Cross-Lock: Dense Layout-Level Interconnect Locking using Cross-bar Architectures

Kaveh Shamsi[1], Meng Li[2], David Z. Pan[2], and Yier Jin[1]
[1]ECE Department, University of Florida, Gainesville, FL, USA
[2]ECE Department, University of Texas at Austin, Austin, TX, USA
kshamsi@ufl.edu, {meng_li, dpan}@utexas.edu, yier.jin@ece.ufl.edu

ABSTRACT

Logic locking is an attractive defense against a series of hardware security threats. However, oracle guided attacks based on advanced Boolean reasoning engines such as SAT, ATPG and model-checking have made it difficult to securely lock chips with low overhead. While the majority of existing locking schemes focus on gate-level locking, in this paper we present a layout-inclusive interconnect locking scheme based on cross-bars of metal-to-metal programmable-via devices. We demonstrate how this enables configuring a large obfuscation key with a small number of physical key wires contributing to zero to little substrate area overhead. Dense interconnect locking based on these circuit level primitives shows orders of magnitude better SAT attack resiliency compared to an XOR/XNOR gate-insertion locking with the same key length which has a much higher overhead.

CCS CONCEPTS

• **Security and privacy → Tamper-proof and tamper-resistant designs**; **Hardware reverse engineering**;

KEYWORDS

Logic Obfuscation; Hardware Security; Logic Locking; SAT Attacks

ACM Reference Format:
Kaveh Shamsi, Meng Li, David Z. Pan, and Yier Jin. 2018. Cross-Lock: Dense Layout-Level Interconnect Locking using Cross-bar Architectures. In *GLSVLSI '18: 2018 Great Lakes Symposium on VLSI, May 16–18, 2018, Chicago, IL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3194554.3194580

1 INTRODUCTION

Logic locking and integrated circuit (IC) camouflaging are two closely related techniques for hiding the design of an integrated circuit. These techniques will make it difficult for an attacker to recover the original netlist of the design. This in turn can hinder various attacks such as reverse-engineering, intellectual property (IP) theft, and malicious modification of the design. Recent reports suggest that IP theft is on the rise and concerns about design integrity through the global IC supply chain remain strong [16]. Hence, logic locking and camouflaging if made secure with low overhead can alleviate great concerns in the semiconductor industry.

Logic locking is based on inserting additional "key inputs" into the circuit so that the circuit produces incorrect outputs for incorrect keys. The key bits are programmed post-fabrication by a trusted entity hiding the original design from the foundry as well as end-users. IC camouflaging is based on using layout constructs and special fabrication technologies to hinder reverse-engineering. In camouflaging, while the foundry knows the true functionality, it is difficult for an end-user attacker to obtain an accurate netlist of the camouflaged layout through IC delayering and imaging [24].

The security of a locking/camouflaging scheme versus the overhead that it incurs to the design is the primary metric of evaluation. The security of the protection scheme heavily relies on the considered threat model. The "oracle-guided" threat model which is widely used, allows the attacker to make arbitrary input-output queries on a functional/unlocked oracle-circuit. Several attacks are possible under this threat model, the strongest being the notorious SAT [5, 21] attack that uses SAT queries to construct and solve a system of input-output equations to find the correct functionality of the obfuscated netlist.

Several "SAT-resilient" locking/camouflaging schemes [10, 25, 27–29, 31, 32] have been proposed based on inserting large gate-level tree-structures that increase exponentially the minimum number of queries required for the attack. Unfortunately these schemes cannot satisfy entropy criteria due to a fundamental contention between entropy and minimum query count. Hence, they are vulnerable to approximate attacks such as AppSAT [17] and DDIP [19]. In addition they can be vulnerable to removal attacks [30] due to the fixed structure of the inserted trees.

In IC camouflaging it is possible to "flood" the design layout with a prohibitively large number of dummy elements inserted in the abundant empty spaces in the layout [2]. These approaches show a high resiliency against SAT attacks [13] simply by creating very large SAT instances that overwhelm the SAT-solver. These approaches can overcome the fundamental error versus resiliency contention, as they result in complex SAT instances rather than exponentially large minimum query counts. These layout-level approaches however, have not been applied to logic locking. This is primarily due to the key-bit programming circuitry needed for logic locking which not only incurs additional overhead, but also reveals the location of the ambiguous circuit elements, narrowing the search for the attacker. This hinders large-scale layout-level logic locking. As a result, the majority of existing logic locking schemes are forced to insert large gate-level primitives without the ability to exploit special transistor structures.

In this paper however, we start with the design of transistor-level primitives, in particular key bit programming architectures and then build netlist-level routines upon the primitive. We present

GLSVLSI '18, May 16–18, 2018, Chicago, IL, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5724-1/18/05…$15.00
https://doi.org/10.1145/3194554.3194580

a layout-inclusive locking scheme based on cross-bar programmable interconnect architectures. We demonstrate how using cross-bar circuit architectures a large number of key elements can be programmed sequentially with a much smaller number of physical key wires. We show that without adding a single gate to the design, just by densely obfuscating interconnects, small circuits can withstand advanced SAT attacks orders of magnitude longer than state-of-art gate-insertion strategies.

The paper makes the following contributions:

- We study various programmable device technologies in terms of attack resiliency and footprint. We focus on metal-to-metal programmable vias and define a general layout-level locking scheme based on these devices with minimal programming structures.
- We analyze the functional level security of the proposed interconnect locking scheme based on these primitives, demonstrating how densely cyclic interconnect locking can result in hard SAT instances for the attack.
- We demonstrate a custom design flow for inserting cross-bar primitives in the layout using a 32/28nm technology library and report design metrics which shows overhead values orders of magnitude better than gate-insertion strategies.

The paper is organized as follows: Section 2 provides a brief background. Section 3 discusses circuit-level locking primitives, and Section 4 discusses netlist-level routines. Section 5 presents experiments and Section 6 concludes the paper.

2 BACKGROUND

Oracle-guided attacks: In studying locking/camouflaging it is a basic assumption that the attacker is able to recover the netlist of the locked/camouflaged IC. Any locked/camouflaged netlist contains a number of ambiguous elements from the attacker's view. These ambiguous parts can be encoded using "key" variables over which a space of possible functions for the netlist is defined. The original circuit is $c_o : I \to O, I = \mathbb{F}_2^n, O = \mathbb{F}_2^m$ and the locked/camouflaged function is $c_e : I \times K \to O$ with $K = \mathbb{F}_2^l$. The function space $C = \{c_e(i,k) | k \in K\}$ is defined and $\exists k_* \in K_*$ s.t. $\forall i \in I \;\; c_e(i,k_*) = c_o(i)$.

Oracle-guided attacks assume input-output access to c_o. The baseline SAT attack [5] starts by finding an input pattern that can distinguish between different keys, namely a discriminating input pattern (DIP), by solving the mitter SAT problem $M \equiv (c_e(i,k_1) \neq c_e(i,k_2))$ with $\hat{i}$, $\hat{k_1}$, and $\hat{k_2}$. It then queries the DIP $\hat{i}$ on the oracle: $\hat{y} = c_o(\hat{i})$, and adds the input-output pair to the SAT problem, $M \leftarrow M \wedge (c_e(\hat{i},k) = \hat{y})$. It terminates once M is not satisfiable, i.e. no more DIPs can be found. At this point any key that satisfies all DIP-output observations is the correct key.

The CycSAT Attacks: While oracle-guided attacks based on solving systems of equations in key variables are generally applicable to any keyed Boolean function (including stateful circuits using model-checking [12]), the baseline SAT attack is only applicable to acyclic combinational circuits. Shamsi et al. [18] proposed using cyclic interconnect obfuscation to thwart the attack. However, Zhou et al. [33] developed an algorithm called CycSAT to add a condition to the mitter M so that the solver avoids cyclic solution breaking cyclic obfuscated circuits. We use the structural CycSAT algorithm as our main attack in this paper.

3 CIRCUIT-LEVEL PRIMITIVES

3.1 Key Storage Element

All locking/camouflaging schemes are built upon a specific physical element that creates ambiguity for the attacker. The security of any netlist/chip-level scheme heavily relies on the security of this physical primitive. For IC camouflaging the main criteria for the element is that it should resist physical IC reverse-engineering procedures while allowing the foundry to resolve and fabricate it. One example for such primitives is vias that have a middle-gap [14] which falsely appear to be connecting two metal layers. Another is Mg-based [1] vias, which when exposed to oxygen during reverse-engineering, all transform into MgO insulating vias hindering the recovery of an accurate netlist.

In logic locking, the physical primitive in addition to being resilient to physical reverse-engineering, should be programmable post-fabrication resulting in fewer options. The size of the device and the programming and read circuitry become important metrics. We study several technologies with respect to these criteria herein.

The first category for logic locking primitives is CMOS based volatile memory elements such as SRAM cells, and D-flip-flops which are all based on the cross-coupled inverter pair. These can store bits as voltage levels on internal wires. In terms of security, since voltage levels are difficult to read on a large scale in advanced technology nodes, these primitives are acceptable[1]. In terms of cell area, the smallest storage unit which is the SRAM cell requires 6 transistors. In terms of programming circuitry while DFFs can be chained together for serial programming of key bits, SRAM cells require memory architectures for programming. The volatility of these elements is problematic as well, since the key must be stored on an external tamper-proof nonvolatile memory and transferred to the device which introduces its own security risks[2].

Embedded Nonvolatile Memory (NVM) technologies such as Flash, EEPROM, Resistive-RAM (RRAM), and Spin-based NVMs that use Magnetic-Tunneling Junctions (MTJs) can be used to store key bits as well. In terms of security, Flash and EEPROMs can be read out with special laborious chip reverse-engineering procedures at small scale [4]. The magnetic polarity of spin-based memory elements such as MTJs can be read with Magnetic Force Microscopy (MFM) with sub-micron resolution on a small scale [7]. RRAM cells are particularly difficult to read under scanning electron microscopy [26]. In terms of area, EEPROM and Flash have 1~2 transistors in their footprint, whereas RRAM cells and MTJs are placed on the metal layers with no substrate area and can be as small as $1F^2$. EEPROM and Flash require CMOS read-out circuits to convert internal charge to digital level voltage. Similarly MTJs require a sense-amplifier to convert the small resistance difference to a logic level voltage. The sense-amplifier can have a minimum of 6 transistors. RRAM cells however, can have very high R_{on}/R_{off}

[1]Non-conventional approaches exist for limited voltage level read-out on integrated circuits [11]. Most are based on pointing a laser beam at the chip and analyzing the variations on the reflected beam that are caused by the electric field between two lines with opposite voltages. A recent study [11] used backside laser imaging which allowed distinguishing between an oscillating signal and a static one on a 60nm FPGA device. Note that if the attacker is able to read voltage levels at a large scale, secure hardware locking/camouflaging is most likely impossible at least with reasonable overhead.

[2]It is possible to overcome the volatility by using CMOS *aging* mechanisms to create bias in the memory element so that it always starts up in the forced state [9]. This comes at the cost of additional aging circuitry.

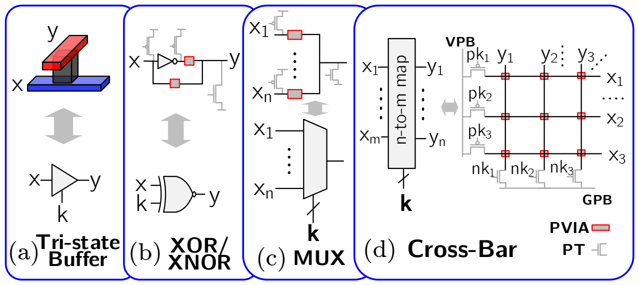


Figure 1: Various PVIA-based locking primtives.

ratios, which allows them to be directly placed on the signal path [23] removing the need for translation circuitry.

The last category is one-time-programmable (OTP) elements such as OTP EPROM, fuse [6] and anti-fuse [3] technologies. These are best suited for logic locking since the locked circuit is typically not configured more than once. Among OTP technologies metal-to-metal anti-fuse devices [3] are particularly suited to interconnect locking. They are a mature technology used in FPGAs and PLAs based on metal-insulator-metal (MIM) structures that connect two adjacent metal layers. They start off in an insulating state and can be made conducting by applying voltage to the two ends. They have a footprint similar to that of conventional vias, and are reasonably secure against invasive attacks [3]. We base our schemes in this paper on this technology, while we do note that any metal-to-metal configurable connector can be used as long as it is secure against invasive attacks, and has a small footprint. We refer to this element as a programmable via (PVIA) in the remainder of the paper.

3.2 PVIA Programming

PVIAs are programmed by applying a programming voltage for a known period of time to the two ends of the device. The specific voltage range for programming is dependent on the particular technology [3]. However, it must remain above the peak AC voltage and current that the element experiences during normal operation to avoid inadvertent programming. It also should be below the gate-oxide breakdown voltage of the logic transistors to avoid damaging them during programming. The power lines of the logic should be separate from the programming circuitry to avoid short circuits. The main approach for PVIA programming is connecting two complementary (NMOS and PMOS) programming transistors (PTs) to the two ends that connect the device terminals to programming supplies [26]. This way by configuring the two transistors and programming supplies, the cell can be programmed and then disconnected from the logic. The PTs can also be replaced with larger multi-level drivers to suppress sneak paths and provide protection.

The key idea that enables area saving is sharing PTs/drivers. If N PVIAs share one terminal they can share a single PT for that terminal. PVIA-based logic locking is any logic locking where the secret values are stored in the PVIAs. It is possible to construct a minimal programming network using this idea for arbitrarily inserted PVIAs in between a set of nets.

3.3 PVIA-based Locking Primitive

While it is possible to randomly insert a series of PVIAs in the netlist and construct a minimal programming structure for them, another strategy is to first use PVIAs to construct several primitives/modules and then perform the locking using these pre-designed primitives. These primitives can be evaluated in terms of their PVIA/PT ratio, sneak-paths and so on.

As seen from Figure 1 PVIAs can be used to implement an array of different locking primitives. Per Figure 1a, the single PVIA implements a tri-state buffer and needs two PTs. The XOR/XNOR gate which is heavily used in traditional logic locking, can be implemented with an inverter, two PVIAs and 3 PTs as seen in Figure 1b. Multiplexers (MUXs) are particularly suited to PVIA implementations. A single-level or multi-level tree can be used to implement an n-to-1 MUX as seen in Figure 1c[3].

The $n \times m$ cross-bar seen in Figure 1d shows the highest PVIA/PT ratio. nm PVIAs can be programmed with only $n + m$ PTs with a sequential write scheme where a horizontal line and a vertical line are connected to opposing voltages while other signals are left floating. If there is only one connected node in each column, the shortest sneak paths will have at least two PVIAs during programming. Otherwise, the non-active lines may need to be configured to a middle-voltage ($\frac{VPG}{2}$) with multi-level drivers. The sequence of configurations is also important in reducing the risk of inadvertent programming. This paper focuses on using cross-bar primitives for interconnect locking which we term *cross-lock*.

Note that the sharing of programming signals can be furthered using the *banking* technique used in memory array designs. Per Figure 1d, the nk_i and pk_i signals of N different cross-bars can be shared while replicating only the VPB and GPB signals N times and using them as arbitrators that select the active cross-bar.

4 NETLIST-LEVEL INTERCONNECT LOCKING

We can perform interconnect locking by inserting individual PVIAs or PVIA-based primitives in the netlist. Security against various attacks per the amount of overhead added to the design is the main criteria for netlist level locking which we discuss herein.

4.1 Security Against SAT Attacks

SAT attacks are the strongest oracle-guided attacks against combinational locking/camouflaging. The SAT attack and its variants are based on constructing and solving systems of input-output equations. The first natural approach towards thwarting these attacks, which includes the majority of the existing literature, is to increase the minimum number of queries (equations) required for the attack to succeed. The second which has seen little progress in literature, is making the SAT instances for querying and system-solving intractable. Due to the fundamental contention between query count and error rate, the only way to achieve a high error rate while maintaining attack resiliency is to ensure that the system of input-output observations cannot be solved efficiently even if it has very few equations (queries) in it.

To better understand SAT attack resiliency it is useful to look at similar problems across other domains. It has been noted in logic obfuscation literature that the SAT attack is related to the problem of active-learning [10] and the "query-by-disagreement probably-approximately-correct (PAC)" learning schemes [17]. While these relations are interesting, they cannot practically help advance SAT attack resiliency checking beyond what simple truth-table based analysis of Boolean functions provides us with.

[3]Note the difference between n tri-state buffers connecting n inputs to one output, and an n-to-1 MUX. In the former multiple inputs can be connected to the output while the latter is a one-hot connection.

A much more useful connection is to the domain of cryptography. A cryptographic function $f(x, k)$ is said to be Chosen-Plaintext-Attack (CPA)-secure, if k cannot be found from arbitrary queries of the form $(x_i, f(x_i, k))$. Researchers have been studying attacks against cryptographic functions for decades. A major category of attacks is *algebraic attacks* in which the main goal is to construct and solve a system of input-output equations to find the key. Algebraic attacks model deobfuscation attacks very closely, with the distinction that in practical algebraic attacks the system is typically pre-compiled and optimized whereas the SAT deobfuscation attack builds the system itself iteratively. Interestingly, in recent years SAT-solvers have become a promising tool for algebraic attacks against various block and stream ciphers [8].

While there are many aspects of this relation that we can exploited, here we focus on the "transition-phase" in the runtime behavior of SAT-based algebraic attacks[8], where SAT instances suddenly becomes intractable when the number of cipher rounds exceeds a certain limit, while fewer rounds are solved in a matter of minutes. In experimenting with interconnect locking we have observed a similar behavior where the SAT queries' search trees begin to explode as the obfuscated circuit graph gets closer to a complete-graph, even though the query count is low and the circuit is relatively small.

Hence we gear our netlist locking routines towards increasing the edge density of parts of the circuit using each cross-bar. These strategies are based on first finding the smallest candidate wire set W where $|W| > max(n, m)$ for each $n \times m$ cross-bar (n MUXs of size m where $m \geq n$). Then n wires are randomly selected from W and opened and routed through the cross-bar which occupies n inputs and n outputs of the cross-bar. Then the remaining $m - n$ cross-bar inputs are also selected from W and connected. The strategies that we implement for picking W include: a) *k-cut*: we search the circuit for a logic cone with k inputs and gradually increase k until a large enough set is found. b) *wire-cut*: we search for a cone with a sufficient number of wires by backward breadth-first-searches on each wire in the circuit. c) *diagonal*: we start exploring the wires in the distance d of each wire in the forward and backward directions while gradually growing d until a sufficiently large candidate set is found. Figure 2 shows an example of inserting a 3×3 cross-bar where $W = \{w_1, w_2, O\}$.

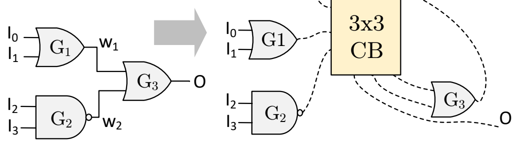


Figure 2: inserting a 3x3 cross-bar in a netlist.

4.2 Security Against Removal Attacks

Removal attacks or oracle-less attacks target special structures in the obfuscated netlist that can allow removing key possibilities without making queries. As for any MUX-based interconnect locking, we discuss two possible removal attacks. First is that if a key-controlled MUX output can reach one of its own inputs without interfering with any other key bit in the circuit, that MUX input can be removed if the assumption is that the original circuit is acyclic (removable backward edge [18]). We check for such edges using a simple readability search on the circuit graph starting from the output wires. We avoid creating such edges in cross-lock. Luckily few such edges arise when density driven selection strategies are used. Another possible removal attack is proximity based attacks that try to connect the given edge to the nearest location. These attacks can be thwarted by avoiding edges between distant nodes in the circuit which is ensured by the density driven selection strategies.

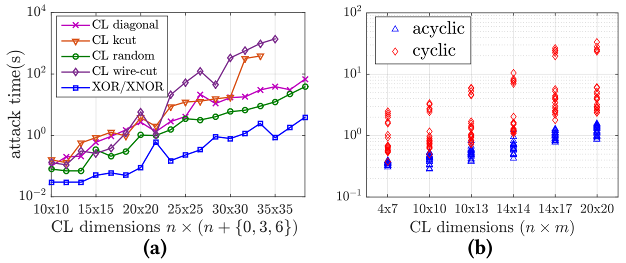


Figure 3: a) Different cross-lock insertion strategies and dimensions on the c432 benchmark circuit versus SAT attack runtime. b) Maximally acyclic versus maximally cyclic insertion attack run-times on random 8-input circuits.

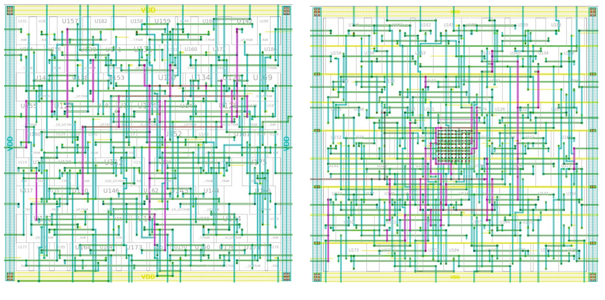
5 EXPERIMENTS

5.1 Resiliency Testing

We first perform netlist-level resiliency tests. Netlist-level attacks and defenses are implemented in a C++ framework. The baseline SAT attack is the structural CycSAT attack with AppSAT intermediate key extraction and error characterization capabilities included, using MiniSat as the back-end SAT-solver. Tests were run on a server machine with 16 Intel Xeon E5 CPUs clocked at 2.6GHz, running linux with 128GB of memory.

CNF-SAT modeling: When deobfuscating an $n \times m$ cross-bar the attacker can model the cross-bar in several ways: a) with n m-input MUXs each of which implemented using a tree of 2-input MUXs and transformed to conjunctive-normal-form (CNF) formula. b) n m-input MUXs each implemented with AND/OR/INV gates. c) nm key-controlled tri-state buffers (TBUF). The CNF formula for a TBUF depicted in Figure 1a is $(\bar{k} + x + \bar{y})(\bar{k} + \bar{x} + y)$. Only the TBUF-based model implements the "subset-MUX" where any subset of m inputs can be connected to the output. When using the TBUF-based model it is important to include a *no-float* condition on the key to ensure that all TBUFs that control a node do not disconnect at the same time. If the TBUF-based model is used there will be a quadratic number of key-bits. Surprisingly, the SAT attack performs better with the much larger TBUF-based model versus MUX-based models. We suspect that this is related to the system of equations becoming "over-defined" [8]. We use TBUF models to favor the attacker in our experiments. We define the equivalent key-length of the cross-bar as its MUX-based model key-length $n\lceil\log_2(m)\rceil$ for comparison with other locking schemes such as random XOR/XNOR insertion [15].

Density versus Resiliency: Figure 3a shows the running time of the attack for one cross-bar inserted in the c432 benchmark which has 160 primitive gates with different insertion strategies. The attack was run 3 times on each netlist. With high error-rate schemes it is important to rerun the attack several times with different SAT-search seeds and report the shortest time. As can be seen from



(a) (b)

Figure 4: c432 circuit. a) original layout. b) 10×10 **cross-bar inserted in the middle.**

edge-density		0.05		0.1		0.2		0.3		0.4	
#ins	#gates	#DI	t(s)	#DI	t(s)	#DI	t(s)	#DI	t(s)	#DI	t(s)
4	9.2	3.2	<0.0	5.0	<0.0	8.6	0.01	10.2	0.3	12.2	5.5
5	20.6	13.4	0.04	16.0	0.09	14.2	1895	3.8	44.3	TO	TO
6	39.1	20.6	1106	30.0	2697	TO	TO	TO	TO	TO	TO

Table 1: Runtime and query count average for random circuits versus edge density.

Figure 3a, the high density cone-based selection strategies, k-cut and wire-cut, surpass random cross-lock insertion by an order of magnitude exceeding the 1 hours deadline at smaller dimensions. Diagonal selection performs better than random selection, but is not as formidable as the cone-based strategies. This is while none of the circuits locked with random XOR/XNOR locking of the equivalent key-length resist the attack longer than 5s over various dimensions.

The relationship between density and SAT attack complexity becomes clearer when we try to create a complete-graph in the netlist. That is when every wire can be potentially connected to every other wire from the attacker's point of veiw. This is the case when a fully configurable interconnect similar to an FPGA is used. We generated 10 random circuits for small input widths by synthesizing random truth-tables using ABC [22]. We then began adding dummy edges to each netlist to achieve a certain graph density ($d = \frac{2|E|}{|E|(|E|-1)}$). As can be seen from Table 1 beyond a certain density a small circuit with 5-inputs cannot be resolved within the 3 hour deadline. This is an interesting result, since it takes less than a second for the SAT attack to resolve a 5-input look-up-table that implements all 2^{32} different 5-input Boolean functions! Hence, clearly a transition in the nature of the SAT problems is occurring as density increases. None of the existing logic locking schemes, even SAT-resilient schemes can last hours under the attack on a circuit with such a small number of inputs.

Cyclicness versus Resiliency: Another important question is whether cyclicness of the locked circuit improves resiliency. Figure 3b shows attack runtimes on 50 8-input random circuits with random cross-lock insertion with different dimensions when all added edges are forward edges, versus when most edges are feedback edges. This is done by topologically sorting the wires, and in the acyclic selection connecting wires late in the order to wires earlier in the order, while doing the opposite for the cyclic selection. It can be seen that cyclic insertion results in higher attack runtimes compared to acyclic locked circuits. The non-cyclic condition in the CycSAT attack can also result in a significant number of additional clauses for densely cyclic circuits.

bench	#g	#I/O	CL dim	eq-klen	XOR t(s)	CL t(s)
apex2	610	39/3	$2 \times 30 \times 36$	310	93.2	TO
apex4	5360	10/19	$11 \times 30 \times 36$	1706	1072	TO
c1355	546	41/32	$2 \times 30 \times 36$	310	3715	TO
c1908	880	33/25	$2 \times 30 \times 36$	310	624	TO
c2670	1193	157/64	$3 \times 30 \times 36$	465	TO	TO
c3540	1669	50/22	$4 \times 30 \times 36$	620	109	TO
c432	160	36/7	$1 \times 30 \times 36$	155	1.84	TO
c499	202	41/32	$1 \times 30 \times 36$	155	5.96	TO
c5315	2307	178/123	$5 \times 30 \times 36$	775	1011	TO
c7552	3512	206/107	$8 \times 30 \times 36$	1240	TO	TO
c880	386	60/26	$1 \times 30 \times 36$	155	0.82	TO
dalu	2298	75/16	$5 \times 30 \times 36$	775	308	TO
des	6473	256/245	$13 \times 30 \times 36$	2016	427	TO
i4	338	192/6	$1 \times 30 \times 36$	155	18.1	TO
i7	1315	199/67	$3 \times 30 \times 36$	465	7.08	TO
i8	2464	133/81	$5 \times 30 \times 36$	775	20.2	TO
seq	3519	41/35	$8 \times 30 \times 36$	1240	335	TO

Table 2: ISCAS and MCNC benchmark circuits locked with cross-lock and XOR/XNOR obfuscation under the CycSAT attack.

Benchmark Circuit Tests: We performed cross-lock on a set of ISCAS and MCNC benchmarks from [20] as seen in Table 2. We used one 30×36 cross-bar for every 500 gates in the circuit. Hence, the *des* benchmark ends up with 13 cross-bars inserted. Note that with banking, the final chip needs 30 + 36 + 13 physical wires to configure the 30.36.13 PVIAs sequentially for this benchmark. We used the equivalent key-length times the number of cross-bars for XOR/XNOR obfuscation. As can be seen from Table 2 none of the cross-locked netlist are deobfuscated within the 3 hour deadline, while almost all XOR/XNOR lockings were defeated with the exception of the c2670 benchmark which has an internal AND-tree and the c7552 which overwhelms the attack with 1240 XOR/XNOR key bits. As for AppSAT resiliency, both schemes go through a rapid drop in error value after a certain number of iterations. However, the densely locked circuits that resist the exact attack, do not reach this many iterations within the given deadline.

5.2 Physcial Design

We also performed a proof-of-concept physical design flow. The Synopsys generic 32/28nm library and design kit was used for synthesis and place-and-route. Synopsys Design-Compiler (DC) was used for synthesis and IC-Compiler (ICC) was used for place-and-route. We designed the PVIA cells in Custom-Designer between the M2 and M3 layers and added them to the design as physical-only cells. Existing EDA tools lack a standard flow for automatic best-possible placement and routing of a large number of PVIAs. Hence we manually insert the PVIAs in a grid and try to push standard cells away from under this PVIA grid and perform placement. Then the automatic routing was successful in routing the cross-bar with a grid-like network as seen in Figure 4 without DRC violations. If DRC violations occur the utilization rate can be lowered until they are resolved which was not needed for the single cross-bar layouts reported in Table 3.

For power and delay characterization since the locked circuits are cyclic and can create issues for power/delay calculation routines, we take the acyclic netlist and manually add the additional capacitance due to the PVIAs (chosen as 0.3ff which is half of the minimum inverter capacitance in our technology) and nets to the netlist and calculate power/delay which is reported in Table 3. We compare these results with post-route XOR/XNOR locking with

$n \times n$ CL	original			15					25					35				
circuit	a(μm^2)	p(μW)	d(ns)	aX	pX	pCL	dX	dCL	aX	pX	pCL	dX	dCL	aX	pX	pCL	dX	dCL
c432	176.9	12.9	1.2	172.7	102.2	3.2	91.9	0.0	288.6	209.1	6.7	120.3	5.7	421.1	308.2	34.1	135.0	39.0
c499	526.6	50.9	0.9	40.0	13.5	2.1	58.9	2.2	76.7	47.3	4.8	67.8	4.4	119.7	88.0	11.7	93.3	6.7
c880	390.1	27.1	1.2	62.1	24.4	1.3	47.9	0.0	125.9	75.9	11.3	74.4	3.4	197.9	154.6	7.0	81.2	6.0
c1355	541.6	60.3	0.9	46.6	13.8	0.6	114.8	2.3	97.1	26.1	4.0	137.5	4.5	152.4	56.6	7.7	225.0	6.8
c1908	499.4	47.6	1.3	59.0	8.0	0.3	65.4	0.0	110.8	43.0	10.1	81.1	2.4	165.3	81.0	12.5	103.9	3.1
c2670	664.6	48.7	1.4	49.4	10.7	1.7	37.1	0.0	95.3	43.7	3.2	60.1	0.0	135.0	74.3	22.3	90.9	0.0

Table 3: Place and route results. (a, p, d)X/CL: area, power and delay overhead percentage for XOR/XNOR / cross-lock.

the equivalent key-length. While cross-lock inserts no additional gates to the design it incurs overhead to the entire chip through the programming circuitry that is shared among several cross-bars. An XOR/XNOR locking also requires external key-programming circuitry which is typically a scan-chain of DFFs as long as the key-length not included in Table 3. A detailed comparison of these two key-programming approaches and multi-crossbar layouts is left for future work. However, it is clear that scan-chains will be much larger for such large key-lengths compared to a banked cross-lock implementation.

6 CONCLUSION

In this paper we presented a layout-level dense interconnect locking scheme based on cross-bar architectures that has the potential to create high error rate and SAT resiliency simultaneously while suppressing overhead with layout and circuit level techniques. Chip-level overhead analysis, programming architecture optimization, reliability analysis, and interconnect locking schemes with the highest algebraic complexity are important topics of future research.

ACKNOWLEDGEMENT

This work was partially supported by DARPA (FA8650-18-1-7822).

REFERENCES

[1] Shuai Chen, Junlin Chen, Domenic Forte, Jia Di, Mark Tehranipoor, and Lei Wang. 2015. Chip-level anti-reverse engineering using transformable interconnects. In *Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*. IEEE, 109–114.

[2] Ronald P Cocchi, James P Baukus, Lap Wai Chow, and B Jiangyun Wang. 2014. Circuit Camouflage Integration for Hardware IP Protection. In *Proc. IEEE/ACM Design Automation Conf.*

[3] Actel Corp. 2002. Design Security in Nonvolatile Flash and Antifuse FPGAs. (2002). Actel Corp. Technical Report on QuickLogic FPGAs, http://www.actel.com/documents/DesignSecurity_WP.pdf.

[4] Franck Courbon, Sergei Skorobogatov, and Christopher Woods. 2016. Reverse Engineering Flash EEPROM Memories Using Scanning Electron Microscopy. In *Int. Conf. on Smart Card Research and Advanced Applications*. Springer, 57–72.

[5] Mohamed El Massad, Siddharth Garg, and Mahesh V Tripunitara. 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.. In *Network and Distributed System Security Symposium (NDSS)*.

[6] Stephen E Greco. 2011. Integrated circuit fuse. (Nov. 8 2011). US Patent 8,053,862.

[7] Óscar Iglesias-Freire, Miriam Jaafar, Eider Berganza, and Agustina Asenjo. 2016. Customized MFM probes with high lateral resolution. *Beilstein journal of nanotechnology* 7 (2016), 1068.

[8] Philipp Jovanovic and Martin Kreuzer. 2010. Algebraic attacks using SAT-solvers. In *Conference on Symbolic Computation and Cryptography*. 7.

[9] Shahrzad Keshavarz and Daniel Holcomb. 2017. Threshold-based Obfuscated Keys with Quantifiable Security against Invasive Readout. *CoRR* abs/1708.07150 (2017). arXiv:1708.07150 http://arxiv.org/abs/1708.07150

[10] Meng Li, Kaveh Shamsi, Travis Meade, Zheng Zhao, Bei Yu, Yier Jin, and David Z. Pan. 2016. Provably Secure Camouflaging Strategy for IC Protection. In *Proc. Int. Conf. on Computer Aided Design*. 28:1–28:8.

[11] Heiko Lohrke, Shahin Tajik, Christian Boit, and Jean-Pierre Seifert. 2016. No Place to Hide: Contactless Probing of Secret Data on FPGAs. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 147–167.

[12] Mohamed El Massad, Siddharth Garg, and Mahesh Tripunitara. 2017. Reverse Engineering Camouflaged Sequential Integrated Circuits Without Scan Access. *CoRR* abs/1710.10474 (2017). arXiv:1710.10474 http://arxiv.org/abs/1710.10474

[13] Satwik Patnaik, Mohammed Ashraf, Johann Knechtel, and Ozgur Sinanoglu. 2017. Obfuscating the Interconnects: Low-Cost and Resilient Full-Chip Layout Camouflaging. *CoRR* abs/1711.05284 (2017). arXiv:1711.05284 http://arxiv.org/abs/1711.05284

[14] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. 2013. Security Analysis of Integrated Circuit Camouflaging. In *Proc. ACM Conf. on Computer & Communications Security.*

[15] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *Proc. Design, Automation and Test in Europe (DATE '08)*. 1069–1074.

[16] SEMI. 2012. IP Challenges for the Semiconductor Equipment and Materials Industry. (2012). [Online] http://www.semi.org/sites/semi.org/files/docs/2012_IP_White_Paper.pdf.

[17] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. AppSAT: Approximately Deobfuscating Integrated Circuits. In *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*. 46–51.

[18] Kaveh Shamsi, Meng Li, Travis Meade, Zheng Zhao, David Z. Pan, and Yier Jin. 2017. Cyclic Obfuscation for Creating SAT-Unresolvable Circuits. In *Proc. IEEE Great Lakes Symp. on VLSI*. 173–178.

[19] Yuanqi Shen and Hai Zhou. 2017. Double DIP: Re-Evaluating Security of Logic Encryption Algorithms. In *Proc. IEEE Great Lakes Symp. on VLSI*. ACM, 179–184.

[20] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*. IEEE, 137–143.

[21] P. Subramanyan, N. Tsiskaridze, Wenchao Li, A. Gascon, Wei Yang Tan, A. Tiwari, N. Shankar, S.A. Seshia, and S. Malik. 2014. Reverse Engineering Digital Circuits Using Structural and Functional Analyses. 2, 1 (2014), 63–80.

[22] Berkeley Logic Synthesis and Verification Group. 2016. ABC a System for Sequential Synthesis and Verification. (2016). [Online] http://people.eecs.berkeley.edu/~alanmi/abc/.

[23] Xifan Tang, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. 2014. A high-performance low-power near-Vt RRAM-based FPGA. In *Field-Programmable Technology (FPT), 2014 International Conference on*. IEEE, 207–214.

[24] Randy Torrance and Dick James. 2009. The state-of-the-art in IC reverse engineering. In *Proc. Int. Conf. on Cryptographic Hardw. and Embed. Systems*. Springer, 363–381.

[25] Yang Xie and Ankur Srivastava. 2016. Mitigating sat attack on logic locking. In *Proc. Int. Conf. on Cryptographic Hardw. and Embed. Systems*. Springer, 127–146.

[26] Yufeng Xie, Xiaoyong Xue, Jianguo Yang, Yinyin Lin, Qingtian Zou, Ryan Huang, and Jingang Wu. 2016. A logic resistive memory chip for embedded key storage with physical security. *IEEE Trans. on Circuits and Systems II* 63, 4 (2016), 336–340.

[27] Xiaolin Xu, Bicky Shakya, Mark M Tehranipoor, and Domenic Forte. 2017. Novel Bypass Attack and BDD-based Tradeoff Analysis Against all Known Logic Locking Attacks. In *Proc. Int. Conf. on Cryptographic Hardw. and Embed. Systems*. 189–210.

[28] Muhammad Yasin, Bodhisatwa Mazumdar, Jeyavijayan Rajendran, and Ozgur Sinanoglu. 2016. SARLock: SAT Attack Resistant Logic Locking. In *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*. 236–241.

[29] Muhammad Yasin, Bodhisatwa Mazumdar, Ozgur Sinanoglu, and Jeyavijayan Rajendran. 2016. Camoperturb: secure ic camouflaging for minterm protection. In *Proc. Int. Conf. on Computer Aided Design*. IEEE, 1–8.

[30] Muhammad Yasin, Bodhisatwa Mazumdar, Ozugr Sinanoglu, and Jeyavijayan Rajendran. 2017. Removal Attacks on Logic Locking and Camouflaging Techniques. Cryptology ePrint Archive, Report 2017/348. (2017). https://eprint.iacr.org/2017/348.

[31] Muhammad Yasin, Abhrajit Sengupta, M Ashraf, M Nabeel, J Rajendran, and Ozgur Sinanoglu. 2017. Provably-Secure Logic Locking: From Theory To Practice. In *Proc. ACM Conf. on Computer & Communications Security*. 1–1.

[32] Hai Zhou. 2017. A Humble Theory and Application for Logic Encryption. Cryptology ePrint Archive, Report 2017/696. (2017). https://eprint.iacr.org/2017/696.

[33] Hai Zhou, Ruifeng Jiang, and Shuyu Kong. 2017. CycSAT: SAT-Based Attack on Cyclic Logic Encryptions. (2017). Cryptology ePrint archive, Report 2017/626, [Online] https://eprint.iacr.org/2017/626.pdf.