Evaluating the Security of eFPGA-based Redaction Algorithms

Amin Rezaei
California State University Long Beach
Long Beach, California, USA

Raheel Afsharmazayejani
University of Calgary
Calgary, Alberta, Canada

Jordan Maynard
California State University Long Beach
Long Beach, California, USA

ABSTRACT

Hardware IP owners must envision procedures to avoid piracy and overproduction of their designs under a fabless paradigm. A newly proposed technique to obfuscate critical components in a logic design is called eFPGA-based redaction, which replaces a sensitive sub-circuit with an embedded FPGA, and the eFPGA is configured to perform the same functionality as the missing sub-circuit. In this case, the configuration bitstream acts as a hidden key only known to the hardware IP owner. In this paper, we first evaluate the security promise of the existing eFPGA-based redaction algorithms as a preliminary study. Then, we break eFPGA-based redaction schemes by an initial but not necessarily efficient attack named *DIP Exclusion* that excludes problematic input patterns from checking in a brute-force manner. Finally, by combining cycle breaking and unrolling, we propose a novel and powerful attack called *Break & Unroll* that is able to recover the bitstream of state-of-the-art eFPGA-based redaction schemes in a relatively short time even with the existence of hard cycles and large size keys. This study reveals that the common perception that eFPGA-based redaction is by default secure against oracle-guided attacks, is prejudice. It also shows that additional research on how to systematically create an exponential number of non-combinational hard cycles is required to secure eFPGA-based redaction schemes.

KEYWORDS

Logic Locking; Logic Obfuscation; eFPGA-based Redaction; SAT Attack; Hard Cycles; Cycle Unrolling; Cycle Breaking

ACM Reference Format:
Amin Rezaei, Raheel Afsharmazayejani, and Jordan Maynard. 2022. Evaluating the Security of eFPGA-based Redaction Algorithms. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22), October 30-November 3, 2022, San Diego, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3508352.3549425

1 INTRODUCTION

Most leading-edge design houses have outsourced their fabrication to offshore foundries for the sake of lower labor and manufacturing costs. However, few of these foundries can be trusted due to a lack of universal hardware Intellectual Property (IP) protection laws. Additionally, powerful reverse engineering tools exist which can efficiently extract and duplicate the circuit netlist [1]. Traditional hardware obfuscation schemes [7–9] that lock the functionality of a logic design with the help of extra key input were found to be vulnerable to the SAT attack [10], which can report the correct key in a short time with the help of an activated IC. This led to the development of several SAT-resilient obfuscation schemes [11–39] and stronger attacks [40–56].

Field Programmable Gate Arrays (FPGAs), which are made up of Customizable Logic Blocks (CLBs) that contain Look-Up Tables (LUTs), Flip-Flops (FFs), and routing logic, may be programmed to perform arbitrary functions [4]. A newly proposed technique to obfuscate critical components in a logic design with the help of an embedded FPGA is called eFPGA-based redaction [2–6]. Specifically, a sensitive sub-circuit is replaced with an eFPGA, and the eFPGA is programmed to perform the same functionality as the missing sub-circuit. In this case, the eFPGA's configuration bitstream acts as a hidden key only known to the designer. By default, eFPGA fabrics contain numerous cycles due to their flexible interconnect network, which may make the original SAT attack inapplicable [3]. So, there is a conception that hardware redaction using eFPGAs is SAT resilient. However, in Section 3, we refute this conception by showing that stronger variations of the SAT attack, such as CycSAT [40] and IcySAT [41], allow the attack to be successful in adhoc eFPGA-based redaction schemes by introducing certain constraints to the attack formulation. Thus, we argue that provisions are required to create hard cycles that cannot be identified by these attacks.

After investigating the necessary assumptions to make existing cyclic attacks [40, 41] unsuccessful on eFPGA-based redaction, we propose two novel attacks that can recover the bitstream of the eFPGA fabrics even with the existence of hard cycles and large size keys. The main contributions of this paper are as follows:

- Refuting the conception that eFPGA-based redaction schemes are by default secure against SAT attacks.
- Breaking the security of eFPGA-based redaction schemes by a preliminary *DIP Exclusion* attack and an efficient *Break & Unroll* attack.
- Presenting success of our attacks on finding the bitstream of diverse eFPGA fabrics with hard cycles and large key sizes.

The same as the state-of-the-art SAT attacks [10, 40, 41, 43–45], we assume that the attacker has full access to the obfuscated netlist. Moreover, he/she can acquire a functioning circuit from the market as a black box and get the correct outputs for given input vectors. Also, since almost all ICs are sequential circuits, we assume that the scan chain is accessible to the attacker.

2 RELATED WORKS

Random logic obfuscation methods [7–9] are distinguished by adding key gates to random signals in the original circuit. As fanins of these additional gates, key inputs are introduced to the circuit. When a wrong key is inserted, the key gates in the circuit become faulty, affecting the primary outputs. In order to make the

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA

ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549425

obfuscated circuit usable, the correct key needs to be inserted in a tamper-proof memory [7] just after the manufactured ICs return from the foundry.

However, an attacker can acquire a functioning circuit from the market and utilize it as a black box to get the correct outputs for given input vectors. In addition, he/she can have access to the netlist of the obfuscated circuit using powerful reverse engineering tools [1] or simply from the untrusted foundry that manufactured the IC. Then, he/she can apply the SAT attack [10] to efficiently find out the protected secret key. The SAT attack [10] utilizes two copies of the obfuscated circuit with the same input, but different key values under a given constraint to check whether it is still possible to generate different outputs. Such input patterns are called Differentiating Input Patterns (DIPs.) Each DIP is then used to query the functioning circuit to get the correct output. Then, the DIP with the output is used to further constrain the keys under consideration. After a few iterations, only the correct key will remain, and thus be reported by the attack program.

Immediately after proposing the SAT attack, different defensive mechanisms [11, 12] were proposed using point functions to increase the required number of DIPs exponentially with the key size. However, these techniques have two main drawbacks. First, they have shown to be vulnerable to approximate SAT attacks [43–45] that can return an almost correct key in which only a small number of input combinations produce wrong outputs. Furthermore, even their most powerful variant [13] is vulnerable to structural attacks [47, 48] that use structural analysis to extract the original circuit from the obfuscated one.

To address the aforementioned drawbacks, critical components of a logic design can be mapped to an eFPGA. Commercial eFPGAs are usually built on typical 4-1 LUTs and may include a variety of hard macros such as integrated Digital Signal Processor (DSP) modules and memories. In [2], an approach for extracting a piece of a design to be mapped onto the eFPGA is described, with a limited area and latency overhead. The authors of [3] demonstrated that eFPGA-based redaction may be used to protect a design without affecting the Application-Specific Integrated Circuit (ASIC) design flows by redacting a RISC-V control route and a precision GPS code generator. The authors of [4] gave insights into the usage of eFPGAs for redacting Register Transfer Level (RTL) designs via a case study of redacting various hand-crafted hierarchical RTL IPs, where distinct modules constitute candidate units for redaction. Finally, the authors of [6] created a bespoke tool that obfuscates a logic design and converts it into an eFPGA device.

Although eFPGA-based redaction seems to temporarily solve the shortcomings of existing logic obfuscation methods, it is only meaningful when a small-sized sub-circuit is replaced with an eFPGA. The same performance as the original design cannot be guaranteed if large parts of an ASIC design are replaced with eFPGA. In Section 3, we put a spotlight on the security promise of these schemes.

3 PRELIMINARY STUDY

There is a common impression that eFPGA-based redaction [2–6] is by default secure against oracle guided attacks [10, 40, 41] due to the complex structure of the embedded FPGA. To evaluate the correctness of this impression, we ran two of the existing cyclic SAT attacks, CycSAT [40] and IcySAT [41] on the eFPGA-based redaction benchmarks in [5] including different eFPGA fabrics with varying complexity and bitstream sizes. Specifically, these benchmarks have varying input sizes of LUTs (i.e., K) and different numbers of basic logic elements in a CLB (i.e., N.) As shown in the results from Table 1, we were able to find the bitstream of all the 28 benchmarks. The first observation is that **it is a prejudice to assume that the eFPGA-based redaction is by default secure against the oracle-guided attacks.**

CycSAT is able to recover the bitstream in significantly less time than IcySAT. This is due to the nature of the algorithms used in each attack. The approach of IcySAT is to unroll the circuit, making it combinational and allowing the original SAT algorithm to better observe the behavior of each cycle. Converting the original circuit to an unrolled circuit is costly with respect to time, giving IcySAT exponential run-time compared to key size and number of cycles in the circuit. CycSAT assumes there is at least one key pattern which creates an acyclic circuit, and the strategy is to break the cycles in the circuit by finding key values which satisfy an acyclic condition. This means that the algorithm searches for a key which blocks the propagation of cyclic signals throughout the circuit, creating an acyclic circuit without modifying the obfuscated circuit. Based on the results from Table 1, CycSAT has a faster and more capable

Table 1: Attacks on eFPGA-based redaction benchmarks [5]

Bench	Key Size	IcySAT[41]	CycSAT [40]
K3N2	601	155s, Correct Key	3.79s, Correct Key
K3N3	725	343s, Correct Key	5.79s, Correct Key
K3N4	837	798s, Correct Key	9.48s, Correct Key
K3N5	941	6 hours, No result	9.15s, Correct Key
K3N6	1154	6 hours, No result	19.1s, Correct Key
K3N7	1243	6 hours, No result	25.32s, Correct Key
K3N8	1393	6 hours, No result	29.24s, Correct Key
K4N2	639	195s, Correct Key	3.12s, Correct Key
K4N3	810	3178s, Correct Key	5.47s, UNSAT
K4N4	1049	6 hours, No result	7.22s, Correct Key
K4N5	1316	6 hours, No result	21.28s, Correct Key
K4N6	1468	6 hours, No result	20.59s, Correct Key
K4N7	1647	6 hours, No result	29.66s, Correct Key
K4N8	1849	6 hours, No result	31.93s, Correct Key
K5N2	815	6 hours, No result	5.68s, Correct Key
K5N3	1066	6 hours, No result	7.9s, Correct Key
K5N4	1477	6 hours, No result	17.18s, Correct Key
K5N5	1741	6 hours, No result	39.91s, Correct Key
K5N6	2012	6 hours, No result	35.57s, Correct Key
K5N7	2271	6 hours, No result	56.24s, Correct Key
K5N8	2573	6 hours, No result	74.89s, Correct Key
K6N2	1125	6 hours, No result	6.69s, Correct Key
K6N3	1518	6 hours, No result	13.29s, Correct Key
K6N4	2089	6 hours, No result	2.12s, Correct Key
K6N5	2694	6 hours, No result	32.193s, Correct Key
K6N6	2928	6 hours, No result	60.52s, Correct Key
K6N7	3347	6 hours, No result	73.47s, Correct Key
K6N8	3989	6 hours, No result	492.3s, Correct Key

approach than IcySAT when applied to circuits obfuscated with eFPGA-based redaction.

However, CycSAT is not without its flaws. The weakness of CycSAT lies in its inability to break hard cycles [16]. Hard cycles consist of oscillating and stateful cycles that would be missed in the acyclic condition generation of the CycSAT attack. Oscillating cycles do just as their name implies, and the signal of the cycle constantly oscillates between "0" and "1". Unbroken oscillating cycles may cause the SAT algorithm to return incorrect key values, as these cycles are unable to be simulated by the SAT solver. Stateful cycles may have different states and therefore can be satisfied by different values. The existence of stateful cycles cause the SAT solver to enter an infinite loop, continuously applying a wrong key without pruning incorrect values. CycSAT was able to decrypt the benchmarks because they lacked hard cycles. However it cannot break hard cycles from the circuit no matter what topological order it takes [16]. On the other hand, unrolling has the capability of modifying the cycles in the circuit, converting hard cycles to simple combinational cycles and renewing the viability of the SAT-based approach against this defense.

So far, it has been established that CycSAT is overall more efficient than IcySAT yet has a kryptonite of hard cycles. IcySAT is less efficient than CycSAT, but is able to handle hard cycles by unrolling. So, why not have the best of both worlds? In Section 4, by combining cycle breaking and cycle unrolling, we propose a novel attack which is able to recover the bitstream of eFPGA-based redaction schemes even with the existence of hard cycles and large size keys.

4 ATTACK ON FPGA-BASED REDACTION

In this section, after looking at the hard cycles that can prevent existing cyclic attacks from succeeding, we first present an initial but not necessarily feasible attack that excludes problematic DIPs to tackle the hard cycle problem. Then, combining cycle breaking and unrolling, we propose an efficient attack for finding the bitstream of eFPGA-based redaction schemes even with the existence of hard cycles and large key sizes. Finally, we discuss the suggested attacks' time complexity.

4.1 Hard Cycles

The original circuit depicted in Figure 1a has been obfuscated with different feedback signals as shown in Figure 1b. Each feedback in this example contains an additional key bit controlled gate to allow non-cyclic behavior when the correct key is applied. If the target gate is an AND/NAND gate, an OR key bit controlled gate is embedded; here the correct value of the associated key bit is "1". If the target gate is an OR/NOR gate, an AND key bit controlled gate is added with the correct value of "0".

The feedbacks in Figure 1b are combined in such a way that any topological order chosen by CycSAT [40] will result in a cycle being missed. In other words, the inclusion of "hard cycles" characterizes this type of obfuscation. When CycSAT is performed on this obfuscated circuit, it results in an infinite solver loop. This is because the edges in a directed graph cannot always be separated into two disjoint sets, such that each simple cycle is generated by two simple pathways, one made up of edges from each set [57].

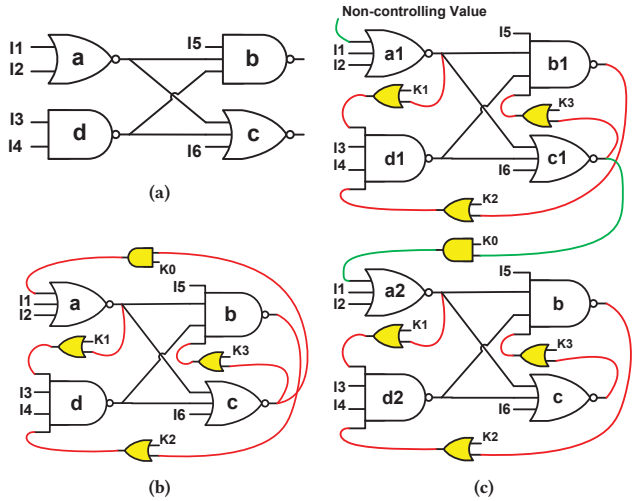

Figure 1: (a) Original circuit (b) Obfuscated circuit with hard cycles (c) Obfuscated circuit with one cycle being unrolled

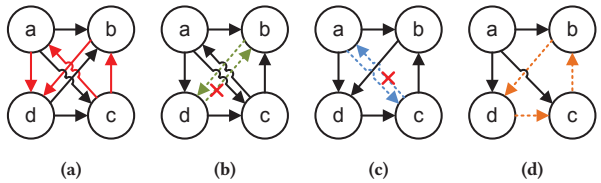

Figure 2: (a) Directed graph of Figure 1b with the topological order {a, b, c, d} (b) Cycle {b, d, b} can be broken (c) Cycle {a, c, a} can be broken (d) Cycle {b, d, c, b} cannot be broken

As a result, no matter what topological order the CycSAT attack employs, certain structural cycles may be overlooked that result in a stateful situation.

Despite the existing of hard cycles, the obfuscated circuit in Figure 1b could be easily deobfuscated by IcySAT [41] since it is a simple example for cycle unrolling. However, IcySAT alone would take exponential time to solve similar examples with larger Conjunctive Normal Form (CNF) sizes.

The graph representation of the obfuscated circuit of Figure 1b is shown in Figure 2a. In this graph, regardless of the traversal order used, at least one of the cycles will be missed in generating the acyclic condition of CycSAT. For example, if the topological order is {a, b, c, d}, CycSAT may break the cycles {b, d, b} and {a, c, a} shown in Figures 2b and c respectively, but not the cycle {b, d, c, b} shown in Figure 2d.

4.2 DIP Exclusion Attack

As mentioned in Section 4.1, the weakness of CycSAT [40] comes from non-combinational hard cycles in the circuit. In our first attacking attempt, we propose to record the DIPs that the attack got stuck on and exclude them from the eFPGA-based redaction benchmarks. The way this can be achieved is by connecting all primary inputs to a NAND gate for each input pattern to be excluded.

Algorithm 1: DIP exclusion attack

Input: Obfuscated circuit $g(x,k)$ and original function $f(x)$
Output: Key vector k^* such that $g(x,k^*) \equiv f(x)$
$W \leftarrow SearchFeedback(g(x,k))$;
// $W \leftarrow \{w_0, w_1, ..., w_m\}$
for *all* $w_i \in W$ **do**
  $F(w_i, w_i') \leftarrow No\_Structural\_Path(w_i)$;
$NC(k) \leftarrow \wedge_{i=0}^{m} F(w_i, w_i')$;
$g(x,k) \leftarrow g(x,k) \wedge NC(k)$;
$DIP_{set} \leftarrow \{\}$;
while $\hat{x} \leftarrow SAT(g(x,k_1) \neq g(x,k_2))$ **do**
  if $LoopDetected(\hat{x}, DIP_{set})$ **then**
    $g(x,k_1) \leftarrow Exclude(\hat{x}, g(x,k_1))$;
    $g(x,k_2) \leftarrow Exclude(\hat{x}, g(x,k_2))$;
  else
    $g(x,k_1) \leftarrow g(x,k_1) \wedge (g(\hat{x},k_1) = f(\hat{x}))$;
    $g(x,k_2) \leftarrow g(x,k_2) \wedge (g(\hat{x},k_2) = f(\hat{x}))$;
    $Add(\hat{x}, DIP_{set})$;
    // DIPs are kept in DIP_{set}
Return $k^* \leftarrow SAT(g(x,k_1))$;

Algorithm 2: Break & Unroll attack

Input: Obfuscated circuit $g(x,k)$ and original function $f(x)$
Output: Key vector k^* such that $g(x,k^*) \equiv f(x)$
while *(True)* **do**
  $W \leftarrow SearchFeedbackSignals(g(x,k))$;
  // $W \leftarrow \{w_0, w_1, ..., w_m\}$
  for *all* $w_i \in W$ **do**
    $F(w_i, w_i') \leftarrow BreakFeedback(w_i)$;
  $NoCyclicCNF(k) \leftarrow \wedge_{i=0}^{m} F(w_i, w_i')$;
  // $NoCyclicCNF(k) \leftarrow BreakFeedback(w_0) \wedge ... \wedge BreakFeedback(w_m)$
  $g(x,k) \leftarrow g(x,k) \wedge NoCyclicCNF(k)$;
  $DIP_{set} \leftarrow \{\}$;
  while $\hat{x} \leftarrow SAT(g(x,k_1) \neq g(x,k_2))$ **do**
    if $LoopDetected(\hat{x}, DIP_{set})$ **then**
      $w \leftarrow$ Select a feedback signal from W set;
      $g(x,k) \leftarrow NewCircuit(w, g(x,k))$;
      Break;
    $g(x,k_1) \leftarrow g(x,k_1) \wedge (g(\hat{x},k_1) = f(\hat{x}))$;
    $g(x,k_2) \leftarrow g(x,k_2) \wedge (g(\hat{x},k_2) = f(\hat{x}))$;
    $Add(\hat{x}, DIP_{set})$;
    // DIPs are kept in DIP_{set}
  if $!SAT(g(x,k_1) \neq g(x,k_2))$ **then**
    Return $k^* \leftarrow SAT(g(x,k_1))$;

Consider a certain bit B of the input pattern which we want to exclude. If $B = 0$, the corresponding input line to the NAND gate must be inverted. If $B = 1$, the input line may be fed directly to the NAND. Using this method, the output of this gate will be "0" only when the primary inputs equal the excluded DIP, and "1" for all the other input patterns. Next, an AND gate is added for every primary output. The inputs of this AND gate will be the original output line and the output of each exclusionary NAND gate. Thus, the excluded input pattern will make every output of the obfuscated netlist "0" regardless of the key value; in this case, the SAT solver will not consider this input pattern as a DIP and avoid entering an infinite loop. Algorithm 1 depicts the *DIP Exclusion* attack.

4.3 Break & Unroll Attack

The overall flow of the proposed *Break & Unroll* attack has been illustrated in Algorithm 2. This algorithm includes two main stages; first, breaking cycles, and second, unrolling the remaining cycles one by one. If the attack in the first step is successful and the correct key is displayed, this indicates the absence of the hard cycles in the circuit. Consequently, breaking the circuit cycles can thwart the cyclic obfuscation scheme. But if the attack fails to reveal the correct key by performing the first part of the algorithm and enters the infinite loop, then the existence of hard cycles has become a problem. To solve this, the attack exploits its second strategy, which is unrolling. By executing this phase, the inhibitory effect of the hard cycles would be neutralized. The details of both parts of the *Break & Unroll* attack will be discussed as follows.

4.3.1 Breaking Phase. Based on a primary topological order, a structural acyclic constraint has been constructed and added to the obfuscated circuit. First, all existing feedbacks of the circuit have been searched and kept in a set called W. In the next step, all cycles will be broken one by one and then, all the broken feedback signals will be accumulated as one CNF. This non-cyclic condition will be added as a new constraint to the obfuscated circuit.

Then, a new obfuscated circuit would be introduced with the hope of being without structural cycles. After adding this constraint, the original SAT solver will run this new version of the circuit. In regards to the discussion in section 4.1 on the consequences of missing a cycle, the SAT solver may get stuck in an infinite loop. The *LoopDetected* function recognizes whether running the first part of the *Break & Unroll* algorithm ends in the infinite loop or not. In this function, the new DIP will be compared with all members of a set that keeps all DIPs from prior iterations. If it does not belong to this set, it demonstrates that the breaking phase operated properly and will reveal the correct key in the next step. But if the function can find the new generated DIP in the set, it shows that the SAT-solver will go into an infinite loop. So, we need to address this issue in the second phase.

4.3.2 Unrolling Phase. The second part of the algorithm tries to conquer the breaking phase weakness by unrolling a single cycle at a time. Unrolling the current cycle will be accomplished by the *NewCircuit* function.

The process for unrolling the circuit is as follows: First, a single feedback is chosen. Next, a copy of every gate is added to the circuit. Considering the cyclic circuit in Figure 1b and the unrolled version of this same circuit in Figure 1c, the following feedback wires exist: a → d, b → d, c → a, and c → b.

The feedback is then broken and a non-controlling value substitutes with the fanout of this wire (i.e., gate a1 in Figure 1c) to enable other signals to pass through the circuit. As a reminder, a logic "1" is a non-controlling value for AND and NAND gates, whereas a logic

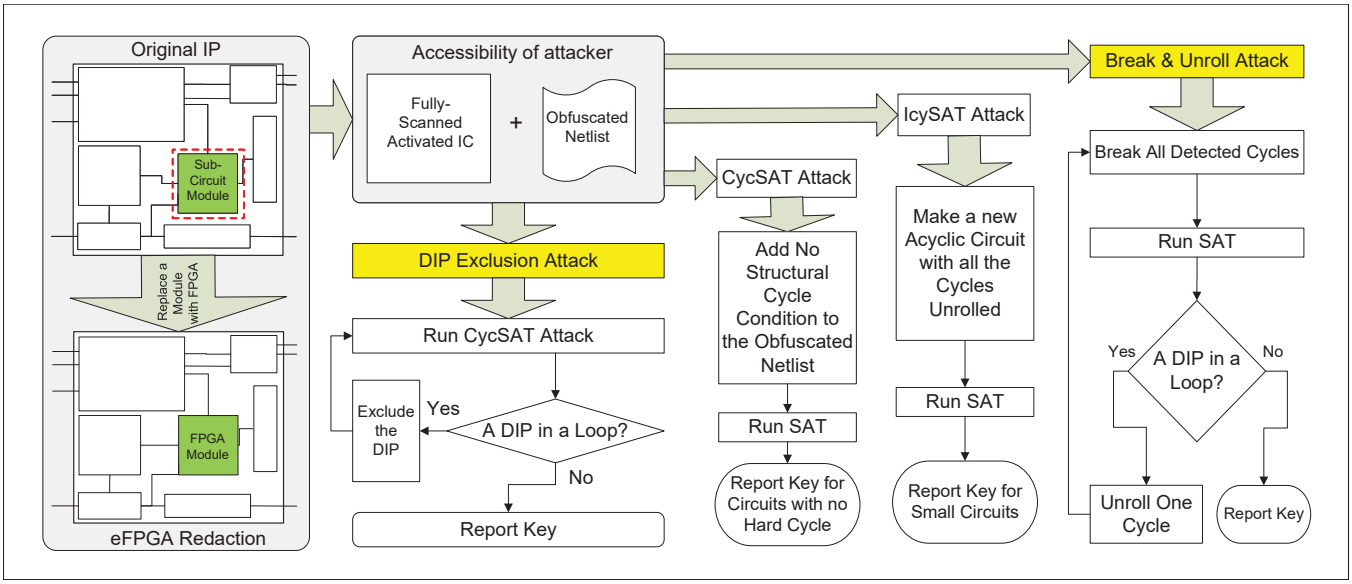

Figure 3: Attacks on eFPGA-based redaction schemes

"0" is a non-controlling value for OR and NOR gates. The feedback's fan-in (i.e., gate c1 in Figure 1c) is linked to the next copy of the gate to which it was previously attached. In Figure 1c, the wire from c1 was originally connected to a1, but this wire is detached and connected to a2 instead. The circuit is successfully unrolled for one iteration after these steps are completed.

After unrolling one cycle, a new version of the obfuscated circuit will be replaced with the previous circuit. The *NewCircuit* function will construct a new version of the circuit with regard to the previously selected feedback. Since more unrollings may be required, it is critical to update the feedback list so that it will not choose to unroll a previously unrolled feedback in the next round. Furthermore, we might find that some new feedback is added to the circuit after performing the unrolling phase. Therefore, set W must be updated each time in the body of the attack algorithm.

4.4 Time Complexity Discussion

If an exponential number of DIPs lead to a stateful situation, the worst-case time complexity of the *DIP Exclusion* attack will be exponential since at each iteration, we can only remove one of the problematic DIPs. On the other hand, the *Break & Unroll* attack has the same best-case scenario in comparison with CycSAT while its worst-case scenario is comparable to IcySAT. In any scenario where CycSAT is able to return the correct key, *Break & Unroll* has the same run-time. Alternatively, in cases where CycSAT returns an infinite loop, *Break & Unroll* is able to return the correct key with a much better average run-time than IcySAT.

Our unrolling technique is designed to reduce the time needed for each iteration. The completion time for unrolling is significantly decreased by choosing one cycle at a time and by creating only one additional copy of the circuit for each unroll. However, *Break & Unroll* may end in the exponential growth of the circuit and consequently exponential time response equivalent to IcySAT. This is possible in cases where every cycle in the circuit must be unrolled. Figure 3 shows a visual summary of the attacks on eFPGA-based redaction schemes.

5 EXPERIMENTAL RESULTS

To launch the attack, we used eFPGA-based redaction benchmarks in [5] with 3 and 4 input sizes of LUTs and different numbers of basic logic elements in a CLB. Then we added key-controlled hard cycles in two different ways. For half of the chosen benchmarks, we inserted a fixed number of 10 hard cycles into the whole circuit, and for the second half, we added the hard cycles compatible with the key size, i.e., the number of added hard cycles is one-tenth of the key size of each benchmark. Each hard cycle embeds four extra key bit controlled feedbacks into the circuit. All attacks are run in Ubuntu with 8GB of RAM. The results are shown in Table 2.

CycSAT gets stuck in an infinite loop for almost all of the benchmarks and cannot return any keys due to the existence of stateful hard cycles. On the other hand, IcySAT is still able to retrieve the correct key for smaller sized benchmarks, while for the rest of the benchmarks, after 6 hours, no result was reported. The benchmarks with hard cycles take more time to be attacked by IcySAT in comparison with the original benchmarks with no hard cycles due to the increased circuit complexity. IcySAT cannot return the correct key for the majority of the benchmarks because it tries to unroll all the cycles of the circuit, and unrolling all the cycles results in the exponential growth of the CNF size of the obfuscated circuit and thus a high response time for the solver.

For the *DIP exclusion* attack, as the number of hard cycles increases, the probability of more DIPs getting stuck in a loop becomes higher. Thus, the attack needs to exclude more DIPs. The *DIP exclusion* attack returns the key for the majority of the benchmarks with a fixed number of 10 hard cycles, but for the ones with a higher number of hard cycles, no results were received after 6 hours of

Table 2: Attacks on eFPGA-based redaction schemes with hard cycles

Bench	Key Size	Hard Cycles	CycSAT [40]	IcySAT [41]	DIP Exclusion (New)	Break & Unroll (New)
K3N2	641	10	Inf Loop	697s, Correct Key	1236.57s , Correct Key	9.46s, Correct Key
K3N3	765	10	Inf Loop	1817s, Correct Key	2163.11s , Correct Key	16.38s, Correct Key
K3N4	877	10	Inf Loop	5506s, Correct Key	5621.3s , Correct Key	41.81s , Correct Key
K3N5	981	10	Inf Loop	6 hours, No Result	7011.37s , Correct Key	36.42s , Correct Key
K3N6	1194	10	Inf Loop	6 hours, No Result	6 hours, No Result	141.34s , Correct Key
K3N7	1283	10	Inf Loop	6 hours, No Result	24011.02s, Correct Key	205.09s , Correct Key
K3N8	1433	10	Inf Loop	6 hours, No Result	6 hours, No Result	410.7s, Correct Key
K4N2	895	64	Inf Loop	1072s, Correct Key	6 hours, No Result	20.93s , Correct Key
K4N3	1134	81	UNSAT	26695s, Correct Key	6 hours, No Result	182.15s, UNSAT
K4N4	1469	105	Inf Loop	6 hours, No Result	6 hours, No Result	194.6s, Correct Key
K4N5	1844	132	Inf Loop	6 hours, No Result	6 hours, No Result	806.14s, Correct Key
K4N6	2056	147	Inf Loop	6 hours, No Result	6 hours, No Result	1519.89s, Correct Key
K4N7	2307	165	Inf Loop	6 hours, No Result	6 hours, No Result	1602.52s, Correct Key
K4N8	2589	185	Inf Loop	6 hours, No Result	6 hours, No Result	6 hours, No Result

running the attack due to the brute-force nature of excluding DIPs one by one.

On the other hand, the *Break & Unroll* attack is able to retrieve all correct keys except for two benchmarks. For $K4N3$, *Break & Unroll* the same as CycSAT, reports UNSAT . We could not find the real reason here, but we suspect that both CycSAT and *Break & Unroll* are breaking a cycle that should not be broken, maybe due to the existing combinational loop in the original benchmark.

Overall, based on the attacking results of Table 2, the superiority of the *Break & Unroll* attack is apparent. However, it appears that when a large number of hard cycles are embedded in the circuit (as in the $K4N8$ benchmark), even the *Break & Unroll* attack may take a long time to report the correct key. This is because, in the case of many hard cycles, the algorithm requires many unrollings that may degrade the performance of *Break & Unroll* due to the growth of the CNF size. This suggests the necessity of systematically inserting an exponential number of hard cycles into eFPGA-based redaction methods. However, this is not an easy task to carry out without significant overhead to the original eFPGA fabrics.

6 CONCLUSION & TAKEAWAYS

The primary goal of logic obfuscation is to protect a precious design from being pirated and overproduced. If we replace large parts of the design with an eFPGA, we cannot guarantee the same performance as the original design. So, eFPGA-based redaction is only meaningful when a small-sized sub-circuit is replaced with an eFPGA. In this case, as shown by our preliminary study, the bitstream of the eFPGA (i.e., the correct key) can be revealed by cyclic attacks like CycSAT [40] and IcySAT [41] if the attacker has access to an oracle (i.e., an activated IC bought from the market) and the obfuscated netlist leaked from an untrusted foundry.

Takeaway 1: The common perception that eFPGA-based redaction is by default secure against oracle-guided attacks is prejudice.

Thus, provisions to insert hard cycles (against CycSAT) and large-size keys (against IcySAT) are required. However, even with these provisions, eFPGA-based redaction can be challenged by our proposed attacks: *DIP Exclusion* attack that excludes problematic DIPs form the CycSAT algorithm to tackle the hard cycle problem, and *Break & Unroll* attack that combines cycle breaking and unrolling to find the bitstream of eFPGA-based redaction schemes even with the existence of hard cycles and large key sizes.

Takeaway 2: Additional research on how to systematically create an exponential number of non-combinational hard cycles is required to produce secure eFPGA-based redaction schemes.

In the future, the *Break & Unroll* attack could be extended to accept an unrolling factor as user input to select the number of unrollings each time a DIP becomes stuck in a loop.

7 ACKNOWLEDGEMENT

We thank reference [5] authors and CSAW'21 logic locking conquest organizers for providing us with the eFPGA-based redaction benchmarks.

REFERENCES

[1] R. Torrance and D. James,"The state-of-the-art in semiconductor reverse engineering," In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 333-338, 2011.

[2] B. Hu, J. Tian, M. Shihab, G. R. Reddy, W. Swartz, Y. Makris, B. C. Schaefer, and C. Sechen, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded FPGA," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 171-176, 2019.

[3] P. Mohan, O. Atli, J. Sweeney, O. Kibar, L. Pileggi, and K. Mai, "Hardware redaction via designer-directed fine-grained eFPGA insertion," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1186-1191, 2021.

[4] J. Bhandari, A. K. Thalakkattu Moosa, B. Tan, C. Pilato, G. Gore, X. Tang, S. Temple, P. -E. Gaillardon, and R. Karri, "Exploring eFPGA-based redaction for IP protection," In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1-9, 2021.

[5] J. Bhandari, A. K. Thalakkattu Moosa, B. Tan, C. Pilato, G. Gore, X. Tang, S. Temple, P. -E. Gaillardon, and R. Karri, "Not all fabrics are created equal: Exploring eFPGA parameters for IP redaction," In *arXiv:2111.04222 [cs.CR]*, 2021.

[6] Z. U. Abideen, T. D. Perez, and S. Pagliarini, "From FPGAs to obfuscated eASICs: Design and security trade-offs," In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1-4, 2021.

[7] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*,

pp. 1069-1074, 2008.
[8] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 953-958, 2012.
[9] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," In *Design Automation Conference (DAC)*, pp. 83–89, 2012.
[10] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137-143, 2015.
[11] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236-241, 2016.
[12] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, p. 127-146, 2016.
[13] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1601–1618, 2017.
[14] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-SAT era by locking & obfuscation," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 13-18, 2020.
[15] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating SAT-unresolvable circuits" In *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 173-178, 2017.
[16] A. Rezaei, Y. Shen, S. Kong, J. Gu, and H. Zhou, "Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks," In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 85-90, 2018.
[17] H. -Y. Chiang, Y. -C. Chen, D. -X. Ji, X. -M. Yang, C. -C. Lin, and C. -Y. Wang, "LOOPLock: Logic optimization-based cyclic logic locking," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2178-2191, 2020.
[18] X. -M. Yang, P. -P. Chen, H. -Y. Chiang, C. -C. Lin, Y. -C. Chen, and C. -Y. Wang, "LOOPLock 2.0: An enhanced cyclic logic locking approach," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 1, pp. 29-34, 2022.
[19] A. Rezaei, Y. Li, Y. Shen, S. Kong, and H. Zhou, "CycSAT-unresolvable cyclic logic encryption using unreachable states" In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 358–363, 2019.
[20] R. Afsharmazayejani, H. Sayadi, and A. Rezaei, "Distributed logic encryption: Essential security requirements and low-overhead implementation," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 127-131, 2022.
[21] A. Rezaei, J. Gu, and H. Zhou, "Hybrid memristor-CMOS obfuscation against untrusted foundries," In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 535-540, 2019.
[22] H. Zhou, Y. Shen, and A. Rezaei, "Vulnerability and remedy of stripped function logic locking," In *Cyptology ePrint Archive*, report 2019/139, 2019.
[23] S. Roshanisefat, H. M. Kamali, H. Homayoun, and A. Sasan, "SAT-hard cyclic logic obfuscation for protecting the IP in the manufacturing supply chain" In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 954–967, 2020.
[24] H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, and A. Sasan, "LUT-lock: A novel LUT-based logic obfuscation for FPGA bitstream and ASIC-hardware protection," In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 405–410, 2018.
[25] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "Cross-lock: Dense layout-level interconnect locking using cross-bar architectures," In *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 147– 152, 2018.
[26] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "Full-Lock: Hard distributions of SAT instances for obfuscating circuits using fully configurable logic and routing blocks," In *Proceedings of Design Automation Conference (DAC)*, pp. 1-6., 2019.
[27] R. Karmakar, S. Chatopadhyay, and R. Kapur, "Encrypt flip-flop: A novel logic encryption technique for sequential circuits," In *arXiv:1801.04961*, 2018.
[28] D. Zhang, M. He, X. Wang, and M. Tehranipoor, "Dynamically obfuscated scan for protecting IPs against scan-based attacks throughout supply chain," In *VLSI Test Symposium (VTS)*, pp. 1–6, 2017.
[29] R. Karmakar, H. Kumar, and S. Chattopadhyay, "Efficient key-gate placement and dynamic scan obfuscation towards robust logic encryption," In *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 2109-2124, 2019.
[30] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 818–830, 2018.
[31] G. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "Timing camouflage: Improving circuit security against counterfeiting by unconventional timing," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 91–96, 2018.
[32] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting," In *Design Automation Conference (DAC)*, pp. 1–9, 2017.
[33] A. Rezaei and H. Zhou, "Sequential logic encryption against model checking attack," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1178-1181, 2021.
[34] J. Sweeney, V. Zackriya, V S. Pagliarini, and L. Pileggi, "Latch-based logic locking," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 132–141, 2020.
[35] K. Z. Azar, H. M. Kamali, S. Roshanisefat, H. Homayoun, C. Sotiriou, and A. Sasan, "Data flow obfuscation: A new paradigm for obfuscating circuits," In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 643-656, 2021.
[36] M. Yasin, C. Zhao, and J. Rajendran, "SFLL-HLS: Stripped functionality logic locking meets high-level synthesis," In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1–4, 2019.
[37] C. Pilato, F. Regazzoni, R. Karri, and S. Garg, "TAO: Techniques for algorithm-level obfuscation during high-level synthesis," In *Design Automation Conference (DAC)*, pp. 1–6, 2018.
[38] R. Muttaki, R. Mohammadivojdan, M. Tehranipoor, and F. Farahmandi, "HLock: Locking IPs at the high-level language," In *Design Automation Conference (DAC)*, pp. 79–84, 2021.
[39] N. Limaye, A. Chowdhury, C. Pilato, M. Nabeel, O. Sinanoglu, S. Garg, and R. Karri, "Fortifying RTL locking against oracleLess (untrusted foundry) and oracle-guided attacks," In *Design Automation Conference (DAC)*, pp. 91–96, 2021.
[40] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 49–56, 2017.
[41] K. Shamsi, D. Z. Pan, and Y. Jin, "IcySAT: Improved SAT-based attacks on cyclic locked circuits," In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2019.
[42] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott, and H. Zhou, "BeSAT: Behavioral SAT-based attack on cyclic logic encryption," In *Asia and South Pacific Design Automation Conference (ASPDAC)*, pp 657-662, 2019.
[43] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 179-184, 2017.
[44] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 95-100, 2017.
[45] X. Xu, B. Shakya, M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," In *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 189-210, 2017.
[46] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based attack on compound logic locking techniques," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1166-1171, 2021.
[47] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," In *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517-532, 2020.
[48] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," In *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514-2527, 2020.
[49] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," In *IACR Transactions on Cryptographic Hardware and Embedded Systems*, Vol 2019, issue 1, pp. 97-122, 2019.
[50] M. E. Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 33–40, 2017.
[51] Y. Kasarabada, S. Chen, and R. Vemuri, "On SAT-based attacks on encrypted sequential logic circuits," In *International Symposium on Quality Electronic Design (ISQED)*, pp. 204-211, 2019.
[52] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," In *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744-759, 2022.
[53] S. Roshanisefat, H. M. Kamali, H. Homayoun, and A. Sasan, "RANE: An open-source formal de-obfuscation attack for reverse engineering of logic encrypted circuits," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 221–228, 2021.
[54] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "SigAttack: New high-level SAT-based attack on logic encryptions," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 940-943, 2019.
[55] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based bit-flipping attack on logic encryptions," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 629-632, 2018.
[56] Y. Shen, A. Rezaei, and H. Zhou, "A comparative investigation of approximate attacks on logic encryptions," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 271-276, 2018.
[57] R. Chen and H. Zhou, "Statistical timing verification for transparently latched circuits," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 9, pp 1847-1855, 2006.