Hardening Logic Encryption against Key Extraction Attacks with Circuit Camouflage

Bryan J. Wang, Lap Wai Chow, James P. Baukus, and Ronald P. Cocchi
SypherMedia International, Inc.
5455 Garden Grove Blvd, Suite 300, Westminster, CA 92683

Abstract: *Conventional logic encryption has been shown to be weak against key extraction attacks from reverse engineers. However, with the presence of camouflaged logic gates, an adversary is fundamentally unable to use an extracted netlist to back-trace observed outputs and determine the state of key data bits at key-gate inputs. Circuit camouflage hardens logic encryption and provides independent protection as well.*

Keywords: circuit camouflage; hardware obfuscation; logic encryption; camouflage; obfuscation; SAT; key extraction; reverse engineering; security; trusted electronics

Introduction

Integrated Circuit (IC) designs are vulnerable to IP theft from reverse engineering, unauthorized cloning and over-production, and device corruption due to Trojan insertion. The risks to the IC industry have been steadily increasing as reverse engineering capabilities increase, and as worldwide IC production capabilities consolidate into a small number of foreign entities.

Logic encryption, also called logic obfuscation, is a hardware obfuscation technology that modifies a circuit so that it operates correctly only when a set of newly-introduced key-data inputs is correctly applied. The key is known only to the original circuit designers and can be programmed into the device's non-volatile storage such as one-time-programmable OTP memory at a secure facility after manufacture. Without the key data, unauthorized devices manufactured by the IC fabricator or by a third party will not function correctly. [1]

Circuit camouflage is hardware obfuscation technology that prevents reverse engineering of a fabricated device by utilizing a relatively small number of camouflaged gates in the design. A camouflaged cell or gate is a logic gate that appears to have one function based on image analysis of the cell layout, but in fact performs a different function. The camouflaged gates appear identical to the library standard cells used throughout the AISC, so it is not readily apparent to a reverse engineer which cells are regular cells and which are camouflaged cells [4-5].

A method to achieve highly effective protection can be achieved through use of camouflaged gates in conjunction with logic encryption. Use of camouflaged gates in conjunction with logic encryption protects logic encryption key data against known key extraction attacks. Additionally, use of camouflaged gates provides an additional, independent level of security against attackers who are not in possession of the production mask data. If the encryption key is compromised, all camouflaged cells must still be correctly identified and modeled before the circuit can be modeled and duplicated.

Security Threats to Logic Encryption

It has been shown that key data in a conventional logic encryption scheme can be determined from the circuit design in linear time with respect to the key length by applying input vectors to an unlocked fabricated device, observing device outputs, and using satisfiability checking (SAT) software to infer the logic encryption key from the observations and the gate-level netlist [2]. Conventional logic encryption is also vulnerable to other attack models [3]. However, an accurate gate-level netlist of the device is required to perform any attack of this class because the state of a device's internal key-gate nodes must be inferred from its primary outputs. When a number of the device's logic gates are obfuscated with circuit camouflage technology, this type of attack becomes much more difficult because a reverse engineer cannot extract a gate-level netlist whose function matches that of the fabricated device. Therefore, it is highly desirable to utilize camouflaged cells in a logic encryption implementation. Camouflaged gates may be used in the logic encryption network as key-gates, control logic, or glue logic, and they may also be used in the core logic of the fabricated circuit itself.

Use of Appropriate Camouflaged Logic Cells

The term "circuit camouflage" has been used in publications and in the industry to describe a variety of hardware obfuscation technologies, not all of which will have the desired effect of hardening logic encryption implementations against key extraction attacks. An effective camouflage implementation must hide both the locations and the functions of the camouflaged cells. Two effective implementations, Camouflaged Foundry Logic Cells and a Fully Camouflaged Circuit, are discussed below. An attacker faces the computationally infeasible problem of considering multiple possible logic functions for every gate in the design when analyzing either of these camouflage implementations.

Camouflaged Foundry Logic Cells: Camouflaged logic cells can be designed to mimic foundry library cells. An ASIC would be constructed primarily with foundry library cells, interspersed with a small number of camouflaged foundry cells. The layouts of camouflaged foundry cells are nearly identical to layouts of their reference foundry cells, such that an attacker cannot readily differentiate between a "normal" cell and a camouflaged cell. Each camouflaged logic cell performs a different logical function than its reference foundry logic cell.

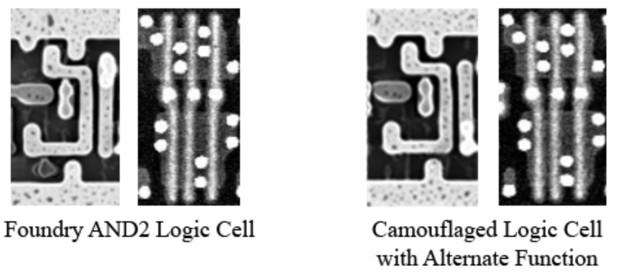

Figure 1. Camouflaged foundry logic cells resemble foundry logic cells but perform an alternate function. For each cell, the left image shows its Metal1 layout and the right image shows Poly, Contact and Active.

Fully Camouflaged Circuit: An ASIC, or a significant portion thereof, can be comprised entirely of camouflaged cells. In such a circuit, the layout appears to be a sea of transistors wherein individual cell functions are not discernable with modern reverse engineering techniques [5]. The camouflaged cells are not designed to mimic the physical designs of a pre-existing foundry library, but instead are designed to appear virtually indistinguishable from one another.

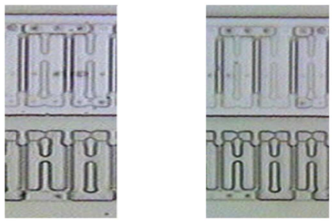

Figure 2. Camouflaged NOR2 gate (left) and NAND3 gate (right) from a fully camouflaged circuit, Active layer

Hardening Logic Encryption against Key Extraction Attacks

Logic encryption is highly resistant to brute force attack because the key length of a logic encryption implementation can be arbitrarily long. With at 2^n possible key combinations, brute force attack quickly becomes impractical. However, as stated previously, conventional logic encryption has been shown to be weak against a class of attacks that are aimed at inferring the logic encryption key data using an unlocked fabricated device.

Threat Model: The capabilities of the attacker can be summarized as follows:

1. The attacker has tools to reverse engineer an IC, such as a SEM, and image processing software.
2. The attacker possesses at least two unlocked devices, one for delayering and imaging and another to be used as a golden reference for application of input vectors.
3. The attacker cannot readily differentiate between a camouflaged cell and a regular cell (if camouflaged foundry cells are used) OR the attacker cannot readily determine the function of any camouflaged gate in the design (if the circuit is fully camouflaged).

Description of Key Extraction Attack: The class of published attacks against logic encryption can be summarized as follows. Note that a gate-level netlist is a necessary component of this class of attack.

1. The attacker obtains two unlocked devices, often purchased on the open market.
2. The attacker extracts the gate-level netlist of the first unlocked device using modern reverse-engineering techniques.
3. The attacker, using analysis software and the extracted gate-level netlist, develops one or more device input vectors with the goal of determining one or more key bits, which are observable at key-gate input nodes.
4. The attacker applies the input vectors from step 3 to the second unlocked device and observes the device outputs.
5. The attacker, using analysis software and the device's gate-level netlist, attempts to infer one or more key bits using the results obtained in step 4.
6. The attacker repeats steps 3-5 until all key bits have been determined.

The use of circuit camouflage technology in the device prevents extraction of an accurate gate-level netlist of the device. This introduces a number of functional discrepancies between the attacker's gate-level netlist and unlocked device, which greatly complicates the attack procedure. The number of functional discrepancies is proportional to the number of camouflaged gates used in the circuit. Since modern reverse-engineering techniques cannot effectively differentiate a camouflaged gate from a normal gate, the attacker is unable to readily determine either the locations or the number of functional discrepancies. Without an accurate gate-level netlist with which to analyze the device's behavior, it is not possible to determine the key-data from inferring the state of key-gate input nodes.

In a conventional circuit with logic encryption shown below, an attacker extracts a gate-level netlist from a fabricated device. When inputs are applied to the fabricated device and outputs are observed, key data can be inferred through application of Boolean logic on an extracted netlist, shown by Rajendran et al.[3]

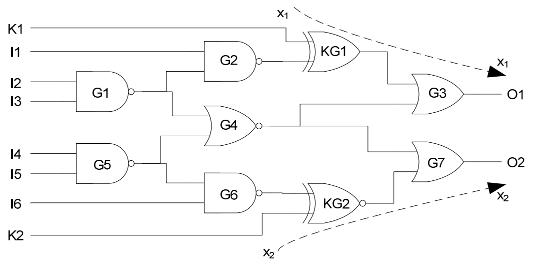

Figure 3. Inferring key bits K1 and K2 through application of Boolean logic.

In a camouflaged circuit with logic encryption shown below (Figure 4), an attacker extracts an erroneous gate-level netlist (Figure 5) from a fabricated device. When inputs are applied to the fabricated device and outputs are observed, functional mismatches between observed and simulated outputs will indicate to the attacker that the extracted netlist is incorrect. Key data cannot be inferred through application of Boolean logic on the erroneous netlist until all netlist errors have been resolved.

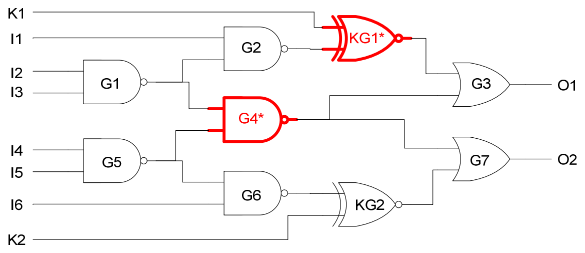

Figure 4. Netlist of a fabricated circuit with camouflaged logic gates KG1 and G4. The layouts of the camouflaged gates suggest different functions than their actual functions.

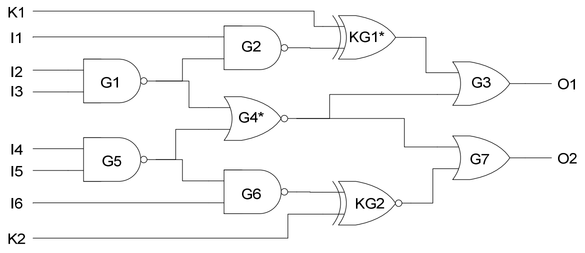

Figure 5. The erroneous netlist extracted from the fabricated device in Figure 4.

Analysis of Camouflaged Logic Encryption

One can quantify the additional protection offered by circuit camouflage to a logic encryption system by utilizing the fact that camouflaged logic cells follow the same design principles as other CMOS cell designs.

Complexity of Camouflaged Logic Functions: The number of inputs and transistors in a camouflaged cell puts an upper boundary on the complexity of the logic function that it can implement. When applied to camouflaged foundry logic cells, this typically means that a Camo cell could perform logic functions that are equal in complexity or less complex than their non-camouflaged cell that they mimic. The physical design of the foundry logic cell may place additional constraints on which logic functions are practical to implement.

When applied to a fully camouflaged circuit, one can potentially determine the number of possible logic functions for a given Camo gate based on its number of inputs and number of transistors, as one could do for any CMOS logic gate. However, due to the difficulty of determining cell boundaries in a circuit with such an extremely regular layout of transistors, it is unlikely that an attacker can do this. Assuming that the attacker finds a method to reliably identify cell boundaries, the problem he faces is of equal or greater complexity than that of analyzing and identifying camouflaged foundry logic cells.

Analysis of an Example Circuit: Analysis of an example camouflaged circuit designed with a foundry logic library is used to quantify the complexity of attacking a camouflaged logic encryption implementation. Table 1 below shows an example of possible camouflaged logic functions for a selection of common one and two-input foundry logic gates. The physical designs of some foundry logic cells may preclude some Camo functions from being realized without significantly altering the layout of the cell.

When the alternate Camo function utilizes fewer pins than the foundry logic gate, the camo logic function may be realized in several ways by utilizing different input pins. For example, an apparent NAND2 gate layout that is camouflaged to perform an inverter function may invert either pin A or pin B, for two possible inversion functions. The number of possible camouflaged functions increases very quickly for logic gates with three or more inputs. For any gate, it is possible to create a camouflaged gate that has an alternate Camo function of static VDD or VSS. For brevity, this analysis has been limited to a selection of one and two-input gates.

Table 1. Possible alternate camouflaged logic functions for selected one and two-input logic gates in an example foundry logic cell library

Physical Appearance	Possible Alternate Camo Functions
XOR2	11 (XNOR2, AND2, NAND2, NOR2, OR2, INV A, INV B, BUF A, BUF B, VDD, VSS)
XNOR2	11 (XOR2, AND2, NAND2, NOR2, OR2, INV A, INV B, BUF A, BUF B, VDD, VSS)
AND2	9 (OR2, NAND2, NOR2, INV A, INV B, BUF A, BUF B, VDD, VSS)
OR2	9 (AND2, NAND2, NOR2, INV A, INV B, BUF A, BUF B, VDD, VSS)
NAND2	7 (NOR2, INV A, INV B, BUF A, BUF B, VDD, VSS)
NOR2	7 (NAND2, INV A, INV B, BUF A, BUF B, VDD, VSS)
BUF	3 (INV, VDD, VSS)
INV	2 (VDD, VSS)

When faced with the possibility that the device under analysis contains camouflaged logic cells, an attacker must consider alternate functions for each gate in the design. Figure 6 below shows the example circuit from Figure 5, with logic gates replaced by boxes representing the number of possible functions for consideration. The number of possible functions for a given logic gate is the number of possible alternate functions from Table 1 plus 1, the apparent function of the gate.

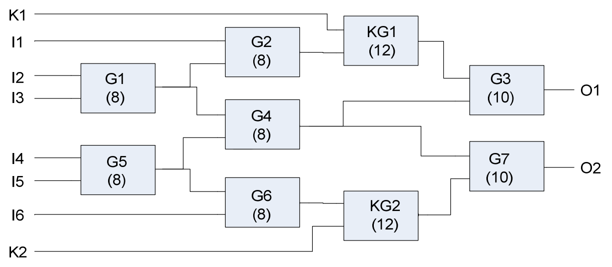

Figure 6. Possible logic functions to consider when resolving functional differences between the extracted netlist (Figure 5) and the fabricated device.

To arrive at the total number of possible configurations for the 9-gate netlist in Figure 6, one would multiply together the number of possible functions for each gate. In this example the number of configuraitons would be $5.9*10^7$ (8*8*8*8*12*12*10*10). This far exceeds the total number of possible input patterns, even considering the key inputs (6 input bits plus 2 key bits allows for 2^8, or 64 possible input combinations). It would be easier for the attacker to analyze the device as a black box with brute force.

Power, Area, and Delay Overheads: Overhead for power and area are highly design dependent. For an implementation using camouflaged foundry logic cells, camouflaged cells have similar power, area, and delay characteristics to the reference foundry library. Non-switching circuitry, such as a gate with a static output, is an area and static power overhead. Partially non-switching circuitry, such as an inverter that is designed to look like a NAND gate, is also an area and static power overhead because one is using a larger footprint than necessary to perform a given logic function. However, camouflaged gates whose actual function is of the same complexity as its apparent function do not incur an area overhead. Highly effective levels of circuit camouflage can be attained with 1-5% extraneous circuitry. Camouflaged cell timing is highly layout-dependent. Some camouflaged cells will have similar timing characteristics to their foundry counterparts, and some may be slower. If critical paths are avoided when placing camouflaged cells, there is effectively no timing penalty.

For a fully camouflaged circuit, one is comparing camouflaged versus non-camouflaged standard cell libraries at a given technology node. Since camouflaged cell design techniques don't inherently impose power, area, or timing penalties, it's not possible to generalize these overheads. However, when comparing a fully camouflaged circuit against an implementation using camouflaged foundry logic cells, the fully camouflaged circuit needs no extraneous circuitry because every logic gate is already camouflaged.

Conclusions

Use of camouflaged gates in a design containing logic encryption is an effective means to harden the circuit against circuit analyses that would lead to extraction of logic encryption key data, as well as providing an independent layer of security against reverse engineering.

References

1. Roy, J.A., Koushanfar, F., and Markov, I.L., "Ending Piracy of Integrated Circuits", *Design, Automation, and Test in Europe*, 2008.
2. Subramanyan, P., Ray, S., and Malik, S., "Evaluating the Security of Logic Encryption Algorithms", *Hardware Oriented Security and Trust,* 2015.
3. Rajendran J., Pino, Y., Sinanoglu, O., Karri, R., "Security Analysis of Logic Obfuscation", *Proceedings of the 49th Annual Design Automation Conference*, 2012.
4. "Circuit Camouflage Technology, SMI IP Protection and Anti-Tamper Technologies", www.smi.tv/SMI_SypherMedia_Library_Intro.pdf.
5. Cocchi, R., Baukus, P., Chow, L.W., Wang, B., "Circuit Camouflage for Hardware IP Protection", *Proceedings of the 51st Annual Design Automation Conference*, 2014.