-DFS also need to resort to a chain of temporary registers to transfer the keys. This means there is also a duplicated register per each SC in both R-DFS and mR-DFS. Furthermore, compared to *MUX*41 in both R-DFS and mR-DFS, only one *AND* gate and one *NOT* have been used in each 1wSC, which slightly improves the area overhead.

5 EXPERIMENTAL RESULT

To analyze the security of the kt-DFS, and to provide better comparative results, we engage the same ITC-99 and ISCAS-89 benchmark circuits as used in mR-DFS [16]. We engaged strong logic locking (SLL) [4] in all experiments to determine the location of key gates, and the number of key bits is 128. All the experiments have been accomplished on a Dell PowerEdge R620 equipped with Intel Xeon E5-2670 2.50GHz and 64GB of RAM, using Synopsys Design Compiler 2017.09, Tetramax 2017.09, and VCS 2017.12 tools along with the Synopsys generic 32nm library.

Table 4 represents the area overhead, test coverage, and the leakage of R-DFS, mR-DFS, and our proposed kt-DFS. The number

Table 4: Area Overhead, Test Coverage, and Leakage Comparison between R-DFS, mR-DFS using MSSD, and proposed kt-DFS for Identical Timing Constraints. (Key Size = 128, Number of Scan Chain = 1)

Benchmark	Original	R-DFS [21]			mR-DFS using MSSD [16]			Proposed kt-DFS		
	Test Coverage (%)	Area Overhead (%)	Test Coverage (%)	Key Recovered (#)	Area Overhead (%)	Test Coverage (%)	Key Recovered (#)	Area Overhead (%)	Test Coverage (%)	Key Recovered (#)
s35932	100	12.49%	100	127	8.16%	100	127	5.21%	100	0
s38417	100	13.84%	100	128	9.31%	100	128	5.91%	100	0
s38584	100	14.85%	100	128	11.17%	100	128	6.27%	100	0
b17	99.91	6.24%	99.72	127	4.79%	99.69	127	1.84%	99.67	0
b18	99.77	3.08%	99.78	126	1.75%	99.77	126	0.55%	99.73	0
b19	99.8	1.17%	99.78	127	0.67%	99.79	127	0.24%	99.78	0

of (regular) scan chains (composed of RCs) is set to be one. This is because with a large number of scan chains, the length of each scan chain will be short. Hence, the chosen LCs can leak the content of a smaller number of SCs from their shorter scan chains, which decreases the success rate of the shift-and-leak attack. Hence, we assume that there is only one scan chain in the circuit to make it the best-case scenario for the shift-and-leak.

Considering that the access to the scan chain is restricted, the SAT attack cannot be deployed. This does not prevent an attacker from deploying the unrolling or bounded-model-checking (BMC) [12] attack that only relies on PI/PO. However, this group of attacks runs into scalability issues as they rely on two sub-routines which are in PSPACE and NP [16]. Even the accelerated version of this attack (described in [7]) fails to terminate for even small designs. Besides, new techniques such as DFSSD [20] shows how low overhead techniques, like deep faults and shallow state duality, could be used to break the state-of-the-art sequential SAT attacks. Hence, Table 4 only reflects the effectiveness of the shift-and-leak attack.

To evaluate the possibility of the leakage in R-DFS, we engage the shift-and-leak attack in [16] with no change. However, for mR-DFS, we integrated the glitch propagation that described in Section 3.3 with the shift-and-leak attack. Although the mR-DFS is resilient against the original shift-and-leak attack, assuming the potential glitches in MSSD, as well as controlling the *sys_clk* using an external clock generator, the security of mR-DFS and R-DFS is at the same level, and both could be broken by leaking the key bits onto PO. However, there is no leakage possibility in the proposed kt-DFS, and the adversary cannot recover the content of any SC.

Regarding the area overhead, we assume that both R-DFS and mR-DFS use a temporary registers to transfer the key values from tpNVM to SCs (this has lower overhead compared to using a very wide memory). Hence, as a part of the logic obfuscation circuit, these temporary registers affect the area overhead. Note that in our proposed kt-DFS, these temporary registers are part of 1wSC. Also, compared to SCs in R-DFS and mR-DFS less basic gates are used in kt-DFS. Overall, compared to R-DFS and mR-DFS, kt-DFS reduces the area overhead by 61% and 44%, respectively.

As discussed previously, all three schemes block the SO after the first attempt of key loading from tpNVM. Hence, all have to rely on the PO for the functional test. As shown in Table 4, the test coverage loss is negligible and almost identical in all schemes.

6 CONCLUSION

In this paper, we first evaluated the information/key leakage possibility and design methodology drawbacks of recently published DFS architectures in the presence of scan chain locking. Then, we proposed a new obfuscated DFS solution, denoted as key-trapped DFS (kt-DFS) that addresses the prior art shortcomings. In kt-DFS, we introduced a new secure storage cell for the storage of key values. The proposed secure cell allows us to trap the key after being loaded, preventing different forms of shift and leak attacks (glitch based or logic-based), while safely removing the key upon transition from functional to test mode. At the same time, we illustrated that using the proposed DFS, the design can safely undergo manufacturing and functional testing without incurring any significant limitation in terms of increase in the test time (functional or manufacturing) while maintaining desirably low overhead.

7 ACKNOWLEDGEMENT

This research is supported by the Defense Advanced Research Projects Agency (DARPA-AFRL, #FA8650-18-1-7819), National Science Foundation (NSF, #1718434), and partly by Silicon Research Co. (SRC TaskID 2772.001).

REFERENCES

[1] A. Yeh. 2012. Trends in the Global IC Design Service Market. *DIGITIMES* (2012).

[2] H. M. Kamali *et al.* 2019. Full-lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks. In *Design Automation Conf. (DAC)*. 1–6.

[3] H. Zhou *et al.* 2017. CycSAT: SAT-based Attack on Cyclic Logic Encryptions. In *Int'l Conf. on CAD (ICCAD)*. IEEE, 49–56.

[4] J. Rajendran *et al.* 2012. Security Analysis of Logic Obfuscation. In *Design Automation Conf. (DAC)*. 83–89.

[5] J. Roy *et al.* 2008. EPIC: Ending Piracy of Integrated Circuits. In *Design, Automation & Test in Europe Conf. (DATE)*. 1069–1074.

[6] K. Shamsi *et al.* 2017. Cyclic Obfuscation for Creating SAT-unresolvable Circuits. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*. 173–178.

[7] K. Shamsi *et al.* 2019. KC2: Key-condition Crunching for Sequential Circuit Deobfuscation. In *Design, Automation & Test in Europe Conf. (DATE)*. 534–539.

[8] K. Z. Azar *et al.* 2019. COMA: Communication and Obfuscation Management Architecture. In *Int'l Symp. on Research in Attacks, Intrusions and Defenses (RAID)*. 181–195.

[9] K. Z. Azar *et al.* 2019. SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks. *IACR Trans. on Cryptographic Hardware and Embedded Systems (CHES)* (2019), 97–122.

[10] L. Alrahis *et al.* 2019. ScanSAT: Unlocking Obfuscated Scan Chains. In *Asia and South Pacific Design Automation Conf. (ASP-DAC)*. 352–357.

[11] M. El Massad *et al.* 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.. In *NDSS*. 1–14.

[12] M. El Massad *et al.* 2017. Reverse Engineering Camouflaged Sequential Circuits without Scan Access. In *Int'l Conf. on CAD (ICCAD)*. 33–40.

[13] M. Rostami *et al.* 2014. A Primer on Hardware Security: Models, Methods, and Metrics. *Proc. IEEE* 102, 8 (2014), 1283–1295.

[14] M. T. Rahman *et al.* 2018. The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes. *Int'l Symp. on Hardware Oriented Security and Trust (HOST)* (2018), 1–10.

[15] M. Yasin *et al.* 2017. Provably-Secure Logic Locking: From Theory to Practice. In *ACM SIGSAC Conf. on CCS*. 1601–1618.

[16] N. Limaye *et al.* 2019. Is Robust Design-for-Security Robust Enough? Attack on Locked Circuits with Restricted Scan Chain Access. *Int'l Conf. on CAD (ICCAD)* (2019), 1–8.

[17] P. Subramanyan, *et al.* 2015. Evaluating the Security of Logic Encryption Algorithms. In *Int'l Symp. on Hardware Oriented Security and Trust (HOST)*. 137–143.

[18] P. Tuyls *et al.* 2006. Read-proof Hardware from Protective Coatings. In *Int'l Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. 369–383.

[19] R. Karmakar *et al.* 2018. Encrypt flip-flop: A Novel Logic Encryption Technique for Sequential Circuits. *arXiv preprint arXiv:1801.04961* (2018).

[20] S. Roshanisefat *et al.* 2020. DFSSD: Deep Faults and Shallow State Duality, A Provably Strong Obfuscation Solution for Circuits with Restricted Access to Scan Chain. In *VLSI Test Symposium (VTS)*. 1–6.

[21] U. Guin *et al.* 2018. Robust Design-for-Security Architecture for enabling Trust in IC Manufacturing and Test. *IEEE Trans. on VLSI* 26, 5 (2018), 818–830.

[22] W. Wang *et al.* 2019. A Secure DFT Architecture Protecting Crypto Chips Against Scan-Based Attacks. *IEEE Access* 7 (2019), 22206–22213.

[23] X. Wang *et al.* 2017. Secure Scan and Test using Obfuscation throughout Supply Chain. *IEEE Trans. on CAD* 37, 9 (2017), 1867–1880.

[24] Y. Xie, and A. Srivastava. 2017. Delay Locking: Security Enhancement of Logic Locking against IC Counterfeiting and Overproduction. In *Design Automation Conf. (DAC)*. 9.