Opening the Doors to *Dynamic* Camouflaging: Harnessing the Power of Polymorphic Devices

Nikhil Rangarajan*, *Member, IEEE,* Satwik Patnaik*, *Graduate Student Member, IEEE,* Johann Knechtel, *Member, IEEE,* Ramesh Karri, *Fellow, IEEE,* Ozgur Sinanoglu, *Senior Member, IEEE,* and Shaloo Rakheja, *Member, IEEE*

Abstract—The era of widespread globalization has led to the emergence of hardware-centric security threats throughout the IC supply chain. Prior defenses like logic locking, layout camouflaging, and split manufacturing have been researched extensively to protect against intellectual property (IP) piracy at different stages. In this work, we present *dynamic camouflaging* as a new technique to thwart IP reverse engineering at all stages in the supply chain, viz., the foundry, the test facility, and the end-user. Toward this end, we exploit the multi-functionality, post-fabrication reconfigurability, and run-time polymorphism of spin-based devices, specifically the magneto-electric spin-orbit (MESO) device. Leveraging these unique properties, *dynamic camouflaging* is shown to be resilient against state-of-the-art analytical SAT-based attacks and test-data mining attacks. Such dynamic reconfigurability is not afforded in CMOS owing to fundamental differences in operation. For such MESO-based camouflaging, we also anticipate massive savings in power, performance, and area over other spin-based camouflaging schemes, due to the energy-efficient electric-field driven reversal of the MESO device. Based on thorough experimentation, we outline the promises of *dynamic camouflaging* in securing the supply chain end-to-end along with a case study, demonstrating the efficacy of *dynamic camouflaging* in securing error-tolerant image processing IP.

Index Terms—Hardware security, IP protection, Layout camouflaging, Dynamic camouflaging, Post-fabrication reconfigurability, Dynamic morphing, Functional polymorphism, Spin devices.

1 Introduction

As the aggressive scaling of complementary metal-oxide-semiconductor (CMOS) technology nodes reaches physical device limits, traditional CMOS-based architectures face significant challenges ranging from saturated performance gains to increased power density and variability, coupled with concerns for reliability. The continual miniaturization of CMOS technology has not only complicated chip design but also necessitated advanced and expensive fabrication facilities. Alternative technologies are being pursued extensively, which can augment CMOS in enabling higher memory and logic efficiency. These include several emerging devices such as silicon nanowire field-effect transistors (SiNW-FETs) [1], memristors [2], negative capacitance FETs (NCFETs) [3], spin devices [4], etc. Spintronic devices, in particular, have emerged as one of the top contenders for the post-CMOS era [4].

The expeditious globalization of the electronics industry has resulted in the outsourcing of the integrated circuit (IC) supply chain. Such a distributed supply chain, which is often spread across geographically different locations, enables various attacks, ranging from piracy of intellectual property (IP) to illegal and unauthorized overproduction of ICs, and targeted insertion of malicious circuits known as hardware Trojans. IP piracy, in particular, is quite multi-faceted and an attacker has different avenues to mount such an attack, ranging from an *untrustworthy foundry*, an *untrustworthy test facility*, to *malicious end-users* (Fig. 1). Estimates suggest loss to the tune of billions of dollars annually due to infringement of IP cores. While malicious employees residing in an untrusted foundry or an end-user could pirate the design by reverse engineering (RE) and/or mounting Boolean satisfiability (SAT)-based attacks [9], [10] to decipher the chip IP, an adversary in the test facility can misuse test patterns to compromise the security of a chip [5], [11]. Various design-for-trust schemes have been proposed in the literature (including few which have been demonstrated on silicon) over the past decade to counter IP

TABLE 1
IP Protection Techniques Versus Untrusted Entities in IC Supply Chain
(✓: Protection Offered, ✗: No Protection Offered)

Technique	FEOL/BEOL	Test Facility	End-User
Logic Locking	✓/✓	✓ ([5])	✓
Layout Camouflaging	✗/✗ (✓/✗ [6])	✗	✓
Split Manufacturing	✓/✗	✗	✗ (✓ [7], [8])
Dynamic Camouflaging	✓	✓	✓

*Nikhil Rangarajan and Satwik Patnaik contributed equally.

- Satwik Patnaik and Ramesh Karri are with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University (NYU), Brooklyn, NY, 11201, USA.
- Nikhil Rangarajan, Johann Knechtel and Ozgur Sinanoglu are with the Division of Engineering, New York University Abu Dhabi (NYU AD), Abu Dhabi, 129188, United Arab Emirates.
- Shaloo Rakheja is with the Holonyak Micro and Nanotechnology Laboratory, University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, 61801.
- Corresponding authors: Nikhil Rangarajan (nikhil.rangarajan@nyu.edu) and Satwik Patnaik (sp4012@nyu.edu).

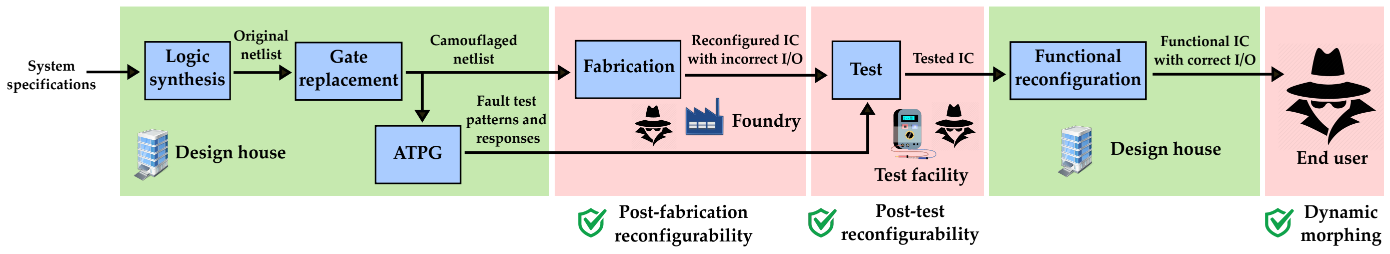

System specifications
Logic synthesis
Original netlist
Gate replacement
Camouflaged netlist
Fault test patterns and responses
ATPG
Design house
Fabrication
Reconfigured IC with incorrect I/O
Foundry
Test
Tested IC
Test facility
Functional reconfiguration
Functional IC with correct I/O
Design house
End user
Post-fabrication reconfigurability
Post-test reconfigurability
Dynamic morphing

Fig. 1. Threat model for *dynamic camouflaging*-based IP protection. Green and red blocks represent the *trusted* and *untrusted* entities, respectively. The protection schemes—which are all flavors of dynamic camouflaging—employed for each of the untrusted entities are mentioned below the respective red blocks, and indicated by green shields. Gate replacement, which can either be random or through some designer's chosen heuristic, involves the selective replacement of gates in the original netlist with polymorphic gates (magneto-electric spin-orbit (MESO) gates in this work). After fabrication and testing, the design is sent back to the design house (or some trusted facility) for functional reconfiguration before being deployed in the open market. ATPG stands for automatic test pattern generation.

piracy. Table 1 summarizes the protection offered by some of these techniques in the face of untrusted entities; they are discussed briefly next.

1.1 An Overview of IP Protection Schemes

Logic locking (LL) protects the underlying design IP by inserting dedicated locks, which are controlled by a secret key. A locked circuit contains additional inputs, which are referred to as *key inputs*, and are driven by an on-chip *tamper-proof memory* (TPM). Most common locking mechanisms are realized by inserting additional logic (e.g., XOR/XNOR gates, AND/OR gates or look-up tables (LUTs)). The locked IC is activated by a trusted facility or the design house after fabrication and testing (but before deploying in the open market), namely by loading the secret key onto the chip's dedicated TPM. Examples include random logic locking (RLL), fault analysis-based locking (FLL) [12], Anti-SAT [13], and stripped functionality logic locking (SFLL) [14]. Note that the overall security of LL hinges on the *secure realization of TPMs*, which remain under active research and development [15].

Layout camouflaging (LC) obfuscates the layout implementation—and thereby attempts to obfuscate the functionality—by using specialized camouflaged cells which aim to be indistinguishable across several functions. This can be achieved by (i) using dummy contacts [16], (ii) leveraging threshold voltage-dependent cells [17], (iii) incorporating AND-tree camouflaging [18], and (iv) obfuscating the interconnects [6]. An important consideration for LC is that almost all prior works need to *trust the foundry* for implementing their obfuscation mechanisms.

Split manufacturing (SM) entails the physical separation of the entire chip stack into front-end-of-line (FEOL) and back-end-of-line (BEOL) layers, across geographically distinct foundries. Typically, the FEOL consists of transistors (device layer) and lower metal layers (M1–M3) which are fabricated by an advanced, off-shore *untrustworthy* foundry, while the remaining metal layers are manufactured on top of the incomplete chip at a *trustworthy*, in-house, low-end facility [19]. This physical separation of the design IP avoids dissemination of the complete layout information to one untrustworthy foundry. A multitude of techniques has been proposed in the recent literature to safeguard FEOL layouts for SM, e.g., [19], [20], [21]. However, it is essential to note that, SM can safeguard the design IP from untrusted foundries only, *but not against untrusted end-users*.

To summarize, although IP protection techniques have been proposed to safeguard the supply chain against malicious entities, each of these solutions have some caveats. Logic locking has the potential to protect the IC supply chain end-to-end but, in its current state, the resilience depends on a TPM to store the secret key.

1.2 Role of Emerging Devices in Securing Hardware

Emerging devices are prime candidates for augmenting hardware security [22], [23], [24], [25]. The controllable ambipolarity in SiNW-FETs has been exploited to implement camouflaged layouts in [22]. Recent research in the field of emerging device-based security has explored the domain of spintronics [26]. Spin devices like the charge-spin logic and magneto-electric spin-orbit logic (MESO) [4] possess innate *run-time polymorphism* and *post-fabrication reconfigurability* capabilities, which are typically not afforded by CMOS and other emerging devices. The additive nature of the input spin currents coupled with a magnetic tunnel junction (MTJ)-based differential voltage readout enables these spin devices to implement majority logic directly and exhibit polymorphic characteristics. Recent works [24], [25] on using emerging devices for LC have leveraged polymorphic logic for static camouflaging. However, the *true potential* of polymorphic devices lies in *dynamic camouflaging*, which is unexplored yet—therefore, exploring dynamic camouflaging is the focus of this paper.

1.3 Dynamic Camouflaging

Dynamic camouflaging involves obfuscating and switching the device-level functionality *post-fabrication*, as well as *during run-time*, thereby hindering various attacks throughout the IC supply chain. We study *dynamic camouflaging* using polymorphic spin devices and establish *security* and *computational accuracy* as two entangled design variables, especially for error-tolerant applications such as image processing and machine learning. We focus on such scenarios as we believe they are meaningful, but we note that polymorphic gates can in principle result in any arbitrary dynamic behavior. However, such behavior can be impractical, as it would come along with an excessive loss of computational accuracy. In other words, we study dynamic camouflaging based on run-time reconfiguration among functionally equivalent or approximately equivalent circuit structures with the help of polymorphic gates, while maintaining the practicality of such circuits. For such applications, dynamic camouflaging can thwart both exact [9], [10] and approximate SAT (*AppSAT*) attacks [27], as we show in this work.

In general, we discuss extensively about securing the supply chain end-to-end using spin-based devices, and circumventing the risks associated with untrusted foundries, test facilities, and end-users (Fig. 1 and Table 1).

1.4 Contributions

The contributions of this work are summarized as follows:

1) We introduce the concept of *dynamic camouflaging* leveraging the inherent functional polymorphism of spin devices. Toward this end, we demonstrate the promising security properties pertaining to the MESO device as a representative spin device. We choose the MESO device owing to its superior performance metrics and CMOS compatibility.
2) We propose a secure end-to-end solution to counter IP piracy across the distributed IC supply chain, encompassing an untrusted foundry, untrusted test facility, and an untrusted end-user. This is the *first* work in the context of LC to safeguard the supply chain end-to-end. Extensive simulations demonstrate the superior resilience of our proposed scheme against state-of-the-art attacks.
3) From the purview of an untrusted foundry, we show that advanced "inside foundry" attacks do not compromise our security claims, as we rely on the concept of *post-fabrication reconfigurability*.
4) The idea of post-fabrication reconfigurability is also leveraged to demonstrate resilience against attackers in an untrusted testing facility. By employing *post-test configuration*, we protect the design IP against test-data mining attacks like *HackTest* [11]. We carry out detailed simulations on various test cases for static and dynamic camouflaging.
5) We extend the benefits of dynamic camouflaging, through *dynamic morphing*, to protect also against untrusted end-users, especially for error-tolerant applications such as image processing. We show the implications of using approximate SAT-based attacks (*AppSAT*) [27] for the same.
6) Finally, we project the superior cost in terms of synthesis-level power, performance, and area (PPA) for full-chip camouflaging in contrast with other selected, spin-based camouflaging schemes.

2 Background and Motivation

Here, we discuss the recent advancements in LC along with demonstrated attacks, which have been tailored toward static camouflaging. Further, we report on some early studies directed toward the notion of dynamic camouflaging.

2.1 Static Layout Camouflaging & SAT-Based attacks

Early research in the field of LC was aimed primarily toward (i) selection of gates to be camouflaged, and (ii) the design of camouflaged cells. Most of the existing LC schemes have a high layout cost (in terms of PPA) and are therefore limited for practical implementation. The ambiguous XOR-NAND-NOR camouflaged cell proposed in the seminal work of [16] has a power overhead of 5.5×, timing overhead of 1.6×, and area overhead of 4×, respectively, when compared to a conventional 2-input NAND gate. Promising works such as the threshold-dependent, full-chip LC proposed in [17] induces overheads of 14%, 82%, and 150% on PPA, respectively. Therefore, existing LC schemes can be applied only selectively due to their significant impact on PPA budgets. Such a constrained application of these techniques (e.g., camouflaging fixed set of gates) leads to a compromise in security, which is discussed next. It should also be noted that most existing camouflaging schemes necessitate the use of a *trusted* foundry and the camouflaging effected by them is *static*.

In 2015, Subramanyan *et al.* [9] and Massad *et al.* [10] independently challenged the security guarantees offered by LL and LC, respectively. The attack—commonly referred to as SAT-based attack in the literature—leverages Boolean satisfiability to compute so-called *discriminating input patterns* (DIPs). By definition, a DIP generates different outputs for the same input pattern across two (or more) different keys, which indicates that at least one of the keys is incorrect. The attack then proceeds in a step-wise fashion where different DIPs are evaluated until all wrong keys have been eliminated. Inspired by the promise raised by the SAT-based attack, research groups focused on SAT-resilient camouflaging schemes [18] which force the attack to explore exponential numbers of DIPs. Such SAT-resiliency is achieved by inserting so-called point functions for, e.g., AND-trees, OR-trees, which ultimately leads to very low output corruptibility. High-corruptibility schemes like FLL [12] are integrated for such SAT-resilient schemes to improve output corruptibility, thereby providing a two-layer defense. Shamsi *et al.* [27] formulated *AppSAT*, which reduces such compound schemes to their low-corruptibility constituent by "peeling off" the high-corruptibility portion.

2.2 Toward Dynamic Camouflaging

Dynamic camouflaging builds on the foundations of polymorphic computing which is a subset of reconfigurable computing. Reconfigurable computing using programmable devices (such as field-programmable gate arrays, FPGAs) typically fix the logic functionality of the chip *before* run-time. In polymorphic computing, however, the devices are reconfigured in time and space *during* run-time. Therefore, dynamic camouflaging involves dynamically obfuscating the circuit at the device/circuit level. Individual gates are configured correctly *only after fabrication and testing*, and these gates can further switch between different functionalities at run-time by application of certain control inputs—we refer to this approach as *dynamic morphing*. Contrary to static camouflaging schemes like [6], [16], [17], [18], dynamic camouflaging requires polymorphic logic gates.

Prior work using programmable CMOS for IP protection leverage reconfigurable logic barriers [28] and reconfigurable key gates [29]. These techniques *do not* use functional polymorphism, but rather fix the logic functionality

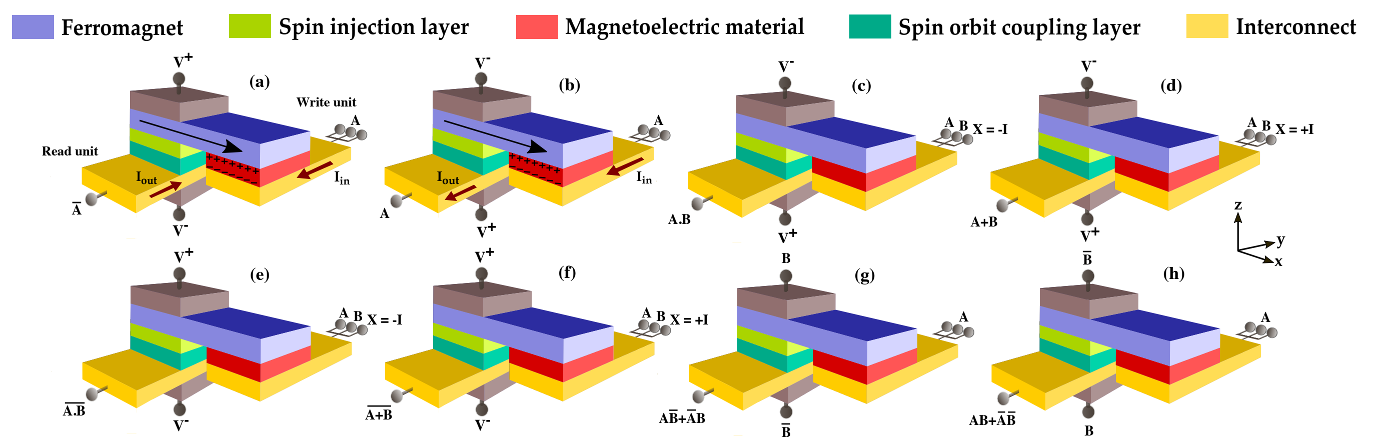


Fig. 2. (a-h) Implementation of INV, BUF, AND, OR, NAND, NOR, XOR, XNOR with a single MESO device, using different input configurations. Signals A and B are logic inputs, and X is a control input required for some functionalities. Note that INV, BUF, XOR, and XNOR gates have dummy wires/contacts at their input terminals, to make them optically indistinguishable from other implementations.

using select lines and/or key-bits. Although it is possible to implement functional polymorphism using CMOS-based reconfigurable units, such as LUTs within FPGAs, the overheads incurred by such schemes can be high, as discussed further in Section 8.

The notion of dynamic functional obfuscation was put forward by Koteshwara *et al.* [30], where sequentially triggered counters are leveraged to provide security guarantees. This scheme requires additional circuitry to alter the key, which is potentially prone to removal attacks. Another study leverages *hot-carrier injection* to program threshold voltage-based CMOS gates post-fabrication [31]. The authors also showed a proof-of-concept implementation by fabricating obfuscated adders in 65-nm bulk CMOS process. However, they *do not support run-time reconfiguration* and suffer from large PPA overheads. For example, a camouflaged NAND gate incurs power overhead of 9.2×, delay overhead of 6.6×, and area overhead of 7.3×, all with respect to a regular 2-input NAND gate.

Run-time polymorphism and, hence, dynamic camouflaging is challenging to implement for CMOS at the device level, owing to fundamental limits. Our scheme enables a radically different solution, wherein we use the unique properties of spin devices to achieve truly polymorphic chips.[1] This is especially useful for error-tolerant applications such as image processing. We argue that dynamic camouflaging is also particularly promising for approximate computing applications, which trade-off computational accuracy for better energy-efficiency (Sec. 7).

3 Dynamic Camouflaging: Working Principle

3.1 The Magneto-Electric Spin-Orbit (MESO) Device: Construction and Operation

The spin device considered in this study is the MESO device, whose operation is based on the phenomena of magneto-electric (ME) switching [32] and inverse spin-orbit effects [33]. The schematic of the MESO device implementing different Boolean functions is shown in Fig. 2. The inputs/outputs are electric currents, and the logical information is encoded in the direction of the current flow. A detailed description can be found in [4].

During the writing phase, an input electric current flowing in the $\pm\hat{y}$ direction through the non-magnetic interconnect sets up an electric field in the $\pm\hat{z}$ direction within the ME capacitor (red in Fig. 2). The resulting ME field switches the magnetization state of the ferromagnet (purple) along the $\pm\hat{x}$ direction. Information is written into the MESO device by transducing the input electric current into the magnetization state of the device. Typical room-temperature multiferroics used for the ME capacitor include $BiFeO_3$ and $LuFeO_3$. The charge accumulation across an ME capacitor in response to an applied electric field is given as $Q_{\mathrm{ME}} = A_{\mathrm{ME}}(\epsilon_0\epsilon_{\mathrm{mf}}E + P_{\mathrm{mf}})$, where A_{ME} is the cross-sectional area of the capacitor, $\epsilon_0 = 8.85 \times 10^{-12}$ F/m is the permittivity of free space, ϵ_{mf} is the relative dielectric permittivity of the ME, and P_{mf} is the saturated ferroelectric polarization. For the $BiFeO_3$ capacitor considered in [4], $A_{\mathrm{ME}} = 10^{-16}$ m^2, while $\epsilon_{\mathrm{mf}} = 54$. The electric field to be applied to the ME capacitor to switch it all-electrically is $E = E_{\mathrm{mf}}B_c/B_{\mathrm{mf}}$, where $E_{\mathrm{mf}} = 1.8 \times 10^6$ V/m refers to the electric switching field, $B_{\mathrm{mf}} = 0.03$ T is the exchange bias at switching field, and $B_c = 0.1$ T is the ME switching field.

After the writing process is complete, which takes ~ 200 ps [4], the supply voltages V^+ and V^- are turned on to initiate the reading phase. In the reading phase, a spin-polarized current is injected into the spin-orbit coupling (SOC) layer, which converts the spin current into electric current at the output node (I_{out}), due to the inverse spin-Hall and inverse Rashba-Edelstein effects [34]. These topological effects result in the shifting of the Fermi surface of the high-SOC material in k-space. This shift causes a charge imbalance and hence a charge current in the Fermi surface, in a direction orthogonal to the injected spin density. The magnitude of the charge current transduced by the SOC layer as a result of the applied spin density is given by

$$j_c = \frac{\alpha_R \tau_s}{\hbar} j_s = \lambda_{\mathrm{IREE}}\, j_s, \tag{1}$$

1. While we choose the MESO device as a representative example for our work, the concepts presented in this work can be readily extended to any emerging device which exhibits qualities like functional polymorphism and post-fabrication functionality reconfiguration.

where α_R is the Rashba coefficient, τ_s is the spin relaxation time and λ_{IREE} ($\sim 1.4 \times 10^{-8}$m) is the inverse Rashba-Edelstein length of the SOC material [4].

The direction of the output current is determined by the polarity of the supply voltages V^+/V^- (+/- 100 mV) applied on the nanomagnet, and the final magnetization state of the ferromagnet. For instance, when the ferromagnetic moment is along $+\hat{x}$ and the flow of the injected spin current is along $-\hat{z}$, with spin polarization along $+\hat{x}$, the direction of the charge current generated is along $+\hat{y}$ (Fig. 2a). However, when the ferromagnet is reversed to the $-\hat{x}$ direction, with the injected spin current direction unchanged but the spin polarization now along $-\hat{x}$, the output charge current reverses to $-\hat{y}$. The same reversal in the direction of output current can also be achieved by keeping the ferromagnetic moment constant and flipping the voltage polarities V^+/V^-.

The total intrinsic switching time of the MESO device is a combination of the time taken to charge the multiferroic capacitor, τ_{ME}, and the ferroelectric polarization/magnetization reversal time, τ_{mag}. These are given as $\tau_{ME} = 2Q_{ME}/I_{ISOC}$ and $\tau_{mag} = \pi/\gamma B_c$, where I_{ISOC} is the current produced by the spin-orbit effect and γ is the gyromagnetic ratio of the electron. Evaluating these switching times according to the parameters provided in the supplementary material of [4] yields an intrinsic switching time of ~230 ps. The total switching time of the MESO device is then obtained as ~258 ps, by adding the interconnect delay of 2.9 ps (quoted from the supplementary material of [4]) and the extrinsic peripheral delay of ~25 ps which corresponds to multiplexers (MUXes) simulated using *Cadence Virtuoso* for the 15-nm CMOS node, considering the NCSU FreePDK15 FinFET library, for a supply voltage of 0.8V.

For a further, in-depth analysis about the switching and transduction processes in the MESO device, interested readers are kindly referred to [4]. Finally, we note that the MESO device has sufficient gain, namely ~ 10 [4], to drive multiple fan-out stages.

3.2 Polymorphic Gates

By switching the polarity of the supply voltages, we can implement a buffer (BUF) or an inverter (INV) using the same device (Fig. 2(a,b)). Further, we can implement complex gates such as majority logic, by leveraging the additive nature of the input signals. As shown in Fig. 2 (c,d), A and B are the signal inputs and X is the tie-breaking control input. The polarity of X decides the functionality of the MESO gate. Here, for $X = -I$, it realizes an AND gate and for $X = +I$, it realizes an OR gate. To implement NAND and NOR gates, the polarities of the supply voltages are flipped (Fig. 2 (e,f)). For XOR and XNOR gates, the tie-breaking input X is eliminated, and one signal is provided at the input terminal. The other input signal is encoded in the voltage domain and applied directly at the V^+/V^- terminals (Fig. 2 (g,h)).

Illustrative waveforms showing the device operation and functional reconfiguration between AND/OR and NAND/NOR, on flipping the control signal X, are shown in Fig. 4. The MESO device with additional peripheral circuitry is shown in Fig. 3. The *control bits* deciding the input and control signals can either be derived from a control block (Fig. 5), or even from a true random number generator (TRNG), if random reconfiguration is applicable, e.g., for error-tolerant applications such as image (video) processing, machine learning, etc. Configuring the MESO device via different supply voltages and electric currents allow us to dynamically implement all basic Boolean gates within a single structure. This essential feature is used for dynamic camouflaging in this work.

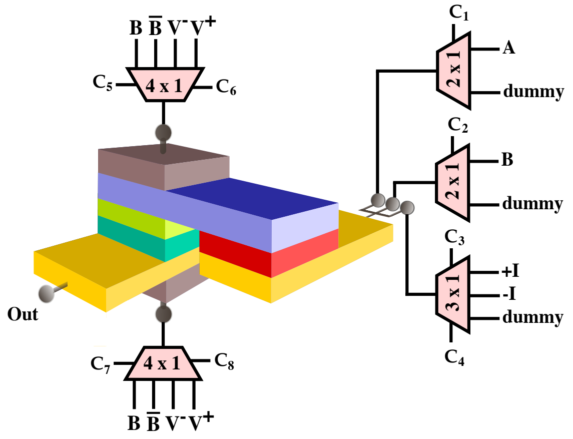


Fig. 3. A generic MESO gate with peripheral MUXes, which dictate the input and control signals through control bits C_1–C_8. This generic structure implements any of the Boolean functionalities in Fig. 2 (a-h) once the appropriate control bits are provided.

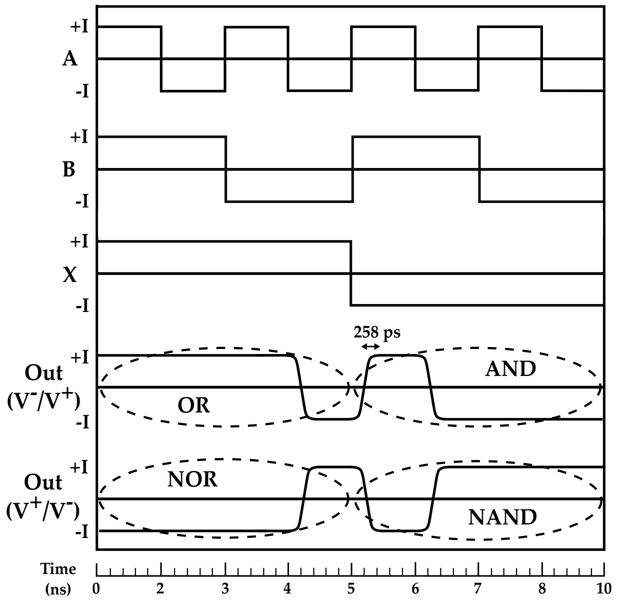


Fig. 4. Timing waveforms for MESO-based AND / OR / NAND / NOR gates from *behavioral Verilog* models, which represent the estimated overall delays of the MESO device along with their intended functionality. The MESO primitive's function morphs based on the value of the voltage terminals and the control signal X. Toggling X allows one to morph between OR ↔ AND and NOR ↔ NAND. Morphing between OR and NOR involves setting the top/bottom voltage terminals as V^-/V^+ or V^+/V^-; the converse is true for AND ↔ NAND morphing. Note that the morphing time is included in the total switching time of ~258 ps, in the form of the peripheral MUX delay.

The design house can either provide a fully-camouflaged layout composed of only MESO devices, or a camouflaged

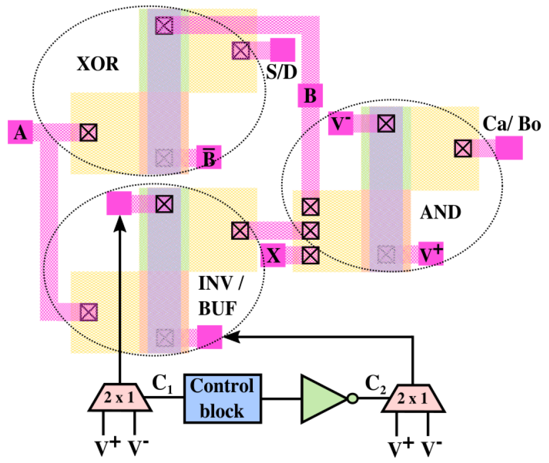


Fig. 5. MESO adder/subtractor highlighting the capabilities for *functional reconfiguration*. The XOR and AND gates are implemented as static MESO gates and the INV / BUF is a polymorphic MESO gate whose function is derived from control bits (C_1 and C_2) fed by a simple control block. *A* and *B* are the inputs, *S* is sum, *D* is difference, *Ca* is carry, and *Bo* is borrow. Note that dummy contacts are omitted here for the sake of simplicity.

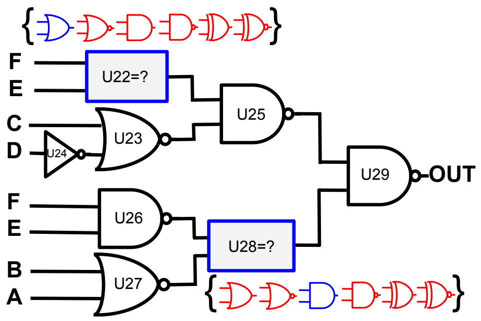


Fig. 6. An example circuit where two gates (U22 and U28) are camouflaged. The camouflaged gates can assume any one of the outlined six 2-input functions. The correct functionality of these camouflaged gates is shown in blue. As the functionality of MESO gates can be reconfigured *post-fabrication*, an attacker's effort of inferring the exact functionality is hindered. With a random gate-guessing attack, an attacker has 36 possible netlists to consider, with only one amongst them being correct.

layout where selected CMOS gates are replaced by MESO gates. The MESO device is compatible with CMOS processes in the BEOL, enabling heterogeneous integration.[2] The proportion of the design camouflaged by a designer depends on the scope of application and impact of camouflaging on PPA overheads. The replacement of logic gates can also be performed in a manner conducive to protecting the critical infrastructure (i.e., design secrets, proprietary IP).

Please note that the MESO-based primitive can also be leveraged for *static camouflaging*. In such a scenario, the peripheral circuitry (Fig. 3) dictating the functionality of the MESO device shall be fed with fixed control bits and control signals. Static camouflaging using spin devices has been explored in prior works; interested readers are referred to [23], [24], [25] for further details.

4 Security Analysis: Untrusted Foundry

An attacker in the foundry can readily infer the IP implemented in CMOS, whereas the MESO gates appear as white-box devices, albeit *without any fixed functionality*. The MESO implementation of Boolean gates is optically indistinguishable concerning their physical layout (Fig. 5), which renders optical inspection-guided RE difficult. Since our approach here relies on *post-fabrication reconfigurability*, it is intuitive that our scheme is resilient to "inside foundry" attacks. As shown in Fig. 6, the *post-fabrication reconfigurability* of MESO gates hinders the attacker's effort to infer the exact functionality. A random gate-guessing attack on the circuit shown in Fig. 6 has a solution space of 36 possible netlists, with only one amongst them being correct.

4.1 Threat Model

The threat model which we adopt for security analysis for an untrusted foundry is outlined as follows:

- A malevolent employee in the foundry has access to the physical design, including material and layout parameters of the MESO gates and the chip interconnects. While an adversary in a foundry can readily obtain the dimensions and material composition of the nanomagnet in each MESO gate and, hence, understand its magnetic properties including saturation magnetization, energy barrier, and critical ME field for switching, these design details do not leak any information about the intended functionality implemented by the gate.
- He/she is aware of the underlying gate selection algorithm, number, and type of camouflaged gates, but is oblivious to the actual functionality implemented by the camouflaged gate.
- For security analysis, we assume that the working chip is not yet available in the open market. Thus, he/she has to apply "inside foundry" attacks which are explained briefly next.

4.2 Attack Model

Recently, researchers have proposed attacks [36], [37], which can be carried out within the confines of an untrusted foundry. These attacks do not leverage an *activated working chip* as an oracle, which is in contrast with algorithmic SAT-based attacks [9], [10], [27]. Though these attacks have been primarily tailored toward LL, we believe these would readily apply on LC schemes as well, given that any LL problem can be modeled as an LC scheme and vice-versa. Besides, for the attacks proposed in [36], [37], the basic premise is that an incorrect assignment of key-bits involves significant logic redundancies compared to the correct assignment of key-bits. The attack in [37] determines the likely value of key-bits individually by comparing the levels of logic redundancy for each logic value.

Example: We illustrate the effect of an incorrect assignment of key-bits (gates), leading to logic redundancy using a simple example. Consider the circuit shown in Fig. 7(a), logic gates U31 and U33 are camouflaged using a NAND/NOR camouflaging primitive, which leads to four combinations for [U31, U33]. The circuits are shown in

2. In general, hybrid *spin-CMOS* designs have been explored in a prior work, e.g., [35].

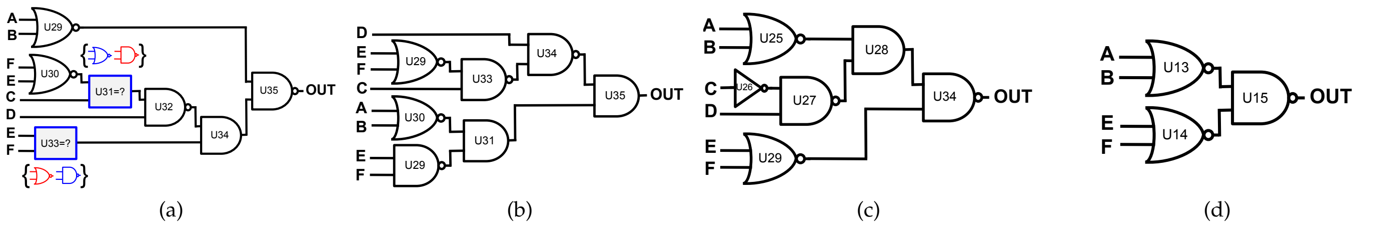


Fig. 7. An illustration of an incorrect gate assignment leading to logic redundancy. In (a), gates U31 and U33 are camouflaged using a simple NAND/NOR primitive, giving rise to four possible options. Correct assignment of camouflaged gates is shown in blue. Incorrect assignment of gates leads to circuit configurations (b), (c), and (d), respectively. Note the reduction of gates in (c) and (d) compared to (a), while the gate count is identical in (a) and (b), albeit (b) functions differently than (a).

Fig. 7(b–d), and they correspond to scenarios [U31 = NAND, U33 = NAND], [U31 = NAND, U33 = NOR], and [U31 = NOR, U33 = NOR], respectively. After re-synthesis, an incorrect combination of gates deciphered by an attacker leads to circuits with fewer gates (Fig. 7(c) and Fig. 7(d)) when compared to the original circuit. We also note that an attacker might end up with cases like that of Fig. 7(b), where the total number of gates is same as the original circuit; however, these circuits differ in functionality.

Having no access to these attacks [36], [37], we refrain from a direct, independent comparison. However, for the sake of completeness of the security analysis, we perform quantitative experiments, based on the essence of findings quoted in the respective works of [36], [37]. For example, the *desynthesis* attack [36] can correctly infer 23 (up to 29) and 47 (up to 59) key-bits for 32 and 64 key-gates, respectively, while the authors of [37] report success rate in 25%–75% percentile distribution. For a fair comparison, we consider similar ranges of correctly inferred gates.[3]

4.3 Experimental Setup

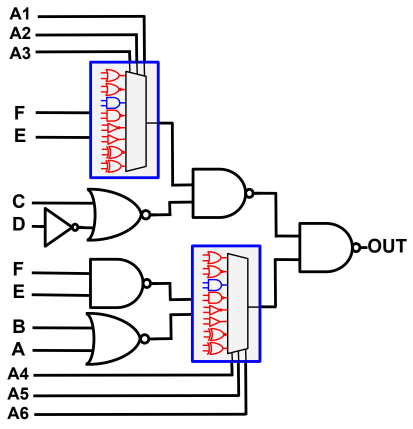


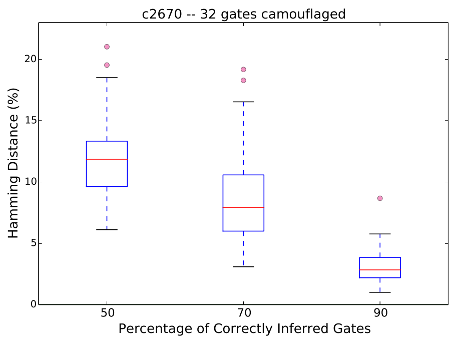
Fig. 8. Modeling of camouflaged circuits using MUXes. Each camouflaged gate is replaced with a corresponding MUX which dictates the functionality based on the value assigned to the select inputs (A1–A6). In this example, the camouflaged cell can implement any one of the eight functions viz. OR, NOR, AND, NAND, INV, BUF, XOR, and XNOR. This modeling has been used throughout the paper.

3. Note that this is a powerful assumption for the attacker's capabilities. This is because the respective attacks [36], [37] tackle LL schemes, where modeling a locked gate requires only one key-bit, whereas for multi-function camouflaging schemes like ours, multiple key-bits are required for modelling one camouflaged gate (Fig. 8).

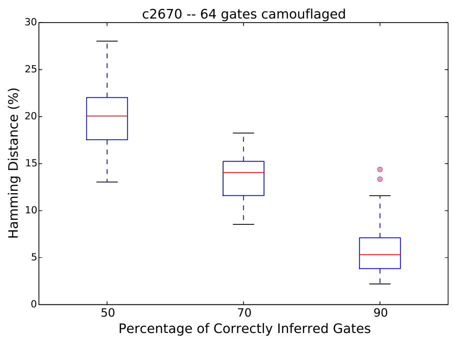
We model the MESO primitive, as shown in Fig. 8. The logical inputs a and b are fed in parallel into all eight possible Boolean functions, and outputs of those gates are connected to an 8-to-1 MUX with three select lines/key-bits. For a fair evaluation, we camouflage the same set of gates for the ISCAS-85 benchmarks c5315 and c7552. Gates are chosen *randomly* at the beginning and then memorized. Ten such sets are created for each benchmark. To emulate the attack results from [36], [37], we employ the following procedure. We implement a script which randomly picks the correct assignment amongst the camouflaged gates such that we obtain three sets, each corresponding to 50%, 70%, and 90% correctly inferred gates. This procedure is repeated ten times, each for ten different iterations of camouflaged gates, giving us 100 unique trials. Our camouflaging scheme has been implemented using *Python* scripts operating on *Verilog* files. Hamming distance (HD) is computed leveraging *Synopsys VCS* with 100,000 input patterns and, functional correctness is ascertained by *Synopsys Formality*.

4.4 Results

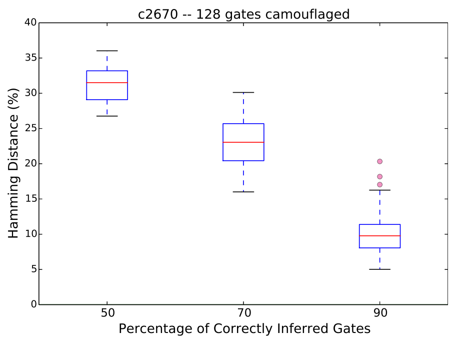
Once we ascertain the percentage of correctly inferred gates for different levels of attack accuracy (50%, 70%, and 90%), we calculate the HD between the reconstructed and the golden netlist. The results are shown as box-plots in Fig. 9 for two ISCAS-85 benchmarks c5315 and c7552. It is intuitive to note that, as the percentage of the correctly inferred gates is increased, there is a steady reduction in the HD, which also hints that the reconstructed netlist becomes functionally similar to the original circuit. For the ISCAS-85 benchmark c7552, assuming an attack accuracy of 90%, the mean HD increases from about 2% when 32 gates are camouflaged (29 are inferred correctly) to 5% when 128 gates are camouflaged (115 are inferred correctly). Note that such HD numbers could already suffice for an attacker recovering an approximate version of the original functionality. However, for attacks which can only recover 50–70% of the total camouflaged gates, the HD for the reconstructed circuit is between 6% to 25%, depending on the size and type of the benchmark, the number of gates being camouflaged, and the number of gates correctly inferred. These findings also imply that camouflaging large parts of a design might suffice to thwart "inside foundry" attacks, which we confirm by a simple experiment as discussed next. For a larger ITC-99 benchmark like b22_C, we camouflage 50% of the total logic gates (7,228 gates) present in the overall design.



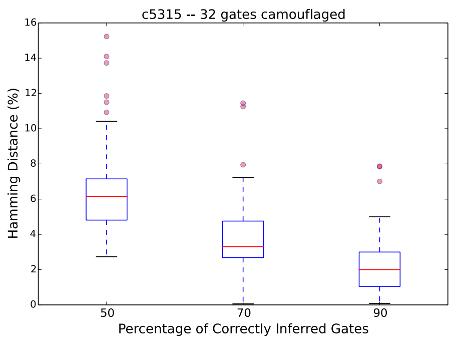
(a)



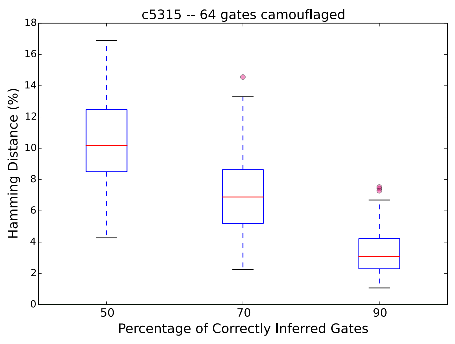
(b)



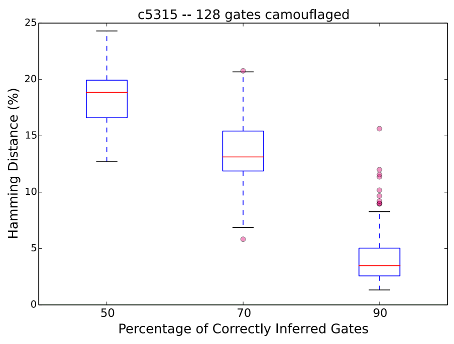
(c)



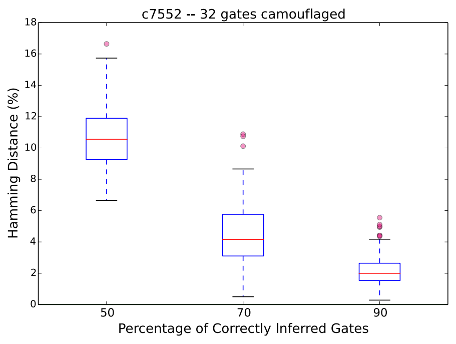
(d)



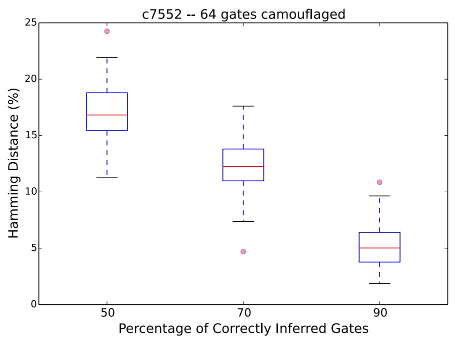
(e)



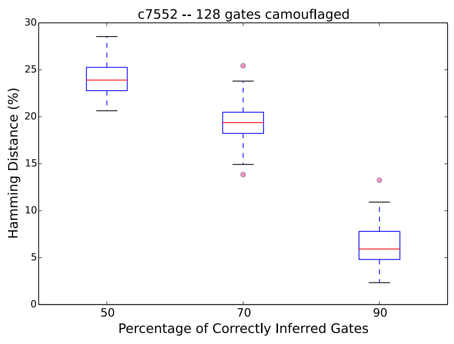
(f)



(g)



(h)



(i)

Fig. 9. Hamming distance (HD) plotted against the percentage of correctly inferred gates (50%, 70%, and 90% of total camouflaged gates) of different sets of camouflaged gates, on selected ISCAS-85 benchmarks c2670, c5315 and c7552. Mean HD is proportional to the number of correctly inferred gates amongst the total number of camouflaged gates. Each box comprises data for 100 trials of random selection of gates to camouflage.

Assuming that an attacker can identify 90% of these gates correctly, this still leaves 722 gates wrongly inferred, which yields an HD of 43% (across ten random trials). Overall, the property of *post-fabrication reconfigurability* for the MESO gates allow us to change the functionality, enabling superior security through dynamic camouflaging.

5 Security Analysis: Untrusted Test Facility

Attackers present in the test facility having access to test patterns and corresponding output responses (generated and supplied by the trusted design house), can jeopardize the security guarantees offered by LL and LC. Modern Automatic Test Pattern Generation (ATPG) algorithms have been designed to maximize the fault coverage (FC) with minimal test pattern count, which directly translates to a lower test cost. Such an approach, however, divulges critical information pertaining to the internal circuit specifics [11].

In the context of VLSI testing principles, detection of a stuck-at-fault involves two principal components, namely (i) fault activation and (ii) fault propagation. In *fault activation*, a faulty node is assigned a value opposite to the fault induced on that particular node. Consider the example shown in Fig. 10; here, the output of logic gate U4 is *s-a-1* (stuck-at-1). In order to detect this fault, fault activation is achieved by setting this node to logic '0'. Next, *fault propagation* entails propagating the effect of the fault along a sensitization path to one of the primary outputs. To achieve fault propagation (here to O2), the output of U3 must be '1'. An input pattern which can detect a fault at a given node by achieving the effects mentioned above is defined as a *test pattern*. In Fig. 10, the input pattern *11001* and the corresponding output response *11* is supplied to the test facility, among others.

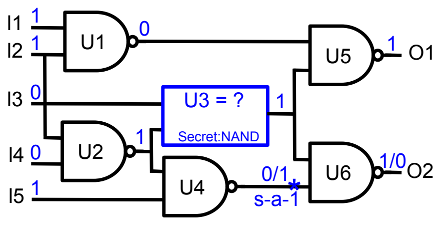


Fig. 10. An input pattern which helps in the detection of stuck-at-1 fault at the output of U4. The circuit output '1/0' at output O2 indicates the response for fault-free/faulty circuit are 1 and 0, respectively. The input pattern 11001, along with the expected output response 11 is provided to the test facility for testing manufactured ICs. The test data hints that U3 cannot be NOR. Note that, here we assume that the camouflaged gate can function only as NAND/NOR.

5.1 Threat Model

Apart from outsourcing of chip fabrication, many design companies also outsource the testing phase to off-shore companies such as Amkor, ASE, SPIL, etc. [11]. The implications of an untrusted test facility in the supply chain have been explored in the context of LL [5] and static LC [11]. However, there has been no thorough analysis yet on the efficacy of test data-based attacks [11] for different static camouflaging schemes as well as dynamic camouflaging. In our threat model, the attacker resides in the test facility and has access to the following assets:

- Gate-level camouflaged netlist, e.g., obtained by RE.
- Knowledge of the test infrastructure, which includes identification of scan chains, compressor, decompressor, et cetera, on the target chip.
- Test patterns and their corresponding output responses, which have been provided by the design house. He/she also has access to ATPG tools used to generate the test patterns.

5.2 Attack Model

Yasin *et al.* [11] proposed *HackTest*, which revealed the true identity of camouflaged gates within minutes by exploiting test data. The attack leverages the fact that the generation of test patterns is typically tuned to obtain the highest possible FC. Hence, given the test stimuli and responses, an attacker can search over the key space (using optimization techniques) to infer the correct assignment of camouflaged gates which maximizes the FC. Arguably, such an attack is more powerful than SAT-based attacks [9], [10] which require access to a working chip.

The process of ATPG is highly dependent on the internal specifics of the underlying circuit, which include the type and count of gates, the inter-connectivity amongst these gates, etc. Years of research have yielded powerful algorithms which lower the test pattern count while achieving a high FC. However, these algorithms do not factor in security (yet), and thereby, test patterns become a rich source of information for an opportunistic attacker. Next, we briefly explain the notion of *HackTest* with a simple example; interested readers are kindly referred to [11] for further details.

Example: Upon performing ATPG for the circuit shown in Fig. 6, for the correct assignment of two camouflaged gates (U22 = OR and U28 = AND), eight test patterns are generated by *Synopsys Tetramax*, providing a fault and test coverage of 100%. Camouflaging two gates with two functions each gives rise to four possible circuit configurations; Table 2 denotes the FC for these configurations. Armed with input patterns and corresponding output responses, both tailored for the correct assignment, an attacker calculates the FC for all possible circuit configurations. As shown in Table 2, maximal FC is observed only for the correct assignment of camouflaged gates. This is because, for *static camouflaging*, test patterns have to be generated for the correct assignment of camouflaged gates. An attacker can easily use FC to guide his/her attack to identify the correct functionality of camouflaged gates.

TABLE 2
Fault coverage achieved for different assignments to the camouflaged netlist in Fig. 6. Here we assume U22 and U28 are implementing either AND/OR. The correct assignment is OR and AND, respectively; note that other assignments result in significantly lower fault coverage.

U22	U28	Fault Coverage (%)
AND	AND	63.33
AND	OR	38.33
OR	AND	100
OR	OR	78.33

5.3 Experimental Setup

We launch *HackTest* on selected benchmarks of ISCAS-85 and ITC-99 suite. Statistics of benchmarks like the number of logic gates (# Gates), number of faults (# Faults), number of test patterns generated by *Synopsys Tetramax* (# Test patterns), and corresponding FC are shown in Table 3. We implement the MESO-based camouflaging primitive along with some selected prior art [16], [22]. As *HackTest* requires a *BENCH* file format, we employ custom scripts to convert *Verilog* files to required formats. For the small-scale ISCAS-85 benchmarks, we prepare ten random sets each for camouflaging 32, 64, and 128 gates, respectively. For the large-scale ITC-99 benchmarks, we camouflage 350 gates. For the sake of uniformity in comparison, the selection of camouflaged gates is random but fixed, i.e., they are common and maintained across all benchmarks for any given camouflaging scheme. The attack is implemented using custom *Python* scripts executing within *Synopsys Tetramax*. All attack experiments are carried out on an Intel Xeon E5-4660 @ 2.2 GHz with *CentOS 6.9* and the time-out (t-o) is set to 24 hours.

5.4 Results

Next, we detail our observations on employing *HackTest* for various test cases. We begin by examining the impact of *HackTest* on various static camouflaging schemes. Finally, we enumerate our findings concerning the resiliency of *dynamic camouflaging*, which is the main focus of this work.

5.4.1 *HackTest on Static Camouflaging*

Yasin *et al.* [11] demonstrated the efficacy of *HackTest* on benchmarks camouflaged with 32/64 gates employed with

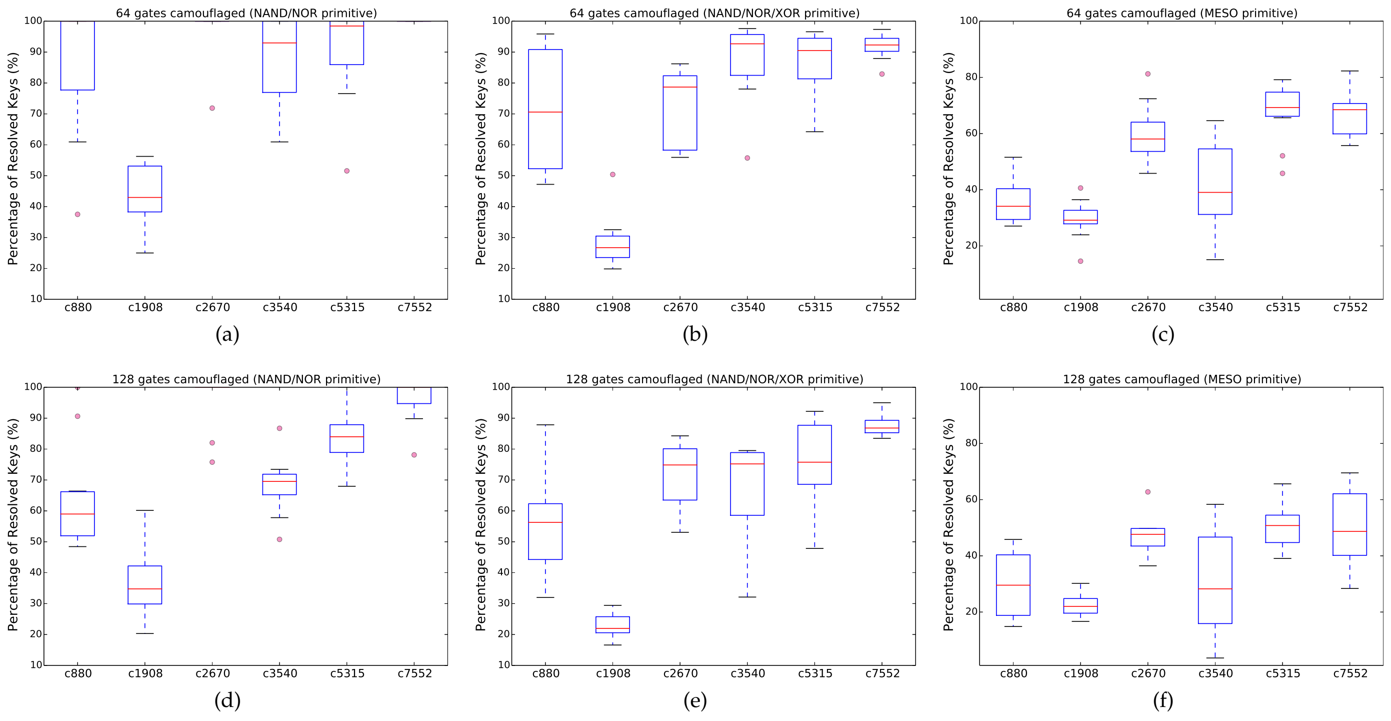


Fig. 11. Percentage of key-bits resolved by *HackTest* for static LC of different sets of camouflaged gates, on selected ISCAS-85 benchmarks. The top three box-plots denote 64 camouflaged gates, while bottom three box-plots denote 128 camouflaged gates for three camouflaging schemes: NAND/NOR, NAND/NOR/XOR, and MESO primitive (implementing 8 functions, Fig. 2). Each box comprises data for 10 trials of random selection of gates to camouflage.

TABLE 3
Statistics of ISCAS-85 and ITC-99 benchmarks used in this work. All benchmarks achieve 100% test coverage and achieve exactly/close to 100% fault coverage.

Benchmark	# Gates	# Faults	# Test Patterns	Fault Coverage (%)
c880	273	1,764	63	100
c1908	230	1,462	80	100
c2670	433	2,936	134	100
c3540	814	5,472	177	100
c5315	1,232	7,708	124	100
c7552	1,197	7,474	167	100
b14_C	4,125	24,668	470	99.99
b15_C	6,978	42,310	812	99.96
b20_C	9,226	54,894	897	99.89
b22_C	14,457	85,852	1,356	99.95

TABLE 4
Impact of *HackTest* on MESO-based static camouflaging on Hamming distance (HD) and Output error rate (OER) for selected ISCAS-85 benchmarks with 128 camouflaged gates. HD and OER are averaged across 10 random trials of camouflaged gates.

Benchmark	# Camo. Gates	# Correctly Inferred	HD (%)	OER (%)
c880	128	23	47	100
c1908	128	29	46	100
c2670	128	51	28	100
c3540	128	39	47	100
c5315	128	65	23	100
c7552	128	64	24	100
Average	**128**	**45**	**35.8**	**100**

NAND/NOR camouflaged cells. For the sake of completeness, we implement this scheme along with a few others. The success rate for *HackTest* [11] is reported as the percentage of key-bits inferred by the attack.

From the box-plots with 64 gates camouflaged (Fig. 11 (a), (b), and (c)), the attack's complexity is demonstrated with the increase in the number of functions implemented by a single camouflaged gate. For the NAND-NOR camouflaging scheme, *HackTest* performs extremely well; all ten random iterations of ISCAS-85 benchmark c7552 can be decamouflaged correctly. We observe a high accuracy rate for other benchmarks as well, except for *c1908*. Though the overall accuracy remains high for the NAND-NOR-XOR camouflaging scheme [16], it fails to compete, especially when compared to the NAND-NOR camouflaging scheme. For the MESO-based static camouflaging scheme, which supports eight functions, a stark reduction in attack's efficiency is observed. From the box-plots with 128 gates camouflaged (Fig. 11 (d), (e), and (f)), the attack's complexity is demonstrated with an increase in the number of camouflaged gates (w.r.t. Fig. 11 (a), (b), and (c)). The overall success rate is lower for all the camouflaging schemes, hinting on the fact that the attack's success rate is proportional to the total number of camouflaged gates and the number of correctly/incorrectly inferred gates.

We also analyze the effect of incorrect key-bits on security metrics HD and Output Error rate (OER) for MESO-based static camouflaging primitive; results are shown in Table 4. The HD for benchmarks c880, c1908, and c3540 approach the ideal value of 50%, while the values are around 25% for benchmarks c2670, c5315, and c7552. The OER, however, is 100% for all the designs. The decrease in HD for benchmarks c5315 and c7552 can be ascertained to the fact that the number of wrongly inferred gates form a very small

TABLE 5
Impact of increasing the number of possible functions implemented by the MESO-based primitive on *HackTest*'s accuracy for selected ITC-99 benchmarks. Test patterns are generated by *Tetramax ATPG* for fault coverage and test coverage of 99% and 100%, respectively. Results are averaged across 10 random trials of camouflaged gates.

Benchmark	# Camo. Gates	3 functions	4 functions	8 functions	16 functions
b14_C	350	20.37	15.13	14.69	11.49
b15_C	350	11.47	10.4	8.58	7.23
b20_C	350	17.03	14.03	11.11	8.51
b22_C	350	27.03	21.52	15.33	12.48
Average	350	18.98	15.27	12.43	9.93

TABLE 6
Impact of *HackTest* for MESO-based static (S. Camo) and dynamic camouflaging (D. Camo) schemes on HD and OER for selected ITC-99 benchmarks. The number of wrongly inferred gates for dynamic camouflaging is higher on average when compared to static camouflaging, which translates to an improved HD. HD and OER are calculated by averaging across 10 random trials.

Benchmark	# Wrongly Inferred Gates		HD (%)		OER (%)	
	S. Camo.	D. Camo.	S. Camo.	D. Camo.	S. Camo.	D. Camo.
b14_C	249	298	36.04	42.09	100	100
b15_C	296	320	32.07	34.23	100	100
b20_C	279	310	32.15	35.03	100	100
b22_C	212	296	20.09	29.57	100	100
Average	259	306	30.09	35.23	100	100

portion of the overall design. For example, we camouflage 128 gates out of 1,197 gates for c7552 which forms about 10.69% of the overall design. *HackTest* resolves 64 gates correctly, bringing the percentage of wrongly inferred gates to 5.35%. Similarly, camouflaging 128 gates out of 273 for c880 forms 46.89% of the design. *HackTest* resolves only 23 gates correctly, which increases the proportion of wrongly inferred gates to 38.46%, which is higher than c7552.

To summarize, we observe that the efficiency of *HackTest* is directly proportional to (i) size and type of the benchmark, (ii) number and type of camouflaged gates, and (iii) number of functions implemented by a camouflaged gate.

5.4.2 *HackTest on Dynamic Camouflaging*

As elucidated before, none of the static camouflaging approaches [16], [18] allow for post-fabrication reconfiguration and, hence, test patterns are generated for the correct assignment of camouflaged gates. MESO-based dynamic camouflaging circumvents this threat by allowing for *post-test configuration*. That is, the fabricated IC can be initially configured with an incorrect I/O mapping and functionality. The "falsely configured" IC and related test data are then sent to the test facility.[4] Accordingly, an attacker will end up with an *incorrect* IP when mounting *HackTest* on the IC.[5] After testing is finished, the MESO gates are reconfigured (by the design house or some trusted entity) to reflect the true, intended functionality.

Table 5 details the effect of increasing the number of possible functions implemented by the MESO-based primitive on *HackTest*'s accuracy. We observe that the attack accuracy reduces (for the same set of camouflaged gates) when the number of functions implemented by the MESO-based primitive is increased. This can be reasoned from the fact that, with an increase in the number of possible functions, the attack has a larger solution space to tackle.

Finally, we also examine the security promises for both static and dynamic camouflaging for the MESO-based primitive; results are shown in Table 6. It can be seen that the number of wrongly inferred gates is higher for dynamic camouflaging. This increase also translates to a higher HD (about 5.14%). The OER, however, remains at 100% for both the schemes. Figure 12 shows the dependence of *HackTest*'s success rate as a function of HD for selected ITC-99 benchmarks. This plot reiterates that the degree of functional reconfiguration (measured as HD) has a strong impact on the accuracy of *HackTest*.

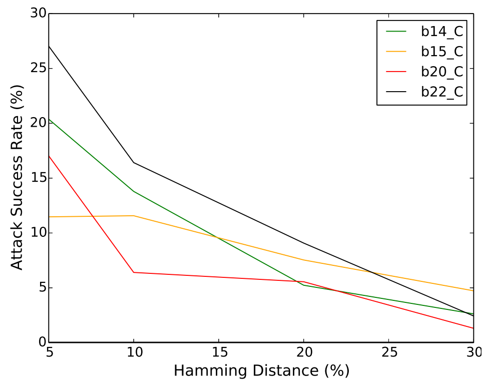


Fig. 12. Success rate of *HackTest* as a function of HD for selected ITC-99 benchmarks. It is evident that the degree of functional reconfiguration, also expressed by HD, can be leveraged by a designer to influence the overall success rate for *HackTest*.

6 Security Analysis: Untrusted End-User

6.1 Threat Model

The threat model which we employ for security analysis for an untrusted end-user follows very closely to the ones described in the literature [9], [10], [18].

- The attacker has access to advanced, specialized equipment for reverse engineering an IC, which includes setup to depackage an IC, delayer it, imaging of individual layers, and image-processing tools.
- Further, he/she can readily distinguish between a camouflaged cell and a regular, standard cell. If hybrid spin-CMOS circuits are used, it is straightforward to identify the CMOS gates, whereas the complexity is increased manifold, if all the gates are implemented using MESO devices.
- The attacker is aware of the total number of camouflaged gates, and the number and type of functions implemented by each camouflaged cell.
- He/she procures multiple chip copies from the open market, uses one of them as an oracle (to observe the input-output mapping), and extracts the gate-level netlist of the chip by reverse engineering the others. This paves the way for algorithmic SAT-based attacks [9], [10], [27].

4. Testing for structural defects does not require the chip to be functional; chips can be configured to any function and tested with no loss in test quality [5], [11].

5. This resonates with the idea of *post-test activation* [5], which is the adopted strategy for safeguarding against untrusted test facilities in logic locking.

- Consistent with the most prior art, we assume that an attacker cannot invasively probe the output of a camouflaged cell.[6] It is straightforward to note that once an adversary is allowed probing capabilities, i.e., to probe the output of a camouflaged cell (or read out contents from a TPM for locking), then the security guarantees offered by these schemes are substantially weakened, if not even nullified.

An attacker may also try to observe various side-channels like power, timing, photonic, acoustic, etc. However, note that we do not consider the effect of side-channels emission from the MESO switch in this work; this remains part of our future work, once efficient circuit- and/or layout-level models are available for MESO gates.

6.2 Attack Model and Setup

In 2015, Subramanyan *et al.* [9] and Massad *et al.* [10] independently demonstrated SAT-based attacks to circumvent security guarantees offered by LL and LC, respectively. Interested readers are referred to the respective papers for further details. We leverage the publicly available attack [9] to perform the security analysis for an untrusted end-user.

6.3 Results and Discussion

Next, we explain how dynamic camouflaging can thwart attacks arising from the perspective of malicious end-users.[7] We illustrate the related concept of *run-time polymorphism for dynamic morphing* through a conceptual example; consider the circuit in Fig. 13. Here, $X4$ is the only camouflaged, polymorphic gate modeled with three key-bits. Assuming a key distribution such that INV, BUF, AND, OR, NAND, NOR, XOR, and XNOR gates correspond to key-bits $\{000, 001, ..., 111\}$, respectively, the dynamic key of the circuit cycles from 100 to 101, and then to 111, as per the outlined functional reconfiguration in Fig. 13.

6. We acknowledge that there is an attack proposed by Keshavarz *et al.* [38], where an SAT-based formulation is augmented with probing and fault-injection capabilities to reverse engineer a relatively small *S-Box*. Still, it remains to be seen how this attack would fare when large-scale camouflaging is effected. Having no access to this attack at the time of writing, we refrain from any empirical analysis.

7. The MESO-based primitive can also be leveraged in the context of static camouflaging to protect against malicious end-users. It has been shown empirically in prior studies [6], [10], [24], [25] that large-scale camouflaging with cells implementing more functions pose significant computational complexity for such attackers.

The application of the SAT-based attack [9], [10] for the simple scenario in Fig. 13 is explained next (Fig. 14). Consider that the oracle (i.e., an actual working chip obtained from the market) implements f_1 during the first iteration of the SAT solver, where the input applied is 101. Note that the oracle is to be configured for test mode, to provide access to the circuit internals through scan chains, as required when modeling the whole circuit for the SAT-based attack. In principle, the oracle may behave differently in the test mode and in the operational (functional) mode. Naturally, the SAT solver is oblivious to the function being active internally in the oracle during *any* iteration. Also note that, once inputs are applied to the oracle, the SAT solver has to wait until the oracle provides the corresponding outputs. Now, that first SAT iteration prunes key combinations k_0, k_2, k_5, and k_7. While this is happening, assume that the gate $X4$ has morphed into NOR, and the oracle is now implementing function f_2. In the second SAT iteration, the input pattern 100, therefore, eliminates keys k_3, k_4, and k_6. Thereafter, the SAT solver concludes that the correct key bit and identity of gate $X4$ are 001 and BUF, respectively.

In essence, dynamic camouflaging can deceive and mislead the SAT solver to converge to an *incorrect key*, leading to an *incorrect* gate assignment. For an exploratory study, we extend the framework from [9] to realize SAT-based attacks on polymorphic versions of ITC-99 benchmarks. Even for 100,000 randomized trials, the related attacks fail due to inconsistent I/O mappings, as these induce *unsatisfiable* (*UNSAT*) scenarios for the attack framework.

Taking this simple example from Fig. 13 further, for error-tolerant applications like image/video processing, the circuit may indeed be reconfigured randomly, e.g., by deriving the control bits of the MESO gates from a TRNG—we present results for *AppSAT* [27] on such error-tolerant applications in Section 7.3.

Besides SAT-based attacks, when concerned about *physical attacks* conduced by an end-user, one has to ensure that the interconnect fabric which routes the control bits and control signals to the MESO gates is resilient against probing; e.g., shielding may be used toward that end [39]. *Removal attacks* targeting the TRNG shall result in floating controls for the MESO gates, leading to noisy outputs, and loss of functionality (as well as hindrance of SAT-based attacks discussed above). Other advanced attacks, e.g., directed at distorting the entropy of the TRNG to change its bias [40], are considered out-of-scope for this work.

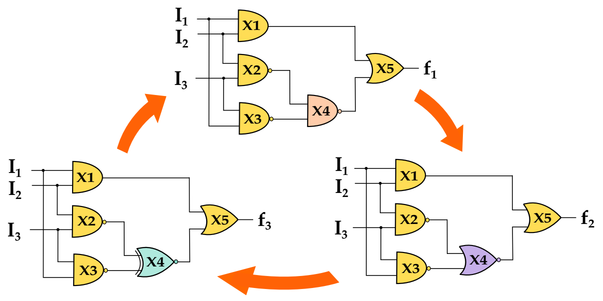
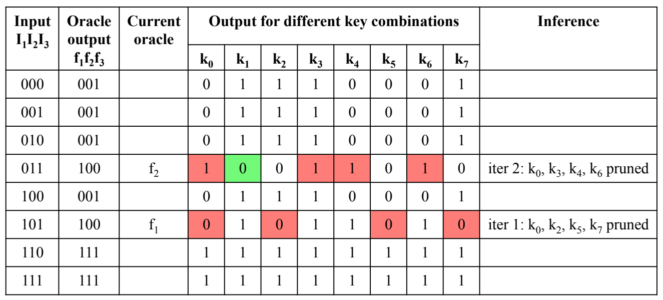


Fig. 13. Dynamic morphing of gate X4 in a representative circuit. Circuit implementing f_1 is the original template and f_2, f_3 are the morphed versions.



Input $I_1I_2I_3$	Oracle output $f_1f_2f_3$	Current oracle	Output for different key combinations								Inference
			k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	
000	001		0	1	1	1	0	0	0	1	
001	001		0	1	1	1	0	0	0	1	
010	001		0	1	1	1	0	0	0	1	
011	100	f_2	1	0	0	1	1	0	1	0	iter 2: k_0, k_3, k_4, k_6 pruned
100	001		0	1	1	1	0	0	0	1	
101	100	f_1	0	1	0	1	1	0	1	0	iter 1: k_0, k_2, k_5, k_7 pruned
110	111		1	1	1	1	1	1	1	1	
111	111		1	1	1	1	1	1	1	1	

Fig. 14. SAT-based attack [9] on the polymorphic circuit of Fig. 13. For k_0 and k_1, the INV and BUF operations performed on the output of $X2$.

7 Case Study: MESO CeNN-Based Approximate Image-Processing IP

In this section, we demonstrate how dynamic camouflaging can help in protecting approximate, error-tolerant circuits. We design a cellular neural network (*CeNN*) using MESO gates. *CeNNs* are a massively parallel neural network-based computing paradigm, which consist of an n-dimensional array of locally interconnected cells that communicate within a neighbourhood. They are typically used in a variety of applications including image filtering and reconstruction, edge detection, solving partial differential equations and optimization problems. The cells of a CeNN are multiple-input single-output processors, characterized by an internal state variable. These processing cells act as neurons that integrate the input currents, and the interconnects between the cells act as synapses that perform weighting of the inputs. The dynamical state equation for a CeNN neuronal cell, put forth by Chua and Yang *et al.* [41], is as follows:

$$C\frac{dx_{ij}}{dt} = -\frac{1}{R}x_{ij} + \sum_{kl} A(i,j;k,l)f(x_{kl}) + \sum_{kl} B(i,j;k,l)U_{kl} + I_{ij} \qquad (2)$$

where x_{ij} is the internal state of a neuron, $\{i,j\}$ represent the neighbourhood of the neuron, A and B are the synaptic weights connecting two neighbouring cells, I is a constant bias current, R and C are the resistance and capacitance of the cell, U_{kl} and $f(x_{kl})$ are the input and state-dependent output of the cells, respectively.

As demonstrated in [42], spintronic devices are able to directly implement a CeNN for image-processing applications, without the need for analog VLSI elements. We chose CeNN as a representative example to highlight the application of dynamic camouflaging in real scenarios owing to the fact that it can function as a relatively simple and low-cost image-processing circuit, with a single layer of input cells. However, the concept of dynamic camouflaging can be extended to any approximate IP, without loss of generality.

7.1 Construction

We adopt the same methodology for the construction of the magnetic synapses and neuron of the CeNN as in [42]. However, we design the neuron cells using the MESO device instead of the all-spin logic device used in [42]. The parameters for the MESO device used for the CeNN cells are obtained from [4]. Fig. 15 (a) highlights the connectivity of cells in the MESO CeNN and Fig. 15 (b) shows the construction of the MESO CeNN. The transient switching of a MESO CeNN cell along with the CeNN templates {A, B, I} used for simple image reconstruction (from [42]) are portrayed in Fig. 15 (c). The central MESO device is connected to eight other MESO devices in a 3×3 grid. The weighting operation can be realized by (i) using a layer of CMOS transistors with different driving strengths, in between the input and output layers, as demonstrated in [42], or by (ii) inserting multi-terminal magnetic domain wall (DW) weighting devices [43] in the interconnects between the input and output layers. Both these weighting mechanisms are able to implement

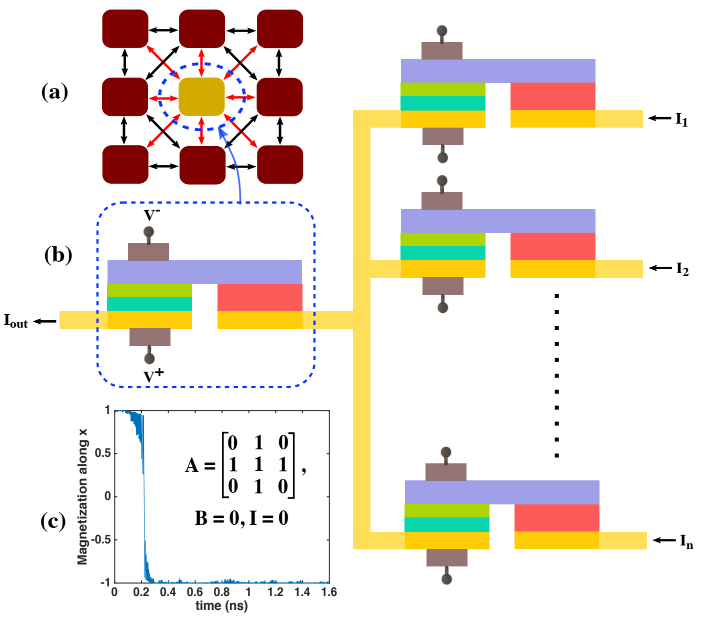


Fig. 15. (a) Inter-connectivity of cells in the MESO-based Cellular Neural Network (CeNN). (b) Construction of the MESO CeNN for image reconstruction. Each cell in the network is implemented by a MESO device. (c) Magnetization vs. time shows the switching of the central MESO CeNN cell, when inputs from its nearest neighbor cells are applied. The switching delay is ~ 200 ps. The MESO CeNN is simulated using a Landau-Lifshitz-Gilbert dynamics framework on *CUDA-C* [44], with 1,000 nanomagnet simulations per cell. Inset shows templates $\{A, B, I\}$ used to configure the CeNN.

several levels of weights for precise image-processing applications. We note that the former approach also requires additional transduction circuitry for converting the current signals from the input layers, into voltage signals that can be fed to the CMOS driving transistors. In the case of the DW weighting devices, dedicated programming terminals are used to set the position of the DW and control the conductance (weight) of the device, which then scales the current passing through its input terminals. Readers are referred to the respective papers for further details. The weighting units are omitted from Fig. 15 (b) for simplicity.

7.2 Experimental Setup

We investigate the implications of attacking an approximate image-processing IP with *AppSAT* [27]. Approximate circuits are vulnerable to such attacks since the attacker can recover a functionally-similar IP. Since the original circuit is approximate to begin with, an attacker might be satisfied by obtaining an IP which has, say, 95% fidelity compared to the original design. For example, consider the case of an approximate image reconstruction hardware module. The application of *AppSAT* on this module may give an attacker a functionally-similar circuit and, if needed, he/she could then augment this reverse engineered module with software ML models to obtain a precision equivalent to the original image reconstruction hardware. In this section, we show that the *AppSAT*-recovered IP of an approximate circuit can deviate significantly from the original IP, enough to render such an attack futile.

To safeguard the MESO-based CeNN against *AppSAT*, we use *run-time polymorphism* for dynamic morphing. This

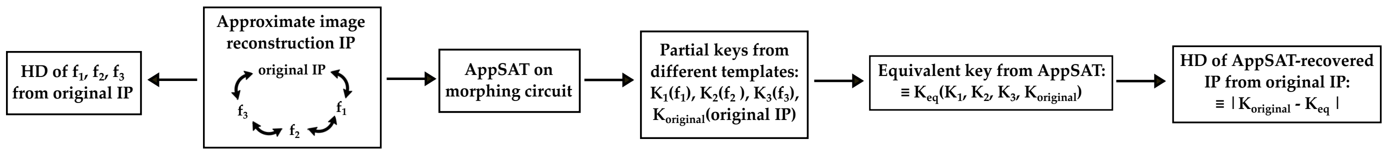

HD of f1, f2, f3 from original IP
Approximate image reconstruction IP
original IP
f1
f2
f3
AppSAT on morphing circuit
Partial keys from different templates: K1(f1), K2(f2), K3(f3), Koriginal(original IP)
Equivalent key from AppSAT: ≡ Keq(K1, K2, K3, Koriginal)
HD of AppSAT-recovered IP from original IP: ≡ | Koriginal - Keq |

Fig. 16. Flowchart illustrating the application of *AppSAT* on a dynamically camouflaged approximate image-processing IP. *AppSAT* recovers partial keys from the different circuit templates, and the equivalent stitched key deviates significantly from the original IP.

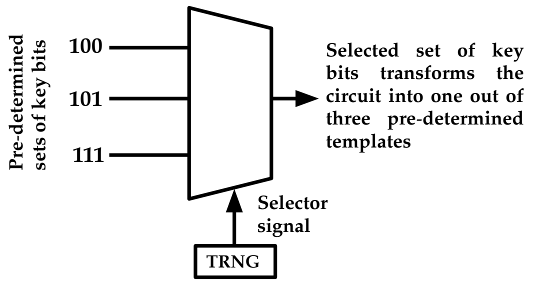

Pre-determined sets of key bits
100
101
111
Selected set of key bits transforms the circuit into one out of three pre-determined templates
Selector signal
TRNG

Fig. 17. Generation of control/selector signal for randomized functional transformation of the circuit between a set of pre-determined templates.

means that in the MESO CeNN circuit of Fig. 15 (b), certain MESO gates will be polymorphic, enabling a circuit-level polymorphic reconfiguration between the original circuit and, say, three templates f_1, f_2, and f_3. Hence, the image-processing IP will work at a *sub-optimal accuracy* which is inversely proportional to the HD between the different morphing circuit templates and the original function.[8] By tuning this HD through system-level design, i.e., selecting gates that need to be polymorphic, one can control how similar the IP recovered by *AppSAT* will be when compared to the original IP. Reconfiguration between these circuit templates is controlled by using a TRNG to drive a selector circuit, which selects one set of key-bits, as shown in a simple example in Fig. 17. Note that, in this scenario, the TRNG does not control the distinct key-bits of the MESO gate individually; rather, the control signal is derived from the TRNG such that it randomly cycles between pre-determined sets, which will transform the gate/circuit into one out of several pre-determined templates. When *AppSAT* is mounted on such a dynamically morphing circuit, the constant functional reconfiguration results in the attack recovering parts of the key from different circuit templates at different instances of time. Therefore, the overall stitched key recovered by *AppSAT* from all the circuit versions may have an HD significantly different from the original IP. Please note that, the HD between the polymorphic templates and the original IP, which dictates the accuracy at the system-level, is *different* from the HD between the *AppSAT*-recovered IP and the original IP. The application of *AppSAT* on the approximate IP, along with the key recovery and HD calculation, is represented as a flowchart in Fig. 16.

Since CAD tools and synthesizable *Verilog* libraries for emerging spin devices like MESO are under development, we present proof-of-concept simulations on large-scale ITC-99 benchmarks. In experiments on b14_C benchmark, $\sim$ 11%, $\sim$ 9% and $\sim$ 14% HDs between the original IP and the polymorphic templates f_1, f_2, and f_3, respectively, translate to $\sim$ 28% HD between the *AppSAT*-recovered IP and the original IP. In Table 7, templates 1-3 are approximate versions of each benchmark, with their respective HD from the original design. We execute *AppSAT* (setup details same as in [27]) considering that the benchmark morphs between its original form and three approximate templates. *AppSAT* provides an *approximate-key* after time-out, unlike the SAT-based attack [9]. With this approximate key, HD is computed between the *AppSAT*-recovered IP and the original IP, whereas results are quoted in the last column of Table 7.

TABLE 7
Comparison of HD (in %) between various polymorphic templates and original function, and HD inferred between *AppSAT* recovered IP and original IP for selected ITC-99 benchmarks. HD is calculated using *Synopsys VCS* for 100,000 patterns

Benchmark	HD (in %) from the original design			HD inferred
	template-1	template-2	template-3	after *AppSAT*
b14_C	11.22	9.26	13.78	28.81
b15_C	12.35	9.62	12.88	32.15
b17_C	11.14	10.62	15.24	36.22
b20_C	12.51	14.37	17.86	34.34

7.3 Results and Discussion

The image reconstructed by the CeNN IP as recovered by *AppSAT* (at various representative values of HD between the *AppSAT*-recovered IP and original IP) is shown in Fig. 18. Although the average HD numbers for the proof-of-concept simulations and attacks above are between 28–36% (see last column of Table 7), here we assume an even more powerful attack. That is, we gauge the resilience offered by dynamic morphing when trying to reconstruct images for representative HD values of 10–25%. As can be seen in Fig. 18 (e), at a sufficiently large HD of 25%, the *AppSAT*-recovered CeNN IP fails to faithfully reconstruct the original image. For the *AppSAT*-recovered IPs incurring even larger HDs in Table 7, the reconstructed image will naturally be even more noisy.

Further, text reconstructed using the approximate CeNN IP, recovered by *AppSAT* (for an HD of 20% between *AppSAT*-recovered IP and the original IP), is incorrectly inferred by optical character recognition (OCR) engines like *Tesseract* [45] (Fig. 19).[9] To substantiate the inability of an

8. Here we use HD as a representative metric for image quality. However, HD can be translated to other image processing relevant metrics and the conclusions of this study do not depend on the choice of this metric.

9. *Tesseract* is the industry standard OCR used by *Google* on its mobile devices and for text detection in Gmail. It has been trained using Google's character dataset containing millions of images, and can identify more than 100 languages.

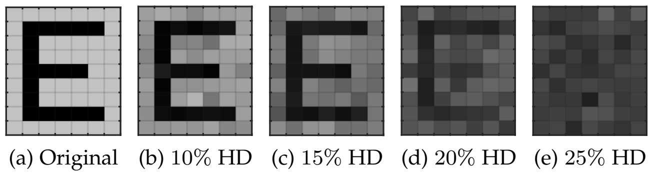


Fig. 18. (a) Original image to the MESO-based CeNN image reconstruction. (b-e) Images reconstructed with approximate IP of the CeNN recovered from *AppSAT*, for HD of 10%, 15%, 20%, and 25%, respectively between *AppSAT*-recovered IP and the original IP. It is essential to note that this HD is different from the accuracy of the approximate circuit reported in Table 7.

attacker to gain a satisfactory approximate IP of the image reconstruction hardware, we use the Long Short Term Memory (LSTM) recurrent neural network module of the *Tesseract 4.0* OCR engine on all alphabets at various HDs between *AppSAT*-recovered IP and original IP. As shown in Fig. 20, the neural network-based OCR is unable to faithfully detect the reconstructed text at higher HDs close to 25%.

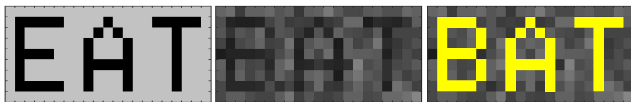


Fig. 19. Incorrectly inferred text, reconstructed using an approximate IP of the CeNN recovered by *AppSAT*, for HD of 20% between *AppSAT*-recovered IP and the original IP.

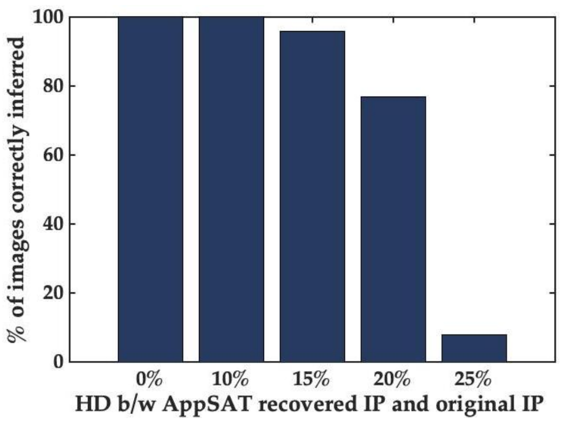


Fig. 20. Proportion of alphabets correctly identified by neural net-based *Tesseract 4.0* OCR engine, where the images were reconstructed using a MESO-based CeNN, for various HD between the *AppSAT*-recovered IP and the original IP.

Thus, we point out that, there is a clear trade-off between the accuracy of the original IP and the resilience to *AppSAT* attacks, in terms of how closely *AppSAT* is able to resolve the original IP. However, for approximate applications like image processing, which can tolerate a certain degree of error, our scheme of dynamic morphing can thwart attempts to recover even an approximate version of the IP. Advanced IP protection mechanisms based on point-functions [13], [18] and stripping of functionality [14] may not be suitable for protecting error-tolerant applications such as the image-processing system considered in this section. This is because the above mentioned techniques trade-off output corruptibility for SAT-attack resilience; the lower the output corruptibility, the stronger the resilience. Hence, attacks working on the notion of recovering an approximate version of the protected IP (e.g., *AppSAT*) are able to successfully recover a *satisfactorily functionally similar* IP if protected using [14], [18].

We finally like to note that dynamic morphing cannot protect systems which demand highly accurate and error-free computations, e.g., cryptographic applications. How reconfiguration at run-time might help in providing additional layer of security for these systems remains an open problem. In general, dynamic camouflaging is suitable for applications that can tolerate a certain degree of error, including machine learning, image processing, neuromorphic circuits etc., which also require protection against reverse engineering due to the sensitive nature of their IP.

8 Synthesis-Level Cost Analysis

In this section, we benchmark the synthesis-level cost for the MESO-based camouflaging primitive along with other spin-based devices. We use the ITC-99 suite for benchmarking, rather than the CeNN image-processing IP demonstrated in Section 7, to showcase the general prospects of full-chip camouflaging using spin-based devices. We note that full-chip dynamic camouflaging may be uncalled for practical applications as in the case study above; again, this analysis here is for benchmarking of different devices.

Setup: We compare the all-spin logic (ASL) primitive [23], the giant spin-Hall effect (GSHE) primitive [25], and the MESO primitive of this work in Table 8. The baseline designs are implemented in CMOS and have been synthesized using *Synopsys Design Compiler* with 2-input gates, in addition to inverters and buffers.

For each synthesized netlist, we replace all cells by their corresponding emerging-device model. Given that libraries and physical-design files for spin-based devices are not available yet, and also given that leading CAD vendors like *Synopsys* and *Cadence* do not support system-level simulations of such spin-based devices yet, this setup is a practical approach. For the MESO primitive, we also include the peripheral MUXes shown in Fig. 3, and we characterize them using *Cadence Virtuoso* for the 15-nm CMOS node using the *NCSU FreePDK15* FinFET library, for a supply voltage of 0.8V.

For benchmarking, for example, the ITC-99 benchmark b17_C comprises 24,228 2-input and inverter/buffer instances. Using the GSHE primitive, along with its peripherals, each of these instances would consume a power of 0.2673 μW [25]. For the MESO primitive, again with peripherals, each gate would consume 0.0615 μW. With simple arithmetic calculations we conclude that the GSHE-based logic would consume 6.5 mW while the MESO-based logic would consume 1.5 mW for b17_C. For area calculations, the same approach is taken. For timing calculations, we keep track of the gates in the critical path, and the delay numbers are summed up. For example, we observe 50 gates in the critical path for b17_C. For the GSHE-based logic, each of these gates would incur a delay of 1.83 ns [25], resulting in a total delay of 91.5 ns. For MESO-based logic, with peripherals, a delay of 0.2579 ns incurs for each instance, which totals to 12.895 ns.

TABLE 8
Comparison of Selected Emerging Device Primitives

Publication	Energy	Power	Delay
ASL [23, a]	0.58 pJ	351.52 μW	1.65 ns
ASL [23, b]	1.16 pJ	351.52 μW	3.3 ns
ASL [23, c]	0.13 pJ	342.11 μW	0.38 ns
GSHE (intrinsic) [25]	0.33 fJ	0.2125 μW	1.55 ns
Obfuscated GSHE [25] (with MUXes)	0.49 fJ	0.2673 μW	1.83 ns
MESO (intrinsic)	**9.3 aJ**	**0.0404 μW**	**0.23 ns**
Obfuscated MESO (with MUXes and interconnects)	**16.04 aJ**	**0.0622 μW**	**0.2579 ns**

The delay for Obfuscated MESO is the sum of the switching time for the intrinsic device (230 ps), the switching time for the interconnect (2.9 ps), and the delay induced by the peripheral MUXes. The corresponding switching energies are 9.3 aJ for the device and 0.18 aJ for the interconnect; these values are extracted from Table 4 of the supplementary material of [4]. The peripheral MUXes (shown in Fig. 3) have been simulated using *Cadence Virtuoso* for the 15-nm CMOS node using the NCSU FreePDK15 FinFET library, for a supply voltage of 0.8V. The area for an intrinsic MESO device, without peripherals, is 0.014 μm^2 [4].

Results: We provide the comparison between selected emerging device primitives in Table 8. The results for full-chip camouflaging are presented in Table 9. We note that ASL-based [23] and GSHE-based [25] full-chip camouflaging incurs excessive power and timing overheads. On the other hand, MESO-based camouflaging offers substantial reductions relative to these spin devices and can be expected to perform even better than CMOS-based camouflaging schemes.[10] This is because the polymorphic MESO device consumes significantly lower switching energy, in the order of $\sim$10 atto Joules, due to its energy-efficient electric-field-driven reversal.

TABLE 9
Comparison between Area, Power, and Performance for ASL-based [23], GSHE-based [25], and MESO-based full-chip camouflaging on selected ITC-99 benchmarks. Absolute values are provided. Area is in μm^2, Power in mW, and Delay in ns. N/A indicates not available.

Benchmark	ASL-based [23]			GSHE-based [25]			MESO-based		
	Area	Power	Perf.	Area	Power	Perf.	Area	Power	Perf.
b15_C	N/A	2,702	54	223.6	2.1	71.4	183.1	0.5	10.1
b17_C	N/A	8,494	71	702.6	6.5	91.5	575.4	1.5	12.9
b18_C	N/A	21,783	137	1,800.2	16.6	115.3	1,474.3	3.8	16.3
b19_C	N/A	42,027	165	3,473.7	32.1	177.5	2,844.9	7.4	25

We also compare synthesis-level PPA cost with a prior CMOS- and LUT-based scheme [28] in Table 10. As mentioned in Section 2.2, functional polymorphism can also be implemented using CMOS-based reconfigurable units, such as FPGA LUTs. We source the implemented scheme of [28] from the set of benchmarks provided in [9]. On average, the scheme [28] incurs area and power overheads of 193% and 206%, respectively, over original designs. The MESO-based reconfiguration scheme does not incur such cost for area and power, but rather significant savings; only for delay/performance, the MESO-based scheme incurs a higher cost than the CMOS-based scheme. Therefore, the use of MESO devices can offer significant advantages for dynamic camouflaging, especially for circuits which are not reliant on high performance.

10. In general, for smaller camouflaging scales and hybrid designs (i.e., emerging spin devices along with CMOS), area and power gains would scale down accordingly, whereas performance will remain similar, given that the emerging devices dominate the switching times.

TABLE 10
Comparison between Area, Power, and Performance for LUT-based obfuscation [28] and MESO-based primitive for dynamic reconfiguration on selected ISCAS-85 benchmarks. Absolute values are provided. Area is in μm^2, Power in mW, and Delay in ns.

Benchmark	Original (CMOS)			LUT-based (CMOS) [28]			MESO-based		
	Area	Power	Perf.	Area	Power	Perf.	Area	Power	Perf.
c432	164.65	0.03	2.79	543.71	0.11	2.96	4.89	0.01	5.67
c880	239.93	0.03	3.31	780.98	0.12	3.48	6.34	0.02	7.48
c1908	250.57	0.05	3.72	674.31	0.14	3.89	5.79	0.02	4.9
c2670	396.87	0.06	3.16	1,193.0	0.18	3.24	10.31	0.03	7.22
c3540	780.18	0.14	3.85	2,308.08	0.42	3.91	22.52	0.06	7.74
c5315	1,029.95	0.17	3.63	2,764.01	0.43	3.73	28.76	0.07	6.45
c7552	1,138.48	0.23	3.93	2,936.11	0.55	3.88	28.81	0.08	7.99
Average Cost	–	–	–	**193%**	**206%**	**3%**	**-97%**	**-56%**	**96%**

9 Conclusion and Future Work

Functional polymorphism has been largely unexplored in the context of securing hardware. We present *dynamic camouflaging* as a novel design-for-trust technique, based on the foundations of run-time polymorphism and post-fabrication reconfigurability exhibited by emerging spin-based devices. Dynamic camouflaging serves well to secure the supply chain end-to-end, including the foundry, the test facility, and the end-user. We show that securing error-tolerant IPs, such as image processors, is suitable from the standpoint of dynamic camouflaging. Finally, MESO-based full-chip camouflaging can offer savings in PPA when compared to both ASL-based and GSHE-based camouflaging approaches.

As a part of future work, we aim to explore viable techniques for securing non-error-tolerant systems like cryptographic applications and/or mission critical systems via dynamic camouflaging. Besides, we will explore the design and implementation of system-level control circuitry for dynamic camouflaging.

Acknowledgments

The work of Satwik Patnaik was supported by the Global Ph.D. Fellowship at NYU/NYU AD. Besides, this work was carried out in part on the HPC facility at NYU AD. This work was supported in part by the Semiconductor Research Corporation (SRC) and the National Science Foundation (NSF) through ECCS 1740136.

References

[1] M. De Marchi, D. Sacchetto, S. Frache, J. Zhang, P.-E. Gaillardon, Y. Leblebici *et al.*, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets," in *2012 International Electron Devices Meeting*. IEEE, 2012, pp. 8–4.

[2] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[3] S. Salahuddin and S. Datta, "Use of negative capacitance to provide voltage amplification for low power nanoscale devices," *Nano letters*, vol. 8, no. 2, pp. 405–410, 2008.

[4] S. Manipatruni, D. E. Nikonov, C.-C. Lin, T. A. Gosavi, H. Liu, B. Prasad *et al.*, "Scalable energy-efficient magnetoelectric spin–orbit logic," *Nature*, vol. 565, no. 7737, p. 35, 2019.

[5] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of logic encrypted chips: Pre-test or post-test?" in *Proc. DATE*, 2016, pp. 139–144.

[6] S. Patnaik, M. Ashraf, O. Sinanoglu, and J. Knechtel, "Obfuscating the Interconnects: Low-cost and Resilient Full-chip Layout Camouflaging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[7] ——, "Best of both worlds: Integration of split manufacturing and camouflaging into a security-driven CAD flow for 3D ICs," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[8] ——, "A Modern Approach to IP Protection and Trojan Prevention: Split Manufacturing for 3D ICs and Obfuscation of Vertical Interconnects," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[9] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. HOST*, 2015, pp. 137–143.

[10] M. E. Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. NDSS*, 2015, pp. 1–14.

[11] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging," *Trans. Inf. Forens. Sec.*, vol. 12, no. 11, pp. 2668–2682, 2017.

[12] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu *et al.*, "Fault analysis-based logic encryption," *Trans. Comp.*, vol. 64, no. 2, pp. 410–424, 2015.

[13] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *International conference on cryptographic hardware and embedded systems*. Springer, 2016, pp. 127–146.

[14] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. CCS*, 2017, pp. 1601–1618.

[15] S. Anceau, P. Bleuet, J. Clédière, L. Maingault, J.-l. Rainard, and R. Tucoulou, "Nanofocused X-ray beam to reprogram secure circuits," in *Proc. CHES*, 2017, pp. 175–188.

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

[17] B. Erbagci, C. Erbagci, N. E. C. Akkaya, and K. Mai, "A secure camouflaged threshold voltage defined logic family," in *Proc. HOST*, 2016, pp. 229–235.

[18] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. ICCAD*, 2016, pp. 28:1–28:8.

[19] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013, pp. 1259–1264.

[20] S. Patnaik, J. Knechtel, M. Ashraf, and O. Sinanoglu, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," in *Proc. ASPDAC*, 2018, pp. 251–258.

[21] S. Patnaik, M. Ashraf, J. Knechtel, and O. Sinanoglu, "Raise your game for split manufacturing: Restoring the true functionality through BEOL," in *Proc. DAC*, 2018, pp. 140:1–140:6.

[22] Y. Bi, K. Shamsi, J.-S. Yuan, P.-E. Gaillardon, G. D. Micheli, X. Yin *et al.*, "Emerging technology-based design of primitives for hardware security," *J. Emerg. Tech. Comp. Sys.*, vol. 13, no. 1, pp. 3:1–3:19, 2016.

[23] Q. Alasad, J. Yuan, and D. Fan, "Leveraging all-spin logic to improve hardware security," in *Proc. GLSVLSI*, 2017, pp. 491–494.

[24] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *Proc. DATE*, 2018, pp. 97–102.

[25] ——, "Spin-Orbit Torque Devices for Hardware Security: From Deterministic to Probabilistic Regime," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[26] S. Ghosh, "Spintronics and security: Prospects, vulnerabilities, attack models, and preventions," *Proc. IEEE*, vol. 104, no. 10, pp. 1864–1893, 2016.

[27] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. HOST*, 2017, pp. 95–100.

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

[29] B. Liu and B. Wang, "Embedded reconfigurable logic for asic design obfuscation against supply chain attacks," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[30] S. Koteshwara, C. H. Kim, and K. K. Parhi, "Key-based dynamic functional obfuscation of integrated circuits using sequentially-triggered mode-based design," *Trans. Inf. Forens. Sec.*, vol. 13, no. 1, pp. 79–93, 2018.

[31] N. E. C. Akkaya, B. Erbagci, and K. Mai, "A secure camouflaged logic family using post-manufacturing programming with a 3.6 GHz adder prototype in 65nm CMOS at 1V nominal VDD," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 128–130.

[32] T. Lottermoser, T. Lonkai, U. Amann, D. Hohlwein, J. Ihringer, and M. Fiebig, "Magnetic phase control by an electric field," *Nature*, vol. 430, no. 6999, p. 541, 2004.

[33] M. Dyakonov and V. Perel, "Current-induced spin orientation of electrons in semiconductors," *Physics Letters A*, vol. 35, no. 6, pp. 459–460, 1971.

[34] K. Shen, G. Vignale, and R. Raimondi, "Microscopic theory of the inverse edelstein effect," *Physical review letters*, vol. 112, no. 9, p. 096601, 2014.

[35] K. Yogendra, M.-C. Chen, X. Fong, and K. Roy, "Domain wall motion-based low power hybrid spin-cmos 5-bit flash analog data converter," in *Sixteenth International Symposium on Quality Electronic Design*. IEEE, 2015, pp. 604–609.

[36] M. E. Massad, J. Zhang, S. Garg, and M. V. Tripunitara, "Logic locking for secure outsourced chip fabrication: A new attack and provably secure defense mechanism," *CoRR*, vol. abs/1703.10187, 2017. [Online]. Available: http://arxiv.org/abs/1703.10187

[37] L. Li and A. Orailoglu, "Piercing logic locking keys through redundancy identification," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 540–545.

[38] S. Keshavarz, F. Schellenberg, B. Richte, C. Paar, and D. Holcomb, "SAT-based reverse engineering of gate-level schematics using fault injection and probing," in *Proc. HOST*, 2018, pp. 215–220.

[39] X. T. Ngo, J. L. Danger, S. Guilley, T. Graba, Y. Mathieu, Z. Najm *et al.*, "Cryptographically secure shield for security IPs protection," *Trans. Comp.*, vol. 66, no. 2, pp. 354–360, 2017.

[40] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson *et al.*, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2012, pp. 151–166.

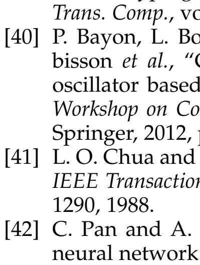
[41] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1273–1290, 1988.

[42] C. Pan and A. Naeemi, "A proposal for energy-efficient cellular neural network based on spintronic devices," *IEEE Transactions on Nanotechnology*, vol. 15, no. 5, pp. 820–827, 2016.

[43] Z. He and D. Fan, "Energy efficient reconfigurable threshold logic circuit with spintronic devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 2, pp. 223–237, 2017.

[44] N. Kani, "Modeling of magnetization dynamics and applications to spin-based logic and memory devices," Ph.D. dissertation, Georgia Institute of Technology, 2017.

[45] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2. IEEE, 2007, pp. 629–633.



Nikhil Rangarajan (Member, IEEE) is a Postdoctoral Associate at the Division of Engineering, New York University Abu Dhabi, United Arab Emirates. He has Ph.D. and M.S. degrees in Electrical Engineering from New York University, NY, USA. His research interests include spintronics, nanoelectronics, device physics and hardware security. He is a member of IEEE.

Satwik Patnaik (Graduate Student Member, IEEE) received the B.E. degree in electronics and telecommunications from the University of Pune, India, and the M.Tech. degree in computer science and engineering with a specialization in VLSI design from the Indian Institute of Information Technology and Management, Gwalior, India. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA.

He is also a Global Ph.D. Fellow with New York University Abu Dhabi, United Arab Emirates. His current research interests include hardware security, trust and reliability issues for CMOS and emerging devices with particular focus on low-power VLSI Design. He is a student member of ACM.

Mr. Patnaik received the Bronze Medal in the Graduate Category at the ACM/SIGDA Student Research Competition held at ICCAD 2018, and the Best Paper Award at the Applied Research Competition held in conjunction with Cyber Security Awareness Week, in 2017.

Johann Knechtel (Member, IEEE) received the M.Sc. degree in information systems engineering (Dipl.-Ing.) and the Ph.D. degree in computer engineering (Dr.-Ing., summa cum laude) from TU Dresden, Germany, in 2010 and 2014, respectively.

He is a Research Scientist with New York University Abu Dhabi, United Arab Emirates. He was a Postdoctoral Researcher with the Masdar Institute of Science and Technology, Abu Dhabi, from 20152016. From 2010 to 2014, he was a Ph.D. Scholar with the DFG Graduate School on "Nano- and Biotechnologies for Packaging of Electronic Systems" hosted at the TU Dresden. In 2012, he was a Research Assistant with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. In 2010, he was a visiting research student with the Department of Electrical Engineering and Computer Science, University of Michigan at Ann Arbor, MI, USA. His research interests cover VLSI physical design automation, with particular focus on emerging technologies and hardware security.

Ramesh Karri (Fellow, IEEE) received the B.E. degree in ECE from Andhra University and the Ph.D. degree in computer science and engineering from UC San Diego. He is currently a Professor of ECE with New York University. He also codirects the NYU Center for Cyber Security. He also leads the Cyber Security thrust, NY State Center for Advanced Telecommunications Technologies, NYU. He co-founded the Trust-Hub and organizes the Embedded Systems Challenge, the annual red team blue team event. He has published more than 240 articles in leading journals and conference proceedings. His research and education activities in hardware cybersecurity include trustworthy ICs, processors and cyber-physical systems, security-aware computer-aided design, test, verification, validation, and reliability, nano meets security, hardware security competitions, benchmarks and metrics, biochip security, and additive manufacturing security. His work on hardware cybersecurity received best paper nominations (ICCD 2015 and DFTS 2015) and awards (ACM TODAES 2018, ITC 2014, CCS 2013, DFTS 2013, and VLSI Design 2012). He received the Humboldt Fellowship and the National Science Foundation CAREER Award. He serves on the Editorial Boards of several IEEE and ACM Transactions (TIFS, TCAD, TODAES, ESL, D&T, and JETC). He has served as the IEEE Computer Society Distinguished Visitor from 2013 to 2015. He has served on the Executive Committee of the IEEE/ACM Design Automation Conference leading the Security@DAC initiative from 2014 to 2017. He delivers invited keynotes, talks, and tutorials on Hardware Security and Trust (ESRF, DAC, DATE, VTS, ITC, ICCD, NATW, LATW, and CROSSING). He co-founded the IEEE/ACM NANOARCH Symposium and has served as program/general chair of conferences (the IEEE ICCD, the IEEE HOST, the IEEE DFTS, NANOARCH, RFIDSEC, and WISEC). He serves on several program committees (DAC, ICCAD, HOST, ITC, VTS, ETS, ICCD, DTIS, and WIFS).

Ozgur Sinanoglu (Senior Member, IEEE) received the first B.S. degree in electrical and electronics engineering and the second B.S. degree in computer engineering from Boazii University, Istanbul, Turkey, in 1999, and the M.S. and Ph.D. degrees in computer science and engineering from the University of California at San Diego, CA, USA, in 2001 and 2004, respectively.

He is a Professor of electrical and computer engineering with New York University Abu Dhabi (NYU Abu Dhabi), United Arab Emirates. He has industry experience with TI, Dallas, TX, USA, IBM, Armonk, NY, USA, and Qualcomm, San Diego, CA, USA. He has been with NYU Abu Dhabi since 2010, where he is the Director of the Design-for-Excellence Lab. His recent research in hardware security and trust is being funded by U.S. National Science Foundation, U.S. Department of Defense, Semiconductor Research Corporation, Intel Corp, and Mubadala Technology. His research interests include design-for-test, design-for-security, and design-for-trust for VLSI circuits, where he has more than 180 conference and journal papers, and 20 issued and pending U.S. Patents. He has given more than a dozen tutorials on hardware security and trust in leading CAD and test conferences, such as DAC, DATE, ITC, VTS, ETS, ICCD, and ISQED.

Prof. Sinanoglu won the IBM Ph.D. Fellowship Award Twice during his Ph.D. He is also the recipient of the Best Paper Awards at IEEE VLSI Test Symposium 2011 and the ACM Conference on Computer and Communication Security 2013. He was a (Guest) Associate Editor for the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the *ACM Journal on Emerging Technologies in Computing Systems*, the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, *Microelectronics Journal* (Elsevier), the *Journal of Electronic Testing: Theory and Applications*, and *IET Computers and Digital Techniques* journals. He is serving as the Track/Topic Chair or Technical Program Committee Member in about 15 conferences.

Shaloo Rakheja (Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA. She was an Assistant Professor of electrical and computer engineering with New York University, Brooklyn, NY, USA. She was a Postdoctoral Research Associate with Microsystems Technology Laboratories, Massachusetts Institute of Technology, Cambridge, USA. She is currently an Assistant Professor of electrical and computer engineering with the Holonyak Micro and Nanotechnology Laboratory, University of Illinois at Urbana–Champaign, Urbana, IL, USA, where she works on nanoelectronic devices and circuits.