Hardware Obfuscation using PUF-based Logic

James B. Wendt and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{jwendt, miodrag}@cs.ucla.edu

***Abstract*—There is a great need to develop universal and robust techniques for intellectual property protection of integrated circuits. In this paper, we introduce techniques for the obfuscation of an arbitrary circuit by using physical unclonable functions (PUFs) and programmable logic. Specifically, we introduce the notion of PUF-based logic which can be configured to be functionally equivalent to any arbitrary design, as well as a new architecture for wire merging that obfuscates signal paths exponentially. We systematically apply our techniques in such a way so as to maximize obfuscation while minimizing area and delay overhead. We analyze our techniques on popular benchmark circuits and show them to be resilient against very powerful reverse engineering attacks in which the adversary has knowledge of the complete netlist along with the ability to read and write to any flip-flop in the circuit.**

I. Introduction

The state of the art in reverse engineering is so advanced that industrial chips with a billion transistors can be reverse engineered in a matter of a few weeks. There are even a number of dedicated companies that on a regular basis reverse engineer new industrial integrated circuits (ICs). Hardware obfuscation is an approach that aims to prevent the possibility of reverse engineering ICs. Recently, several conceptually new and interesting approaches in hardware obfuscation have been proposed and demonstrated. Our goal is to advance the state of the art in hardware obfuscation by presenting a technique in which reverse engineering does not only require that each transistor be fully characterized in terms of its delay, but also enables configurable obfuscation in the sense that each integrated circuit for a given design is obfuscated in a unique way.

The starting point for our approach is the use of physical unclonable functions (PUFs) for our conceptually new task. A PUF is a hardware device that has a complex but definite mapping from inputs to outputs that is practically impossible to reverse engineer. Furthermore, the device cannot be physically cloned. Currently, most PUF designs achieve unclonability and complexity by exploiting silicon manufacturing variability that manifests as variations in circuit element properties, such as delay and leakage energy. The complete functionality of the PUF is only known if the entire input-output (challenge-response) table is enumerated or each gate's internal properties are fully characterized.

We have developed two techniques that exploit the inherent randomness of the PUF in order to obfuscate a circuit. The first technique obfuscates an arbitrary piece of random logic by replacing it entirely. Specifically, we implement the pertinent piece of logic using a physical unclonable function in conjunction with a supporting small piece of configurable fabric, such as a field-programmable gate array (FPGA). The purpose of the PUF is to obfuscate the circuit, since its functionality is known only to the designer and trusted manufacturer, while the purpose of the programmable fabric is to implement the original piece of replaced logic using the PUF whose characterization is only known after fabrication.

The second technique hides circuit functionality by obfuscating signal paths in the interconnect network. Pairs of wires are merged together such that an attacker cannot determine which signal propagates along which path. In this scenario, the PUF is used to control these junctions in such a way that the original circuit functionality is simultaneously preserved and obfuscated. We employ these two techniques in such a way that delay constraints are satisfied while security is maximized for a user specified area and energy overhead.

The most important specification in security analysis is the specification of attacks and analysis of how specific techniques are resilient against these types of attacks. In our analysis we consider two extremely powerful attacks. In both attacks we assume the adversary has complete knowledge of the circuit netlist, but does not have access to individual gate properties such as dopant concentrations and channel lengths. The first attack assumes that an adversary can simultaneously observe all flip-flops in any clock cycle. The second attack is even more powerful and allows that the adversary can not only simultaneously observe all flip-flops but can also control every flip-flop in the design. We develop heuristics that minimize the effectiveness of these two attacks along with simulation based techniques that quantify the attack time effort required in order to break our security. We apply our techniques to a number of ISCAS'89 and ITC'99 benchmarks and successfully obfuscate their functionality with an overhead of up to 10% in area.

It is very important to note that our techniques for configurable obfuscation can also be used for many other security tasks. For example, when using configurable obfuscation it is much more difficult for an attacker to place an unnoticeable Trojan horse in the circuit since he is not aware of the circuit's complete design. Also, the PUF can be used as watermarking information for each and every IC produced of a particular design.

The remainder of this paper is organized as follows. First, we survey recent research and developments in the field of hardware security. Second, we provide a brief overview of the standard delay-based PUF model which we incorporate in our techniques. Next, we describe in detail our PUF-based logic architecture and signal path obfuscation techniques. We describe two types of attacks and present heuristics for configuration of both of our new architectures for reverse engineering prevention. And finally, we analyze our techniques in terms of security and overhead.

978-1-4799-6278-5/14/$31.00 ©2014 IEEE

II. Related Work

While there have been a number of efforts to produce systems in many technologies which are equivalent to PUFs, process variation has enabled the creation of practical and low cost PUFs. An MIT group developed the concept and a large number of silicon prototypes in several technologies which demonstrated their advantages and limitations [1]. PUFs have been used in a number of applications including device authentication, secrete key generation, anti-counterfeiting, key distribution, and secure key storage [2] [3]. Recently, the emergence of the digital PUF has enabled its direct integration into digital logic in both ASIC [4] [5] and FPGA systems [6] [7] [8] [9]. The use of emerging nanotechnologies for PUFs has introduced an entirely new security dimension which has yet to be implemented in traditional designs [10] [11]. There are several excellent surveys on the history and state of the art of PUFs [12] [13].

In the last quarter century numerous techniques for reverse engineering and layout reconstruction on silicon chips have been proposed and demonstrated [14] [15] [16] [17]. For example, both in academia and in industry, state of the art processors, like Intel's general purpose processors, have been reverse engineered. Two excellent summaries on the state of the art in reverse engineering have been presented by Chipworks [18] [19]. Other recent efforts in reverse engineering apply logic synthesis and formula verification techniques to functionally reconstruct significant percentages of logic circuitry [20] [21]. Other interesting and important reverse engineering efforts include imaging-based and side channel attacks [22] [23].

Hardware obfuscation is a task that aims to prevent IC reverse engineering. It can be divided into two broad groups. In the first, unique structures are added to ICs in such a way that only the designer of the circuit can enable and disable correct functional execution [24] [25]. There have also been efforts to obfuscate ICs in such a way that the physical structure of gates is difficult to deduce during reverse engineering because two or more types of gates differ in only ultra small details of implementation that cannot be easily deduced using state of the art reverse engineering techniques [26]. An NYU Poly research group has demonstrated the effectiveness of these techniques and quantified the overhead in terms of important design metrics [27]. Finally, in the late 90s there were a number of efforts to enable a designer's signature to be permanently written to hardware as proof of intellectual ownership [28].

Our work is a conceptually new application of PUFs for hardware obfuscation. While hardware obfuscation enabling and disabling techniques have been previously accomplished using PUFs, this is the first time that the PUF actually implements random logic in such a way that the functionality of the circuit is completely hidden. This new proposed technique is different from all previously proposed hardware obfuscation techniques because it reduces reverse engineering not just to the problem of identifying which gates are connected in which way, but also requires that the number of dopants in each transistor is accurately recovered (i.e. delay characterization), which is well beyond the feasibility of current reverse engineering attacks. Another important difference between state of the art obfuscation techniques and our new approach is that in each instance of an integrated circuit our approach results in a different type of obfuscation. Therefore, even if the attacker is successful in reverse engineering a particular chip he cannot validate the functional correctness of other fabricated chips that are obfuscated.

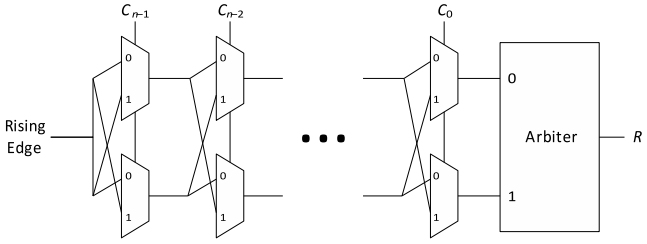


Fig. 1: Standard delay-based arbiter PUF [2].

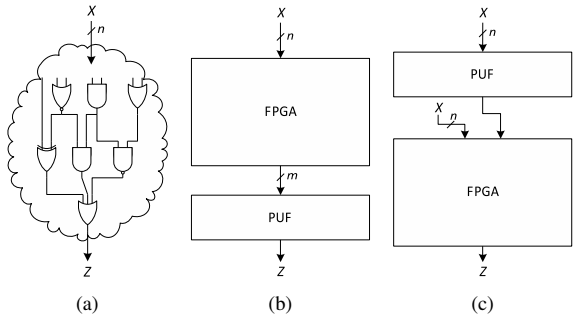


Fig. 2: Three architectures implementing the same functionality. (a) Arbitrary circuitry. (b) PUF-based logic using a preceding FPGA. (c) PUF-based logic using a preceding PUF.

III. Standard PUF Overview

Our obfuscation techniques employ the standard delay-based arbiter PUF originally designed by Suh et al. at MIT [2]. The phenomenon that enables its security and unclonability is process variation. Inherent randomness in manufacturing processes manifest as deviations in gate characteristics from nominal specifications even among designs fabricated on the same die. More specifically, each gate in the standard PUF is ultimately fabricated with different delays and cannot be controlled beyond a certain level of granularity.

The standard delay-based arbiter PUF takes advantage of these delay deviations using the architecture depicted in Figure 1. A challenge input of n bits, represented by the vector $\mathbf{C} = [C_0, C_1, \ldots, C_{n-1}]$ is applied, and a rising edge is sent through the PUF. The rising edge is split into two at the first junction of multiplexors. The path of each rising edge will then switch positions (top or bottom) depending on the challenge vector bit. In this example, a challenge bit of 1 will swap the paths, while a value of 0 will keep the paths propagating along their current line. The first path to arrive at the arbiter determines the output response, R, of the PUF.

For the purposes of circuit obfuscation, the functionality of the PUF must be characterized post fabrication and only by the designer or trusted manufacturer. Immediately following characterization by the trusted authority we remove the ability

for further direct characterization attempts. This can be done by either laser burning access wires or burning supporting fuses [29] [30].

Note that while we use the delay-based arbiter PUF because it is considered to be the standard PUF, our techniques are compatible with the majority of IC-based PUF designs.

It is known that the standard delay-based arbiter PUF can be susceptible to operating conditions (e.g. temperature variations, voltage fluctuations). However, this susceptibility is limited to specific challenge vectors whose corresponding PUF delay paths differ by extremely small amounts. Thus, fluctuations in operating conditions can change the outcome of the PUF response for these input vectors. We discuss how we handle these stability issues in Section IV-B.

IV. Arbitrary Logic Replacement

Our first technique obfuscates an arbitrary circuit by replacing it with PUF-based logic. PUF-based logic directly implements the original functionality of the replaced circuit while also hiding its functionality. We explore two potential architectures, as depicted in Figure 2. Since the functionality of the PUF is a byproduct of manufacturing processes, its characterization is not known until after fabrication. Once fabricated, the designer characterizes the PUF through special supporting hardware, then burns the pertinent access wires to disallow any subsequent unauthorized characterization. Now that the PUF's unique functionality is known only to the designer, the FPGA fabric is programmed using standard synthesis tools in order to implement the original replaced circuitry in conjunction with the corresponding PUF.

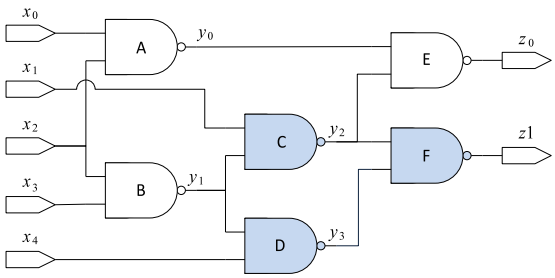
A. Programmable Fabric Configuration

Since each fabricated PUF is unique and unclonable, the supporting FPGA in the PUF-based logic architecture enables the replication of an arbitrary function. Once the PUF on a pertinent circuit has been characterized, the supporting FPGA fabric is configured accordingly given the known PUF mapping using standard synthesis tools.

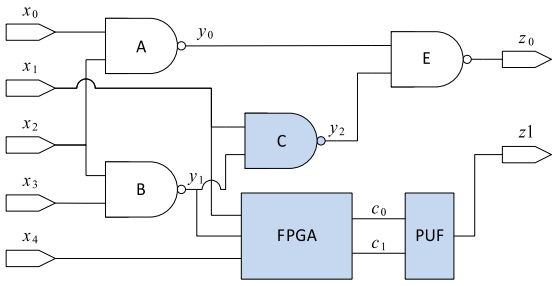
Figure 3 contains a motivational example in which we hide the functionality of the c17 circuit from the ISCAS'85 benchmark suite [31] by obfuscating a portion of the circuitry. The shaded portion in Figure 3a is the logic to be replaced and obfuscated. The resulting obfuscated architecture is depicted in Figure 3b.

Note that gates D and F are eliminated entirely while gate C remains despite participating in the computation of z_1. This is because gates D and F only affect z_1, while the output of gate C (y_2) is also required for the computation of z_0. Thus, we cannot eliminate C entirely from the circuit. Note that while retaining gate C does increase overhead, it does not compromise security in any way since an attacker cannot know with any certainty which remaining gates in the obfuscated circuit are included in the PUF-based logic cell, if any are at all.

We present an example challenge-response table for a small PUF in 3c. In this example, we assume that the challenge $\mathbf{C} = [1, 0]$ is unstable. As such, we program the FPGA so that it will not produce this output.



(a) Original circuit



(b) Obfuscated circuit

C_1	C_0	R
0	0	1
0	1	0
1	0	-
1	1	1

(c) PUF switching table

$$c_1 = y_1'$$
$$c_0 = y_1 x_1' x_4'$$

(d) FPGA implementation

Fig. 3: Motivational example of PUF-based logic replacing a portion of (a) the c17 circuit from the ISCAS'85 benchmark suite [31]. (b) Obfuscated circuit. (c) Example characterized PUF switching table. (d) FPGA implementation enabling the replaced circuit functionality in conjunction with the PUF.

The FPGA is synthesized such that for a particular input vector, $[x_4, y_1, x_3]$, it produces an input to the PUF that maps to the original correct output at z_1. One such specification is depicted in 3d. In this small example, the FPGA produces 3 potential output vectors which ultimately map z_1 to either 0 or 1. Without knowledge of the internal characteristics of the PUF, the mapping cannot be deduced. Of course, since this is a very small example, a brute force attack could easily enumerate all possible inputs to the PUF-based logic cell, however, we later demonstrate that this attack is infeasible for larger designs. Security can be further improved by randomizing the challenges outputted by the FPGA, especially for larger input spaces.

B. Stabilizing the Standard PUF

As previously mentioned, it has been observed that for some input vectors it is possible that a standard delay-based arbiter PUF produces unstable outputs. Specifically, there may exist challenge vectors which could potentially produce different responses depending on operating conditions such as temperature variations and voltage fluctuations. Hence, when employing the standard PUF, we specifically choose to use the

architecture in which the FPGA precedes the PUF, as depicted in Figure 2b. In this way, we can configure the FPGA to eliminate potential unstable inputs.

While the architecture in Figure 2c is not ideal for the standard delay-based arbiter PUF due to the issue of unstable inputs, if we employ a PUF that is stable for all inputs, then this architecture is a viable option and potentially preferable. This architecture provides an additional layer of security that is inherent in the design. By placing the FPGA after the PUF, the PUF's output is not directly known and thus it is more difficult for an adversary to attack without reverse engineering the FPGA inputs from its output first.

V. Signal Path Obfuscation

Our second obfuscation technique utilizes PUFs to obfuscate circuit functionality by directly obfuscating the interconnect network. Specifically, we combine pairs of wires in such a way that their paths are unknown. Figure 4 depicts our architecture for signal path obfuscation. Similar to the PUF-based logic case, after fabrication the pertinent PUFs are characterized and the inputs to each PUF are set such that they swap their corresponding wires according to the original circuit functionality.

This technique becomes very powerful when many wire swapping components are placed throughout the circuit in such a way that storage elements are affected by many potential swaps. For example, if the input to a single storage element is affected by k swapping components, then there exist 2^k possible configurations from which an attacker must determine the correct configuration in order to deduce the correct functionality of the circuit.

This obfuscation architecture should be placed such that the most number of flip-flops are affected the most number of times. We also take into consideration the additional delay overhead that comes with this architecture, and place these wire swapping components between gates with positive slack, so as not to affect the critical path.

It is important to note that our signal path obfuscation and PUF-based logic techniques are orthogonal and can be applied to an arbitrary circuit simultaneously.

VI. Attacks

We assume there exist two types of attacks on the PUF-based logic and signal path obfuscation techniques. In both attacks, we assume an adversary has complete knowledge of the design of the circuit, including the functionality of any pertinent programmable logic, but does not know the input-output mapping of any PUFs. In the first attack, we assume that the adversary has the powerful ability to both read and write to all flip-flops in the circuit. The second attack assumes that an adversary has the ability to read all flip-flops, but can only write to the primary inputs.

In the case of PUF-based logic, a successful attack is one in which all PUFs are characterized. In the first powerful attack, the security of our obfuscation technique relies solely on the attacker's ability to reverse engineer the PUF through application of a complete set of inputs. By reading each corresponding output, a complete characterization table can be

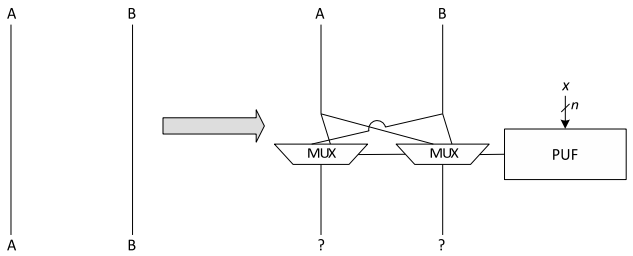


Fig. 4: Signal path obfuscation architecture for wire swapping. The input X can only be set correctly by the designer who knows the functionality of the PUF.

built. Thus, secure obfuscation relies on the size of the PUF, since the total input space grows exponentially with its size. We can make the task more difficult by using the architecture from Figure 2c in which the PUF precedes the FPGA. In this scenario, the attacker will not have direct access to the PUF output, but must instead reverse engineer its output values from the FPGA output.

In the case of the second type of attack, an adversary must intelligently apply inputs at each clock cycle such that he can indirectly apply as many inputs as possible to the PUF and measure the corresponding outputs. This attack is more difficult since the attacker cannot directly control all flip-flops, but instead must attempt to indirectly control them through the primary inputs. In this scenario we can make the reverse engineering task very difficult by placing the PUF in locations that are difficult to control indirectly. We discuss the specifics of these techniques in Section VII-A.

In the case of reverse engineering circuit functionality in the presence of signal path obfuscation, the attacker does not need to fully characterize all PUFs, but instead needs to determine the circuit-wide configuration of all wire swapping components (i.e. whether each individual wire pair is swapped or not). An attacker can test his configuration by applying a circuit input vector with known outputs and testing to see if the resultant outputs are as expected. However, even if the circuit outputs are as expected, the attacker has still not yet completely correctly reverse engineered the circuit since he may have only correctly solved those wire swapping components that received a 0 on one wire and a 1 on the other. For the cases in which both wires are the same value, the configuration is not yet confirmed. Furthermore, for some input vectors there is a possibility that particular wire values do not participate in the final output (i.e. don't-cares). For example, even if a wire swapping component swaps a 1 and a 0, the 1 might propagate to an AND gate containing another input whose value is 0. In this case, the swap had no affect on the final output.

Ultimately, an attacker will be forced to try a large portion, if not all, of the 2^k combinations of configurations, where k is the number of wire swappings affecting a single flip-flop. By testing all of these configurations on a single set of known inputs and outputs, most likely he will find a set of partially correct configurations. The set will only be partial due to the don't-cares and correlation scenarios mentioned above. However, with these known configurations, he can repeat the same steps using the next set of known inputs and outputs to slowly build a more complete configuration.

VII. Techniques

A. Logic Replacement

For the powerful attack case in which an adversary has write access to all flip-flops, the security of the PUF-based logic relies on the size of the PUF to create an exponential input space. Additionally, if we use a stable PUF, we employ the architecture from Figure 2c, in which the programmable fabric follows the PUF, to further prevent attacks by disabling direct access to the PUF output.

We prevent the second type of attack, in which an adversary can read all flip-flops but can only write to primary inputs, by replacing portions of circuitry with PUF-based logic in such a way that it is difficult for the attacker to set the PUF's inputs and thus increase his knowledge of the PUF's functionality by reading the corresponding outputs. Our placement criteria to accomplish this are the following:

- Place PUF-based logic where it is *affected by* many flip-flops, specifically by flip-flops which cannot be set directly by the attacker.
- Place PUF-based logic where it *affects* many flip-flops which cannot be set directly by the attacker.
- Place PUF-based logic where its inputs are highly correlated, thus making it very difficult for the attacker to build a large input-output table.

Since there are an exponential number of possibilities for selecting subcircuits for replacement with PUF-based logic, we simplify the task by performing multiple breadth first searches emanating from the input wires of flip-flops and traversing backwards through the circuit exploring potential PUF-based logic placements and measuring their security properties and overhead. In this way, we reduce the search space and still find many configurations that are both low in overhead and high in security. We discuss our results in detail in Section VIII.

B. Signal Path Obfuscation

For the case of signal path obfuscation, our techniques secure the obfuscated circuit functionality for both types of defined attacks. This is accomplished using the wire swapping architecture from Figure 4 and combining pairs of wires (i) that affect many flip-flops, (ii) that are affected by many flip-flops, (iii) whose inputs are correlated, and (iv) that are affected by previously assigned wire swapping components. By positioning wire swapping components between wires that affect and are affected by many flip-flops we ensure that wire swapping has a large and unpredictable impact on the circuit. By choosing pairs of wires that are highly correlated we ensure that reverse engineering remains difficult since if two wires consistently have similar values it is difficult to deduce whether or not they are being swapped. And finally, by placing wire swappings along paths in which previously assigned wire swapping components are installed we ensure exponential growth in the total number of possible configurations.

We iteratively assign wire swapping components between pairs of wires throughout the circuit according to a linear evenly weighted sum of these heuristics. Specifically, we consider (i) the union of flip-flops affected by pairs of wires as a fraction of the total flip-flops in the circuit, (ii) the union of flip-flops affecting pairs of wires as a fraction of the total flip-flops in the circuit, (iii) the coefficient of determination, R^2, between pairs of wires, and (iv) the union of wire swapping components preceding and affecting pairs of wires. Furthermore, we only consider pairs of wires which are not on the critical path and have positive slack. Thus obfuscating circuity functionality without overhead in terms of overall circuit delay.

VIII. Analysis

We analyze our PUF-based logic and signal path obfuscation techniques in terms of security and overhead by applying them to the circuits in the ISCAS'89 and ITC'99 benchmark suites [32] [33].

Figure 5 depicts results from our signal path obfuscation techniques on six example circuits. Specifically, we show the distributions of flip-flops that are affected by the labeled number of wire swappings on the x-axis. In all cases we successfully obfuscate a majority percentage of flip-flops using a very large number of wire swappings. This in turn creates a hugely exponential configuration search space for an attacker to reverse engineer. Note that wire swappings are applied only to wire pairs which have positive slack, thus ensuring that our final obfuscated circuit has no delay overhead.

Figures 6, 7, 8, and 9 depict the many PUF-based logic configurations that our techniques enumerate in the exponential search space for the six example circuits. Our heuristics find arbitrary portions of circuitry for PUF-based logic replacement in which there are a large number of inputs, large number of affected flip-flops, and are affected by a large number of flip-flops. These figures highlight the numerous PUF-based logic configurations that both satisfy our heuristics while simultaneously minimally impacting overhead.

Finally, we depict the security properties of our PUF-based logic obfuscation techniques in Figure 10. Each plot represents a single portion of arbitrary logic replaced with a single PUF-based logic architecture. We attack the resultant obfuscated circuitry by controlling the primary inputs and measuring the inputs and outputs at the pertinent PUF at each clock cycle. The figures depict the fractional number of correctly reverse engineered input-output mappings for the given PUF-based logic design as it relates to the pertinent PUF's number of inputs, placement in terms of depth in the circuit, number of affected flip-flops, and the number of flip-flops affecting the replaced logic. In each case, a linear increase in the corresponding heuristic causes an order of magnitude increase in difficulty of reverse engineering the circuit.

IX. Conclusion

We have presented new approaches for hardware obfuscation through the use of physical unclonable functions for actual random logic implementation such that the functionality of the replaced logic is completely hidden. Specifically, we presented two techniques, signal path obfuscation and the direct replacement of arbitrary logic using PUFs and reconfigurable logic. We provided algorithms and heuristics for the placement of each of the new techniques, applied them on a host of benchmark circuits, and demonstrated that they successfully obfuscate circuit functionality, impose minimal overhead in

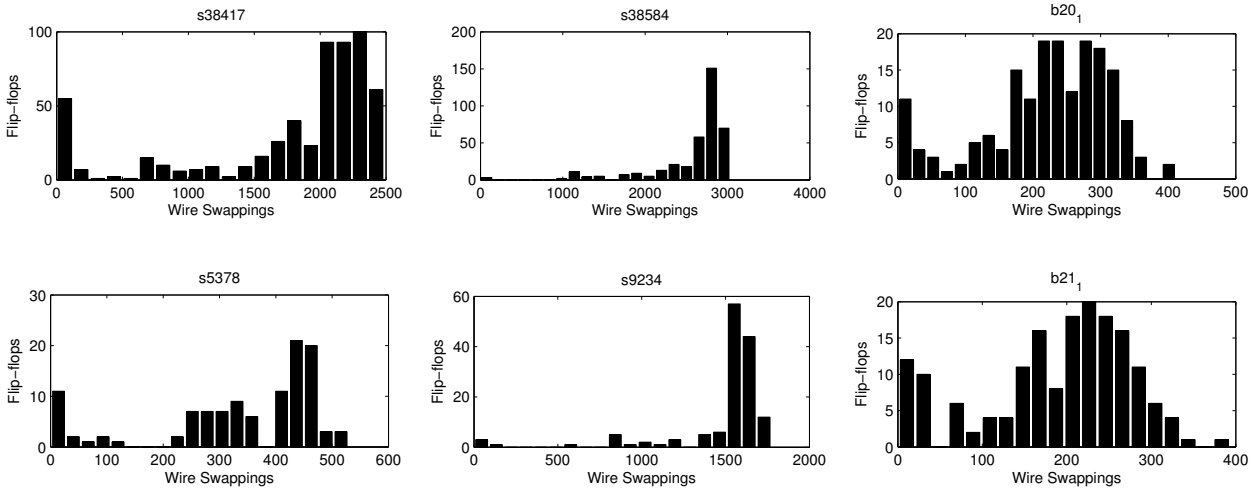


Fig. 5: Total number of flip-flops affected by the labeled number of wire swapping components.

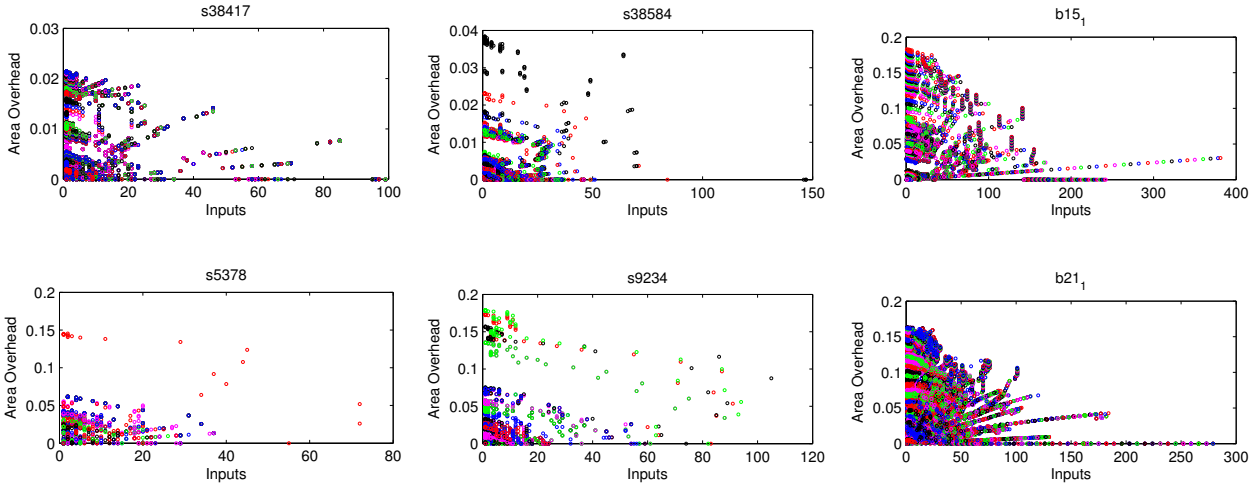


Fig. 6: Area overhead upon replacement of circuitry using PUF-based logic with the labeled number of inputs. The different colors represent the flip-flops whose inputs come from the individual PUF-based logic component.

terms of area and delay, and are robust against very powerful reverse engineering attacks.

ACKNOWLEDGEMENTS

This work was supported in part by the NSF under award CNS-0958369, award CNS-1059435, and award CCF-0926127, and by Samsung under award GRO-20130123.

REFERENCES

[1] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Computer and Communications Security (CCS)*, pp. 148–160, 2002.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference (DAC)*, pp. 9–14, 2007.

[3] J. Guajardo, B. Škorić, P. Tuyls, S. S. Kumar, T. Bel, A. H. Blom, and G.-J. Schrijen, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information Systems Frontiers*, vol. 11, no. 1, pp. 19–41, 2009.

[4] T. Xu, J. B. Wendt, and M. Potkonjak, "Secure remote sensing and communication using digital PUFs," in *Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 1–11, 2014.

[5] T. Xu, J. B. Wendt, and M. Potkonjak, "Matched digital PUFs for low power security in implantable medical devices," in *International Conference on Healthcare Informatics (ICHI)*, pp. 1–6, 2014.

[6] T. Xu, J. B. Wendt, and M. Potkonjak, "Digital bimodal function: an ultra-low energy security primitive," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 292–296, 2013.

[7] J. Zheng, D. Li, and M. Potkonjak, "A secure and unclonable embedded system using instruction-level PUF authentication," in *Field Programmable Logic and Applications*, pp. 1–4, 2014.

[8] J. Zheng and M. Potkonjak, "DPUF: a reconfigurable IP protection architecture for embedded systems," in *Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 1–2, 2014.

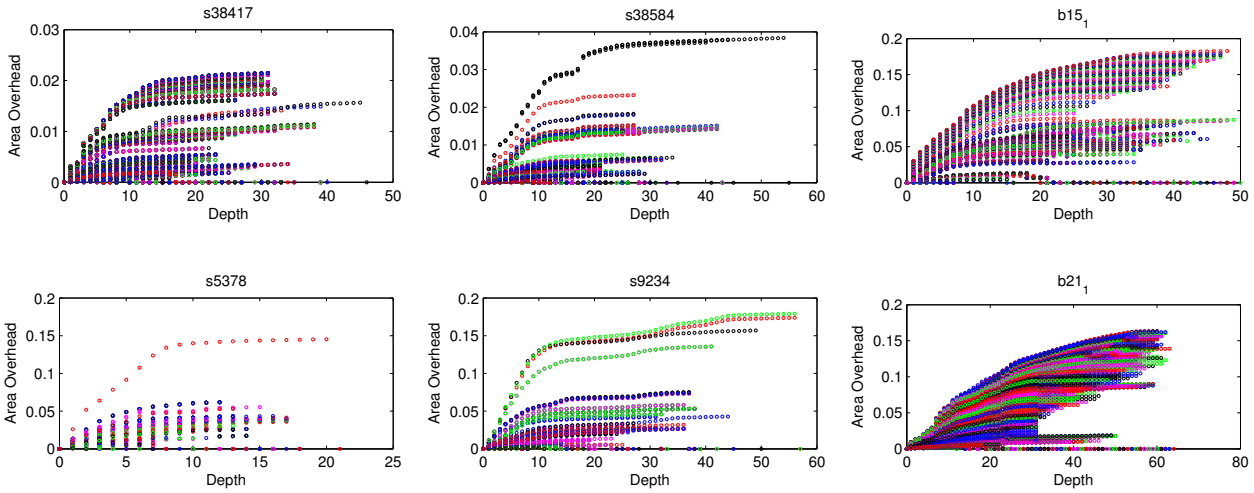


Fig. 7: Area overhead upon replacement of circuitry using PUF-based logic with the labeled circuit depth. The different colors represent the flip-flops whose inputs come from the individual PUF-based logic component.

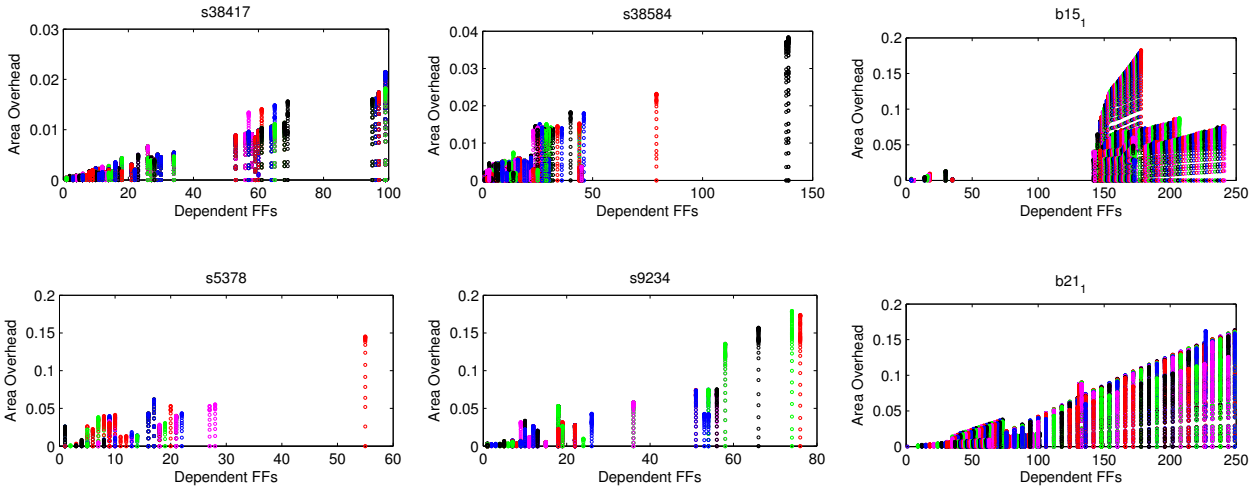


Fig. 8: Area overhead upon replacement of circuitry using PUF-based logic that is affected by the labeled number of flip-flops. The different colors represent the flip-flops whose inputs come from the individual PUF-based logic component.

[9] T. Xu and M. Potkonjak, "Robust and flexible FPGA-based digital PUF," in *Field Programmable Logic and Applications*, pp. 1–6, 2014.

[10] J. B. Wendt and M. Potkonjak, "Nanotechnology-based trusted remote sensing," in *IEEE Sensors*, pp. 1213–1216, 2011.

[11] J. B. Wendt and M. Potkonjak, "The bidirectional polyomino partitioned PPUF as a hardware security primitive," in *Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 257–260, 2013.

[12] U. Ruhrmair, S. Devadas, and F. Koushanfar, *Security based on physical unclonability and disorder*. Springer Verlag: New York NY, 2011.

[13] M. Potkonjak and V. Goudar, "Public physical unclonable functions," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1142–1156, 2014.

[14] F. Luellau, T. Hoepken, and E. Barke, "A technology independent block extraction algorithm," in *Design Automation Conference (DAC)*, pp. 610–615, 1984.

[15] S. Blythe, B. Fraboni, S. Lall, H. Ahmed, and U. de Riu, "Layout reconstruction of complex silicon chips," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 2, pp. 138–145, 1993.

[16] R. Nakagaki, T. Honda, and K. Nakamae, "Automatic recognition of defect areas on a semiconductor wafer using multiple scanning electron microscope images," *Measurement Science and Technology*, vol. 20, no. 7, p. 075503, 2009.

[17] Y. Ren, Y. Shi, and B.-H. Gwee, "A novel gate-level to behavior-level conversion algorithm with high microcell identifiation rate," in *IASTED International Conference*, vol. 712, p. 138, 2010.

[18] R. Torrance and D. James, "The state-of-the-art in IC reverse engineering," in *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 363–381, 2009.

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

[20] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," in *Design, Automation and Test in Europe (DATE)*, pp. 1277–1280, 2013.

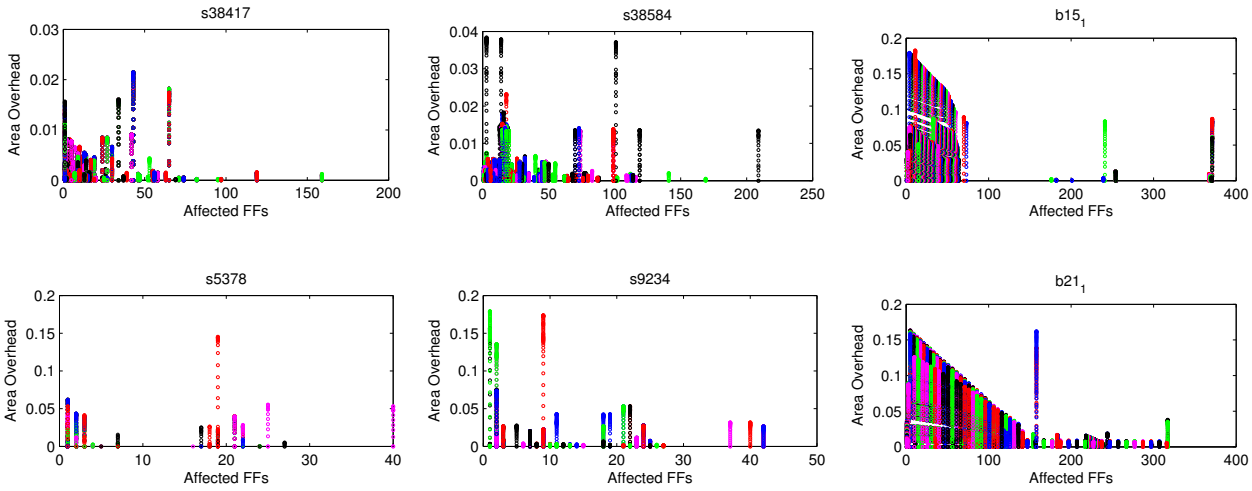


Fig. 9: Area overhead upon replacement of circuitry using PUF-based logic that affects the labeled number of flip-flops. The different colors represent the flip-flops whose inputs come from the individual PUF-based logic component.

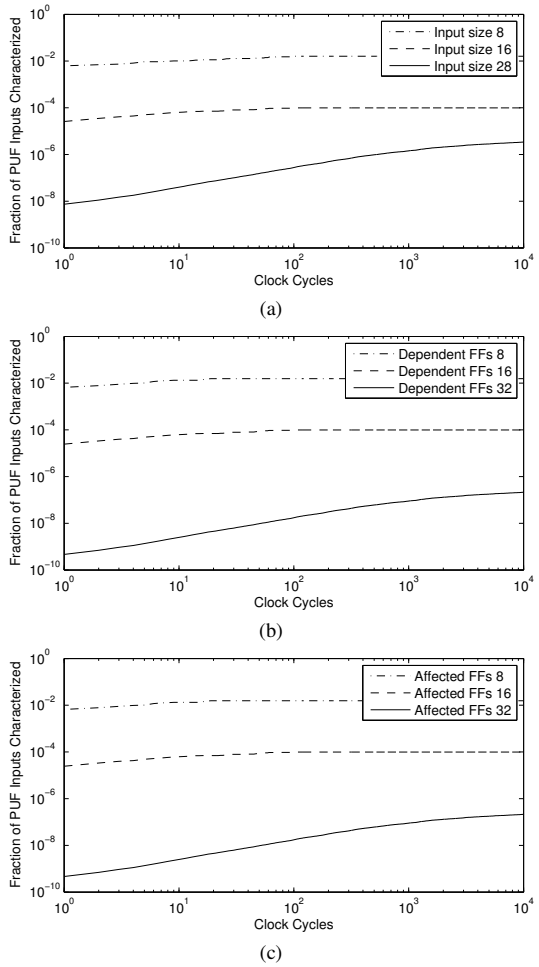


Fig. 10: Fraction of PUF inputs characterized in reverse engineering an obfuscated b12 benchmark circuit.

[21] W. Li, A. Gascon, P. Subramanyan, W. Y. Tan, A. Tiwari, S. Malik, N. Shankar, and S. A. Seshia, "Wordrev: Finding word-level structures in a sea of bit-level gates," in *International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 67–74, 2013.

[22] K. Nohl, D. Evans, S. Starbug, and H. Plötz, "Reverse-engineering a cryptographic RFID tag," in *USENIX Security Symposium*, vol. 28, 2008.

[23] D. Nedospasov, J.-P. Seifert, A. Schlosser, and S. Orlic, "Functional integrated circuit analysis," in *International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 102–107, 2012.

[24] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 674–677, 2007.

[25] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[26] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Design Automation Conference (DAC)*, pp. 83–89, 2012.

[27] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Computer and Communications Security (CCS)*, pp. 709–720, 2013.

[28] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking techniques for intellectual property protection," in *Design Automation Conference (DAC)*, pp. 776–781, 1998.

[29] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *Computer and Communications Security (CCS)*, pp. 733–744, 2013.

[30] "Implementation of security in Actel's ProASIC and ProASICPLUS flash-based FPGAs." http://www.actel.com/documents/Flash_Security_AN.pdf, 2003.

[31] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 663–698, 1985.

[32] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *International Symposium on Circuits and Systems (ISCAS)*, pp. 1929–1934, 1989.

[33] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design and Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.