Camouflaged Programmable Micro Netlists for IP Protection

Bryan J. Wang, Lap Wai Chow, James P. Baukus, and Ronald P. Cocchi
Inside Secure
Westminster, CA

***Abstract*— Programmable camouflaged circuitry hardens ASICs against IP theft from reverse engineers and fabs. Small programmable logic blocks called programmable micro netlists (PMNLs) are scattered throughout the standard cell area. PMNLs contain camouflaged gates that are indistinguishable from surrounding logic. Secret configuration data is required for the ASIC to function, and ASICs protected by this technique resist known attacks to logic locking.**

Keywords—circuit camouflage; hardware obfuscation; logic locking; logic encryption; obfuscation; SAT; reverse engineering; trusted electronics

I. INTRODUCTION

Camouflaged logic cells are logic gates that mimic the physical layouts of foundry library logic gates while performing different logic functions [3]. An attacker who possesses fabricated devices cannot reliably determine the locations or the functions of camouflaged cells using the imaging techniques available to reverse engineering labs. It is inherently difficult to infer the functions of these camouflaged gates using Satisfiability (SAT) attacks because the attacker cannot differentiate between camouflaged and non-camouflaged cells, vastly increasing the attacker's uncertainty [2]. The search space of the attack increases from the simpler case of targeting a relatively small number of known key-gates and obfuscated cells to an open-ended threat model where the function of every logic gate in the ASIC is in question.

To camouflage an IC, camouflaged logic cells and traditional logic cells have previously been organized into small micro-circuits, called Micro Netlists (MNLs), which appear to perform one aggregate function but in fact perform a different function. A new camouflage technique introduces programmed configuration inputs to Micro Netlists, creating Programmable Micro Netlists (PMNLs). PMNLs are a group of camouflaged and non-camouflaged cells that may be configured to perform one of several possible logic functions. They retain all the anti-reverse-engineering properties of non-programmable MNLs, but also allow for secure post-manufacture configuration of their aggregate logic function. The configuration data resides in the IC's NVM block. PMNLs may be used to implement logic locking, requiring a secret key to configure the circuit for correct operation. They may also be used to securely hide cryptographic key data within the logic area of the IC, or to configure regional options to differentiate logic functions between fabricated devices.

The security of PMNLs is based on the difficulty of reverse engineering camouflaged cell designs, and of obtaining the secret contents of the IC's NVM. For a device secured with PMNLs to be successfully cloned, the functions of every camouflaged cell must be successfully identified and the contents of the NVM must be extracted and its contents properly identified. Even if NVM is compromised, extraction of a working model from silicon is still not possible unless every camouflaged cell instance is identified and its function is ascertained, which has proven to be extremely difficult in practice. Because camouflaged cells are physically nearly identical to foundry library cells, they cannot be easily identified in ASIC layouts.

II. PROGRAMMABLE MICRO NETLISTS (PMNLS)

A. Design Components

A PMNL consists of a small number of camouflaged and non-camouflaged cells that perform an aggregate logic function that is different from their apparent function. This set of cells is referred to as the PMNL Core [5]. There may also be local storage, D flip-flops for example, to store configuration data. An ASIC containing PMNLs is shown in Fig. 1.

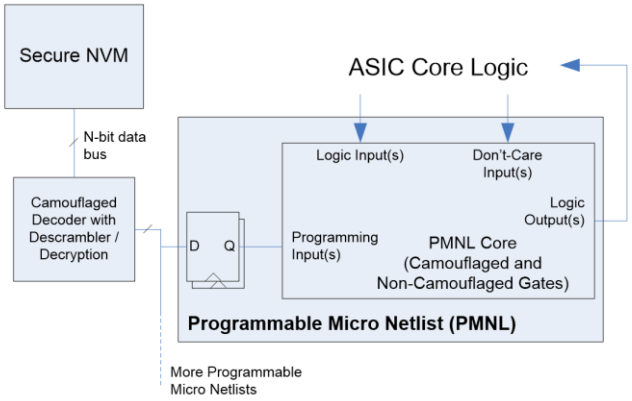

Fig. 1. An ASIC containing PMNLs with local storage of configuration data.

The PMNL's local storage elements are initialized at boot time from a secure NVM. It is likely that a reverse engineer attempting netlist extraction would misidentify the camouflaged logic cells, resulting in an incorrect netlist. The PMNLs are scattered throughout the ASIC core logic area, and are an active part of the ASIC's logic function.

For a reverse engineer to successfully clone an ASIC secured with PMNLs, the correct logical function for each PMNL must be extracted and all secret programming input bits must be applied correctly. A reverse engineer is likely to extract an incorrect logical function for one or more camouflaged cells in each PMNL, and therefore is likely to extract an incorrect function for most if not all the on-chip PMNLs. An example PMNL is shown below. The apparent function (Fig. 2 left) is likely to be extracted by a reverse engineer because the layouts of the camouflaged cells suggest this logical function. The device's actual function (Fig. 2 right) is a NAND2 with one inverted input when P = 0 or an AND2 with one inverted input when P = 1. The secret data to configure a PMNL is stored in the ASIC's secure NVM, and it may be programmed after manufacture.

PMNL designs may contain multiple programming inputs, an example of which is shown in Fig. 3. A reverse engineer is likely to misidentify camouflaged cells and extract the incorrect apparent function of the PMNL core (Fig. 3 left). The PMNL core's actual functional schematic (Fig. 3 right) and reduced schematic symbols for possible combinations of programming inputs {P1, P0} are shown. Utilizing multiple programming bits for a given PMNL increases the search space without offering additional observability to an attacker, making a successful attack more difficult.

B. Design Considerations

One example programming subsystem is described in the previous section, with a camouflaged decoder block connected to a secure NVM block. The NVM should be protected against unauthorized reads and writes. Use of one-time-programmable (OTP) technology also works for this purpose. The contents of the NVM should also be scrambled or encrypted, to be unscrambled or decrypted by the camouflaged decoder block, so the raw NVM data does not expose the actual programming bitstream. Other programming subsystems might incorporate a PUF so that the NVM image for each chip is unique, and one chip's bitstream data may not be used to compromise another.

Another option for programming the PMNLs is to organize the local storage FFs into one or more shift registers, requiring fewer connections from the NVM to the PMNLs. This may help reduce routing congestion. It is not recommended to allow the unscrambled secret data that resides in PMNL local storage of an unlocked device to ever be shifted out of the device, or otherwise observed by a user of the device.

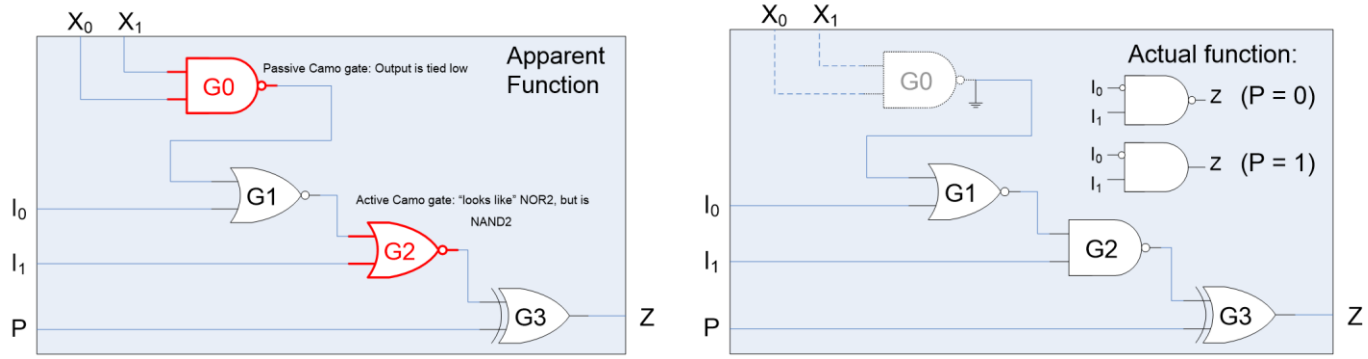

Fig 2. The apparent function (left) and actual function (right) of a PMNL Core with a single programming bit.

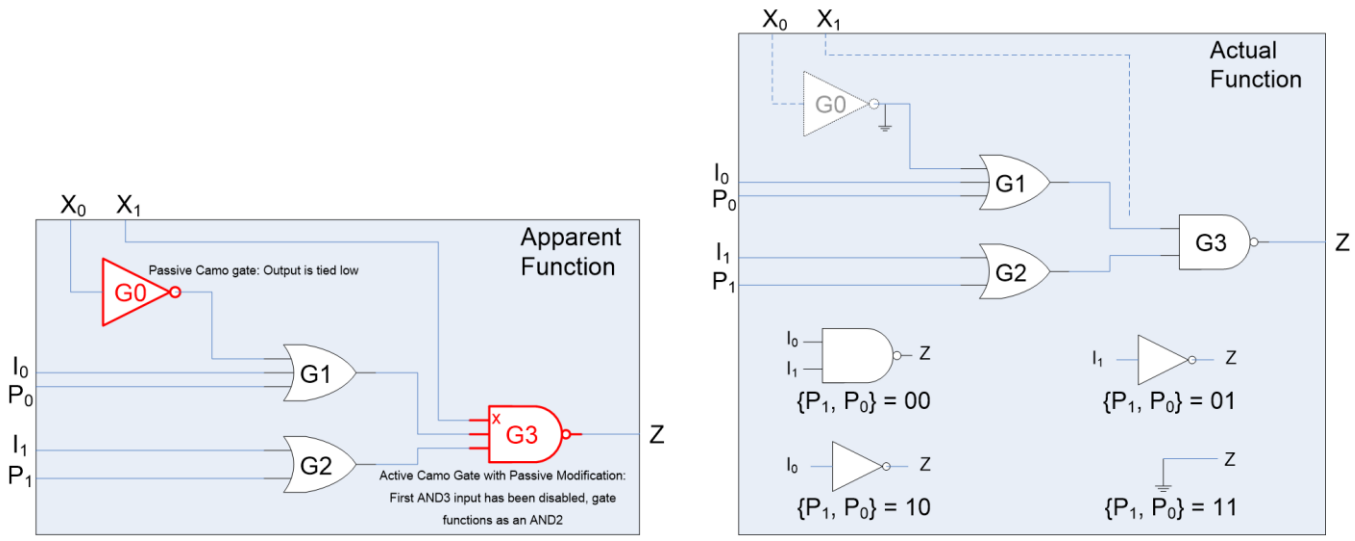

Fig 3. The apparent function (left) and actual function (right) of a PMNL Core with multiple programming bits.

To complicate SAT-based analysis of the circuit, a relatively small number of camouflaged logic gates should be placed throughout the ASIC regions to be protected, as opposed to using them only within PMNLs. With proper application of camouflaged gates, a reverse engineer attacking the circuit faces the daunting possibility that any gate in the design could be a camouflaged gate with an unknown function.

C. *Resource Overhead*

PMNLs should be used sparingly, as each PMNL requires significant hardware overhead. A single-bit-programmable PMNL would require one NVM bit, several gates in the PMNL core, and one flip-flop for local storage. These items scale linearly with the number of PMNL programming bits used in the device. For a secure implementation, the programming subsystem would also require camouflaged decoding and descrambling logic, which scales roughly logarithmically with the number of programming bits. If the bitstream is to be encrypted, then a decryption engine is also required, which consumes a number of gates independent of the number of programming bits. Enough PMNLs and programming bits must be used such that brute force attacks are not practical.

III. Applications

Two applications of PMNLs are outlined in this section.

A. *Logic Locking*

Logic locking, sometimes referred to as logic encryption, is a hardware obfuscation technique that modifies a digital circuit, inserting key-gates with additional key inputs into a circuit, to lock the circuit so that it will not behave correctly unless the correct key inputs are applied [1]. A PMNL may be used as a key-gate for logic locking. The designer then has flexibility to use the PMNL to realize complex logic functions in the ASIC. PMNLs may be arbitrarily complex, and can be designed to replace groupings of combinational logic gates in an ASIC.

Utilizing multiple programming bits for a given PMNL increases the search space of an attack without offering additional observability to an attacker, making a successful attack more difficult. Logic cones that propagate and reduce down to a single signal or a small number of signals are good candidates for replacement because of their limited observability. A PMNL may be constructed to replace an arbitrary grouping of logic gates with a camouflaged programmable circuit that can be configured to perform many functions, of which only one function is valid.

B. *Limited Configurability*

With the presence of NVM, designers may allocate some persistent storage to serve as configuration data. This might be used to enable or disable hardware features, or to alter the behavior of a hardware circuit. For example, one device may contain a customized cryptographic engine that can be configured to behave differently for different customers. With the presence of camouflaged PMNLs, supporting this type of limited configurability can be done in a more secure manner than with NVM alone, since the camouflaged logic protects the underlying hardware IP from reverse engineering from either a customer or from an outside attacker.

IV. Analysis

Logic locking with PMNLs is highly resistant to brute force attack, with 2^n possible key combinations. The key length can be arbitrarily long, with the primary restriction on length being the overhead in circuit area and routing resources that is required to implement the logic locking. The goal of logic locking attacks is to obtain the correct secret key in a much shorter amount of time.

A published analysis of a camouflaged circuit protected with logic locking is applicable to the camouflaged PMNL circuits described in this paper. As stated therein, the capabilities of the attacker can be summarized as follows [2]:

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 reliably differentiate between a camouflaged cell and a regular cell.

The fact that the attacker cannot differentiate between camouflaged and non-camouflaged cells complicates an attack tremendously by creating uncertainty as to the logic function of every logic gate in the ASIC, whether or not that gate is camouflaged. The analyst is unable to isolate and analyze logic locking circuitry within a network of known logic gates. The functions of all logic gates in the design must be treated as unknown until they are determined individually through analysis or physical probing.

To apply the published analysis from [2] to an ASIC of any practical size, one would see a numerical explosion of circuit function possibilities far exceeding the number of possible logic locking key combinations. An attacker can reduce this number considerably by applying additional constraints to the analysis (for example, to assume that no more than 10% of all gates are camouflaged). Although such constraints may appear reasonable, it's not possible to create a foolproof framework of constraints to analyze any camouflaged circuit. Ultimately, the application of such constraints will rely on the attacker's judgment. When a constraint is incorrect, the attacker's analysis may be invalidated.

Currently there is no known method of logical analysis to reliably determine the logic functions of the logic gates of a camouflaged circuit. Physical probing of cells on the die is a method that can determine the logic functions of camouflaged cells, but because every logic gate is potentially camouflaged, physical probing is not economical or reliable enough for use on every gate comprising an ASIC.

Note that a successful attack requires both the programming bitstream data and the circuit function. Neither the secret programming data nor the circuit function alone will enable an attacker to clone the device.

V. Conclusion

The use of camouflaged programmable micro netlists (PMNLs) offers the benefits of logic locking while resisting known vulnerabilities to attack, especially when used in conjunction with other circuit camouflage techniques throughout the ASIC region to be protected.

References

[1] J. Roy, F. Koushanfar, and I. Markov, "Ending Piracy of Integrated Circuits", Design, Automation, and Test in Europe, 2008.

[2] B. Wang, L. Chow, J. Baukus, and R. Cocchi, "Hardening Logic Encryption against Key Extraction Attacks with Circuit Camouflage", GOMACTech Conference, 2017.

[3] "Circuit Camouflage Technology, SMI IP Protection and Anti-Tamper Technologies", www.smi.tv/ SMI_SypherMedia_Library_Intro.pdf.

[4] P. Subramanyan, S. Ray, S. Malik, "Evaluating the Security of Logic Encryption Algorithms", Hardware Oriented Security and Trust, 2015.

[5] L. Chow, B. Wang, J. Baukus, and R. Cocchi, "Secure Logic Locking and Configuration with Camouflaged Programmable Micro Netlists", US Patent Application No. 65/542,049, 2017.