Time Domain Sequential Locking for Increased Security

Kyle Juretus and Ioannis Savidis

Department of Electrical and Computer Engineering
Drexel University
Philadelphia, Pennsylvania 19104
{kjj39@drexel.edu, isavidis@coe.drexel.edu}

***Abstract*—In this paper, the state space of an integrated circuit (IC) is used to increase the security of an IC against a variety of threats including intellectual property theft, IC counterfeiting, and IC overproduction. Hidden state transitions, state dependent keys, and temporal based transitions are implemented as a means to combat probing style attacks, such as the SAT attack. SPICE simulations are performed on modified state machines to characterize the overhead of implementing the three techniques in a circuit. Implementing temporal based transitions increases the area of the circuit by 68.42%, the power by 43.17%, and did not impact circuit delay. However, increasing the circuit size significantly reduces the overhead of state space encryption. For example, encrypting two registers in the s15850 ISCAS89 benchmark circuit resulted in an area overhead of 0.026%, presenting a low overhead means of securing sequential logic.**

I. Introduction

Fabrication facilities capable of producing integrated circuits (ICs) in advanced technology nodes require a multi-billion dollar investment [1], which has led to the rise of commercial foundries to manufacture ICs [2]. In addition, IC design firms are incorporating third-party intellectual property (IP) to reduce costs and enhance functionality [2]. Many traditional commercial companies have shifted from a once vertical IC flow to a horizontal design flow with a variety of untrusted entities including third-party IP vendors, fabrication facilities, test facilities, and end-users. Concerns over IP theft, counterfeiting and overproduction of ICs, and the insertion of harmful circuit modifications (hardware Trojans) all arise from the introduction of untrusted third-parties into the IC design flow. The challenge for creating secure circuits then becomes protecting against the wide variety of possible threats, including denial of service, theft of information, and/or corruption of functionality [3].

An area of research that addresses the security risks of IP theft, IC counterfeiting, IC overproduction, and hardware Trojan insertion is logic encryption/locking [4]. Logic encryption adds additional circuitry (key gates) to an IC to hide the functionality of the circuit from an adversary. Without the application of the correct input key, the IC does not function as intended. Essentially, a malicious foundry no longer possesses all of the information to reverse engineer the entire design, making it more difficult to steal the IP, counterfeit the IC, produce extra ICs, and even insert hardware Trojans. However, the SAT based attack presented in [5] circumvents the security provided by logic encryption and, therefore, requires the development of novel measures to defend against the wide variety of threats facing ICs.

In this paper, the change in the state space of a circuit over time is used to increase circuit security. Hidden transitions that are frequency dependent, state dependent keys, and temporally dependent transition keys are presented as methodologies to increase security against SAT based attacks. An overview of the assumed threat model and a description of the SAT attack is provided in Section II. Prior related work on sequential logic encryption is presented in Section III. The implementation and characterization of frequency based transitions, state dependent keys, and temporally dependent transitions into state machines is described in Section IV. Finally, concluding remarks are provided in Section V.

II. Threat Model

The assumed threat model includes an attacker who has access to an activated IC that produces correct input-output (IO) pairs, similar to the threat model for the SAT attack [5]. In addition, the attacker has reverse engineered the locked netlist from the integrated circuit layout, which allows the attacker to create a set of miter formulae that represent the circuit for analysis using satisfiability (SAT) algorithms and the correct IO pairs determined from an active IC. In this work, it is assumed that the output from the scan-chains used for circuit verification is not available to the adversary, as exposing the state transitions and logic to the adversary through the activated IC makes securing the locked netlist virtually impossible. Essentially, an adversary with scan-chain access has the opportunity to break many low complexity circuits generated from the partitioning of the IC for formal verification instead of the large complexity of analyzing the entire active circuit.

A. SAT Attack Vulnerability

Logic encryption relies on the complexity of the circuit to create a large unknown logic space that renders a brute force attack infeasible. The process of ensuring the key is correct for all IO pairs was assumed secure, as the number of IO pairs and key combinations increases exponentially with each additional bit and becomes infeasible to exhaustively test. However, the SAT attack demonstrated that with the correct IO pairs from an activated IC, the key used for logic encryption is extracted

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

for 95% of the ISCAS85 benchmark circuits within 10 hours [5]. Testing of every IO pair with a given key is no longer required as the addition of a single IO constraint to the SAT solver prunes a significant amount of the key space, weakening the perceived security of logic encryption.

B. Preventing the SAT Attack

The SAT attack is a probing based attack that requires access to the key signals and the known possible values of the key to significantly reduce the complexity of the circuit logic space [5]. Countering the SAT attack requires circuit information that is not leaked by reverse engineering the encrypted IC. A method is proposed that shifts state machine functionality into the time domain with a key dependency, generating circuit information that only the IC developer possesses. In addition, the pruning of incorrect key sequences is limited as the correct key may also be pruned if applied in an incorrect state.

III. Related Work

The sequential logic encryption introduced in HARPOON [6] partitions the state machine into an obfuscated and functional mode. An example of finite state machine (FSM) partitioning is shown in Fig. 1, which only enters the correct operation after the enabling key sequence of P0, P1, and P2 is applied. In addition, *black hole* states, which are never functionally exited, or *gray hole* states, which are very hard to functionally exit, are introduced to a state machine to further deter adversaries from trying to reverse engineer the IC [7].

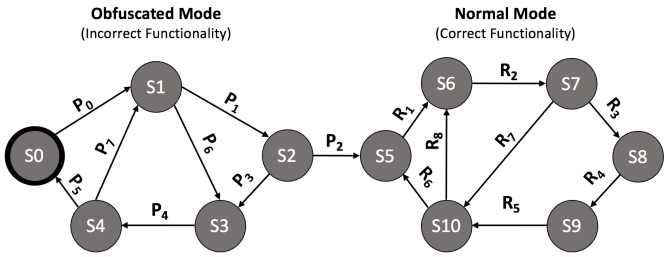


Fig. 1: Modified FSM with an enabling sequence of P0, P1, P2 required to enter normal circuit operation [6].

State machine partitioning into an obfuscated and functional mode is also described in [8]. However, a code-word is generated from state transitions during the obfuscated mode that results in incorrect state transitions during functional mode if the applied code-word is incorrect. There are security concerns in the partitioning of the state machine as described in [9], including vulnerabilities to FSM extraction [10] and fault-injection attacks [11]–[13].

The SAT attack is also applicable to security measures implemented on a FSM that is unrolled n times. The n unrollings provide the SAT solver with context for n FSM transitions, allowing for the elimination of keys similar to the description provided in Section II-A. The work in [9] removes the partitioning of the state machine in favor of adding incorrect state transitions during normal operation of the circuit based on an applied key. The insertion of incorrect state transitions, or edges, makes it more challenging for the successful implementation of the SAT attack as the number of times the FSM must be unrolled increases. The work in [14] also aims to eliminate partitioning of the FSM into obfuscated and functional modes by utilizing dynamic state deflection based on an applied key vector.

The objective of the work described in this paper is to mask circuit information from an adversary that has already reverse engineered the IC by adding temporally dependent keys. The work in [15] explores the use of timing as a means to secure circuits as well, but adds a key that is vulnerable to probing, which exposes all of the functional design information to reverse engineering. Instead, this work masks circuit information by requiring a time domain dependence that is only available to the IC designer and limits the efficacy of a SAT attack to decrypt an IC.

IV. Adding FSM Time Dependencies

A means to 1) add hidden transitions to a state machine based on the frequency of an IC, 2) require different keys in different circuit states via state dependent keys, and 3) add temporally dependent transitions are described in this section.

A. Frequency Based Hidden State Transitions

The first technique modifies the FSM to create hidden state transitions that are not present within the logic of the FSM. The original state machine is shown in Fig. 2.

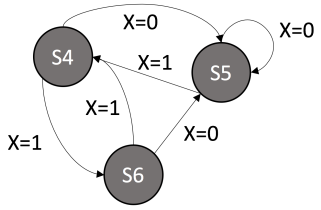


Fig. 2: Original state machine with single input *X*.

Altering the original state machine to a partitioned state machine utilizing a hidden state transition from *S3* to *S4* is developed, as shown in Fig. 3. A logical representation of the transition is no longer available to adversaries. Therefore, FSM recovery and other techniques that exploit state reachability are not able to discern the state transition. The states *S7* and *S8* are also partitioned from the original circuit, creating the appearance of four distinct state machines to the FSM recovery tools. Only the IP developer knows that there is a transition from *S3* to *S4* at a frequency of 5 GHz created by a strategic timing based glitch in the circuit. In addition, if the 5 GHz frequency is maintained for too long, in this case three clock cycles, the design transitions to *S8*. In this circuit, *S8* serves as a black-hole state that is not exited. However, other options including entering another state machine partition or restarting the sequence are possible.

Operation of the implemented FSM with the hidden states was verified through SPICE simulation. The results are shown

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

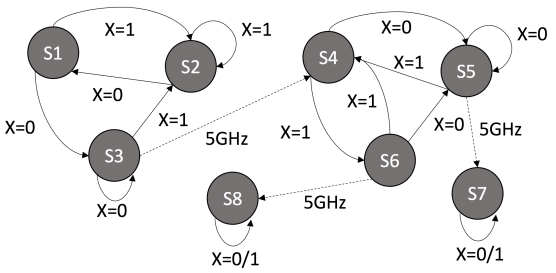
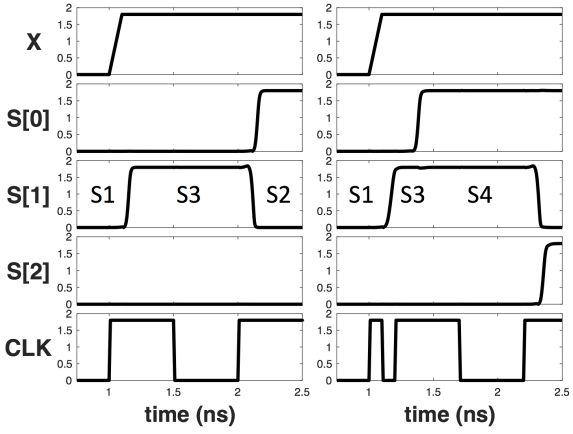


Fig. 3: State machine with hidden transitions from *S3* to *S4*, *S6* to *S8*, and *S5* to *S7* that activate at 5 GHz. A single input *X* is applied to the circuit.

in Fig. 4b and indicate that while in *S3*, when the frequency is increased to 5 GHz, the circuit transitions to *S4* (*S1* to *S3* to *S4*). However, if the frequency is unaltered, as shown in Fig. 4a, the circuit stays in *S3* and transitions to *S2* on the next clock edge.

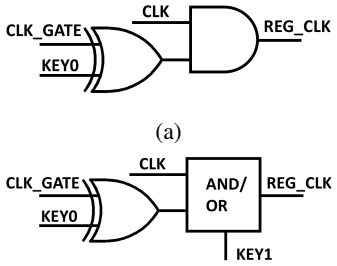
Including hidden states in the state machine, as shown in Fig. 3, incurs an overhead of 97% in area, 44.5% in power, and 32.1% in delay, calculated by synthesizing the state machine with and without a hidden state present. Increasing circuit size substantially reduces the overhead since the required modifications to the state space are relatively independent of the size of the circuits as each additional flip-flop expands the number of states exponentially. In addition, many circuits include *don't care* states that can be used for hidden state transitions, requiring only a small increase in combinational logic that is carefully selected to avoid the critical path.

If altering the clock frequency directly is undesired, the topologies shown in Fig. 5 also create hidden transitions. The first topology, shown in Fig. 5a, gates the clock and applies an XOR to the clock gating control line. As a result, the state of a given register is held at any given time and state when *KEY0* is *0*, allowing for hidden transitions to occur. The toplogy depicted in Fig. 5b allows for more flexibility through the use of an AND/OR gate to implement clock gating [16], [17]. The second topology allows for the clock to not only be held low



(a) (b)

Fig. 4: Simulation of the state machine with (a) no frequency based state transition, and (b) a temporary increase in clock frequency to enable a hidden state transition to *S4*.



(a)

(b)

Fig. 5: Hidden transition circuit topologies: (a) Clock signal gated with an AND gate and an XOR control of the clock gating, and (b) clock gated with an AND/OR [16], [17] and with an XOR control of the clock gating.

at the register input but also to insert clock pulses when the clock is logic low by setting *KEY1* to *1*.

Note that fault injection attacks can still be utilized to enter the functional state machine from the obfuscated partition (*S1, S2, S3*) shown in Fig. 3. However, methodologies that discourage fault injection attacks are described in Sections IV-B and IV-C that limit malicious state transitions from bypassing added security features.

B. State Dependent Key Values

The use of sequential elements in a circuit allow for the creation of keys that are state dependent. As the SAT attack selects a single key sequence, the attack must now divide the circuit into a subset of states to verify the key and also determine the proper temporal change in key values. Essentially, all key combinations in all states must be verified since an incorrect key for one IO pair no longer implies a pruning of a set of keys, as is done with the SAT attack. The incorrect key value may be required in a different state or with a different IO pair to enable correct circuit functionality.

An example of adding state dependent keys to a circuit is shown in Fig. 6, which also includes the hidden states discussed in Section IV-A. In addition, all of the original states are re-encoded with a randomly assigned new state bit to ensure that there is no imbalance in the switching probability of the newly added register in comparison to the original state machine registers. Notice that when *K0* is *1*, many of the transitions from the original state machine now return to the obfuscated mode, which implies that a simple fault injection is difficult to execute without knowledge of the key.

Implementing the modifications to the state machine incurs a significant overhead in terms of power, performance, and area for the small FSM used as an example. An increase of 457.89% in area, 881.23% in power, and 40.48% in delay was observed. However, a significant portion of the overhead is due to the required increase in switching activity on the additionally inserted flip-flop. If the comparison is completed between the partitioned FSM with and without the state dependent key values, there is an increase of 99.99% in area, 117.91% in power, and 25.53% in delay. As with the hidden

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

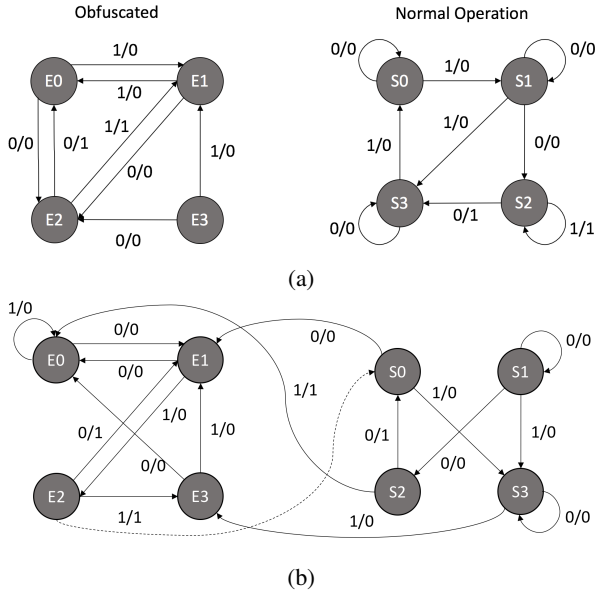


Fig. 6: State dependent key circuit topology for (a) K0 = 0, correct key for functional mode, and (b) K0 = 1, correct key for obfuscated mode. The hidden transition, represented by a dotted line, only exists when K0 is 1 in the obfuscated mode.

state transitions, the overhead is significantly reduced as the size of the circuit is increased.

There is also a trade-off between the number of state dependent key edges and the overhead in power, area, and delay as edge insertion allows for possible logical simplification. However, such an insertion strategy benefits an adversary who is aware of the insertion methodology by leaking information on the states that require an altered key. While increased complexity is added to the design, the added combinational logic also leaks information on which transitions are key dependent and possibly even the correct transition itself. A verification procedure is, therefore, required to assure no additional information is leaked.

C. Temporal Based Transitions

The original FSM is modified by replacing original transitions with incorrect transitions. For the FSM shown in Fig. 2, the transition *S5* to *S4* for *X = 1* is replaced with *S5* to *S6* and the transition *S5* to *S5* for *X = 0* is replaced with the transition *S5* to *S4*. The reverse engineered state machine now appears as shown in Fig. 7. The only addition to the circuit is the inclusion of the structures shown in Fig. 5 at the clock input of one or more registers within the design. For this specific example, the topology in Fig. 5a is chosen and is applied to both state registers. The correct functionality, when in *S5*, is achieved by setting *K0* to *0* when *X* is *1* and *K1* to *0* when *X* is *0*, as shown by the thick red transitions in Fig. 7.

After reverse engineering the circuit, the adversary no longer observes the original functionality of the FSM, preventing FSM recovery tools from determining the original transitions. Fault injection attacks are also no longer feasible as the state machine is not partitioned into obfuscated and functional regions. Additionally, the temporally dependent key does not

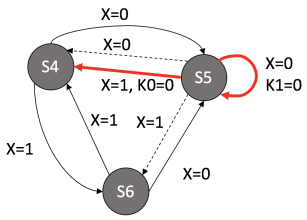


Fig. 7: Altered FSM that requires a temporally dependent key. Dotted lines represent incorrect edges that are corrected by asserting K0 or K1 in *S5* (correct transitions are shown as thick red lines).

allow for the unrolling of the state space and, therefore, limits the efficacy of the SAT attack as a key sequence no longer is pruned if it is invalid during a certain set of state transitions. Rather, the entire state space must be validated to ensure the correct key is obtained.

The implementation of the temporally dependent state machine does not incur any performance overhead as the combinational paths do not require any additional elements. The area and power overhead of, respectively, 68.42% and 43.17% for adding the XOR based clock gating shown in Fig. 5a is large for the implemented FSM. As the logical depth and amount of combinational logic increases in the circuit, the overheads drop significantly. For example, the area overhead of encrypting two registers is 5.08% for the ISCAS89 s298 (75 gates) benchmark circuit and 0.026% for the s15850 (3448 gates).

V. Conclusions

Hidden transitions of the state machine based on 1) the frequency of an IC, 2) utilizing different key values depending on the FSM state, and 3) time-dependent keys to enable correct circuit functionality were presented in this paper as means to increase IC security against intellectual property theft, IC counterfeiting, and IC overproduction. There is an overhead of 97% in area, 44.5% in power, and 32.1% in delay to implement frequency based transitions. The implementation of state dependent keys required an increase of 99.99% in area, 117.91% in power, and 25.53% in delay. The temporally dependent state machine resulted in no performance overhead, but increased area by 68.42% and power by 43.17%. A significant reduction in both area and power overheads is seen as the circuit size increases, with a decrease in area overhead to 0.026% for the 3448 gate ISCAS s15850 benchmark circuit when encrypting two registers utilizing the temporally dependent state space encryption methodology. The use of the developed techniques drastically increases the difficulty of implementing probing based style attacks, such as the SAT attack, and provides a low cost means of securing an IC.

Acknowledgements

This research is supported in part by the Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a, Drexel Ventures Innovation Fund, and the National Science Foundation under Grant CNS-1648878.

References

[1] DigiTimes, "Trends in the Global IC Design Service Market," http://www.digitimes.com/news/a20120313RS400.html?chid=2, March 2012.

[2] IARPA, "Trusted Integrated Chips (TIC) Program," IARPA-BAA-11-09, October 2011.

[3] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design and Test of Computers*, Vol. 27, No. 1, pp. 10–25, February 2010.

[4] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," *Proceedings of the IEEE/ACM Design, Automation and Test in Europe Conference*, pp. 1069–1074, October 2008.

[5] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 137–143, May 2015.

[6] S. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 28, No. 10, pp. 1493–1502, October 2009.

[7] Y. Alkabani and K. Farinaz, "Active Hardware Metering for Intellectual Property Protection and Security," *Proceedings of the USENIX Security Symposium*, pp. 291–306, August 2007.

[8] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall, "Interlocking Obfuscation for Anti-tamper Hardware," *Proceedings of the Cyber Security and Information Intelligence Research Workshop*, pp. 8:1–8:4, January 2013.

[9] T. Meade, Z. Zhao, S. Zhang, D. Pan, and Y. Jin, "Revisit Sequential Logic Obfuscation: Attacks and Defenses," *Proceedings of the IEEE International Conference on Circuits and Systems*, pp. 1367–1370, May 2017.

[10] T. Meade, S. Zhang, and Y. Jin, "Netlist Reverse Engineering for High-level Functionality Reconstruction," *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 655–660, January 2016.

[11] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures," *Proceedings of the IEEE*, Vol. 100, No. 11, pp. 3056–3076, November 2012.

[12] C. H. Kim and J. J. Quisquater, "Faults, Injection Methods, and Fault Attacks," *IEEE Design and Test of Computers*, Vol. 24, No. 6, pp. 544–545, November 2007.

[13] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger, and A. Muehlberger, "High Level Fault Injection for Attack Simulation in Smart Cards," *Proceedings of the Asian Test Symposium*, pp. 118–121, November 2004.

[14] J. Dofe and Q. Yu, "Novel Dynamic State-Deflection Method for Gate-Level Design Obfuscation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. PP, No. 99, pp. 1–13, April 2017.

[15] Y. Xie and A. Srivastava, "Delay Locking: Security Enhancement of Logic Locking Against IC Counterfeiting and Overproduction," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 9:1–9:6, June 2017.

[16] K. Juretus and I. Savidis, "Reduced Overhead Gate Level Logic Encryption," *Proceedings of the IEEE/ACM Great Lakes Symposium on VLSI*, pp. 15–20, May 2016.

[17] K. Juretus and I. Savidis, "Reducing Logic Encryption Overhead Through Gate Level Key Insertion," *Proceedings of the IEEE International Conference on Circuits and Systems*, pp. 1714–1717, May 2016.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE