Spin-Orbit Torque Devices for Hardware Security: From Deterministic to Probabilistic Regime

Satwik Patnaik*, Nikhil Rangarajan*, Johann Knechtel*, Ozgur Sinanoglu, and Shaloo Rakheja

***Abstract*—Protecting intellectual property (IP) has become a serious challenge for chip designers. Most countermeasures are tailored for CMOS integration and tend to incur excessive overheads, resulting from additional circuitry or device-level modifications. On the other hand, power density is a critical concern for sub-50 nm nodes, necessitating alternate design concepts. Although initially tailored for error-tolerant applications, imprecise computing has gained traction as a general-purpose design technique. Emerging devices are currently being explored to implement ultra-low-power circuits for inexact computing applications. In this paper, we quantify the security threats of imprecise computing using emerging devices. More specifically, we leverage the innate polymorphism and tunable stochastic behavior of spin-orbit torque (SOT) devices, particularly, the giant spin-Hall effect (GSHE) switch. We enable IP protection (by means of logic locking and camouflaging) simultaneously for deterministic and probabilistic computing, directly at the GSHE device level. We conduct a comprehensive security analysis using state-of-the-art Boolean satisfiability (SAT) attacks; this study demonstrates the superior resilience of our GSHE primitive when tailored for deterministic computing. We also demonstrate how probabilistic computing can thwart most, if not all, existing SAT attacks. Based on this finding, we propose an attack scheme called probabilistic SAT (*PSAT*) which can bypass the defense offered by logic locking and camouflaging for imprecise computing schemes. Further, we illustrate how careful application of our GSHE primitive can remain secure even on the application of the PSAT attack. Finally, we also discuss side-channel attacks and invasive monitoring, which are arguably even more concerning threats than SAT attacks.**

***Index Terms*—Hardware security, Imprecise computing, Probabilistic computing, Reverse engineering, IC camouflaging, Logic locking, Spin-orbit torque, Giant Spin-Hall effect, Boolean satisfiability.**

I. Introduction

THE notion of imprecise computing already took root in 1956, with the seminal work by von Neumann, where the concept of error was introduced as an essential part of any computing system, subject to thermodynamical theory [5]. Von Neumann further expounded on the control of errors in simple automatons. More recently, in 2007, the ITRS report [6] stated that “relaxing the requirement of 100% correctness [...] may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures [...].” At present, imprecise circuits are primarily tailored for error-tolerant applications including machine learning, voice recognition, and video processing. Still, the proliferation and subsequent pervasiveness of imprecise computing seems inevitable in the near future, and designers have already started to embrace computational errors as a means for achieving stringent requirements on power dissipation [7], [8]. In this context, emerging devices including nanowire transistors, carbon-based electronics, spin-based computational elements, offer further reduction in power consumption as well as higher integration density compared to their CMOS counterparts [9].

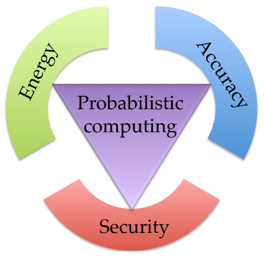


Fig. 1. Interplay between accuracy, energy consumption and security of probabilistic logic.

Meanwhile, hardware security has become a major challenge due to concerns such as theft of design intellectual property (IP), leakage of sensitive data at runtime (via side channels or otherwise), counterfeiting of chips, or insertion of hardware Trojans [10]. Recently it has been advocated that emerging devices can augment CMOS technology to advance hardware security [11]–[13], but very little focus has been given to imprecise computing so far. Hence, it is crucial to discuss hardware security in the context of imprecise computing systems, possibly built with emerging devices. Arguably the most promising aspect of many emerging devices offered toward hardware security is their *functional polymorphism*—a polymorphic gate can implement different Boolean functions, as determined by an internal/external control mechanism [13].

In this work, we consider security as an essential design variable for emerging devices, and we examine the interplay between security, accuracy, and energy (Fig. 1). Although multi-functionality and polymorphism are inherent to spin-orbit torque (SOT) devices in general, we provision the giant spin-Hall effect (GSHE) switch [14] here without loss of generality, since the technology of spin-Hall effect is more mature than other charge to spin conversion devices currently [15], [16]. More specifically, we leverage the GSHE switch demonstrated by Rangarajan *et al.* for energy-efficient computing [17]. We extend the scope of this GSHE switch toward hardware security, i.e., to build polymorphic gates for IP protection. While doing so, we explore both the deter-

*S. Patnaik, N. Rangarajan, and J. Knechtel contributed equally.
This work is an extension of [1].

S. Patnaik, N. Rangarajan, and S. Rakheja are with Department of Electrical Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, 11201, USA. *Corresponding authors*: S. Patnaik (sp4012@nyu.edu), N. Rangarajan (nikhil.rangarajan@nyu.edu), and S. Rakheja (shaloo.rakheja@nyu.edu).
J. Knechtel and O. Sinanoglu are with Division of Engineering, New York University Abu Dhabi, Saadiyat Island, 129188, UAE.

ministic and the probabilistic regime. The presented concepts can be extended to other imprecise computing techniques such as approximate computing, or circuits composed of other emerging devices.

The structure and contributions of this paper can be summarized as follows.

1) We review imprecise computing, hardware security, and prior work in Sec. II. We note that most prior works suffer from low security resilience and/or high layout cost.
2) We design a polymorphic, GSHE-based security primitive in detail in Sec. III. The primitive provides strong security capabilities—given two inputs, all 16 possible Boolean functions can be packed within a single obfuscated instance. Besides, the primitive can readily support deterministic and tunable probabilistic computing.
3) We elaborate on the protection provided by the primitive against various attacks such as imaging-based reverse engineering, side-channel attacks, and analytical SAT attacks (Sec. IV). Regarding SAT attacks, we conduct a comprehensive study (in the deterministic regime) to benchmark our primitive against prior defense schemes, which are mainly based on magnetic devices.
4) The immunity of probabilistic computing against SAT attacks is explored in Sec. V. Most notably, here we present an advanced SAT attack, called PSAT, which allows tackling IP protection for probabilistic circuits. Using conventional SAT and PSAT, we reveal the trade-offs between accuracy, security, and energy for probabilistic computing. Besides, we explore the resilience of polymorphic circuits.
5) In Sec. VI, among other aspects, we outline the prospects for protecting industrial circuits using a hybrid CMOS-GSHE design style. We anticipate that our proposed delay-aware protection can provide strong resilience against SAT attacks with negligible layout overheads.

II. Background and Motivation

A. Imprecise Computing

Three different branches of imprecise computing have emerged, each leveraging unique techniques to harness noise and error for energy-efficient design: (1) stochastic computing, (2) approximate computing, and (3) probabilistic computing.

Stochastic computing is a paradigm that uses deterministic logic blocks for computation, but random binary bit streams. That is, the information is represented by statistical properties of the random bit stream implemented in space and time [18], [19]. This way, for example, multiplication can be achieved using a single AND gate, albeit with relatively low accuracy and long processing times. Digital and analog blocks for stochastic computing were introduced in [20], [21], and have since become popular for not only parallel, error-tolerant applications such as image-processing [22], neuromorphic computing [23], but also for general-purpose low-power designs [7], [24].

Approximate computing is based on inexact logic for the least significant bits (LSBs) of any binary operation. This concept is implemented by altering the design at the circuit level [18]. Using techniques such as logic reduction or pass transistors with lower noise thresholds, approximate computing aims to forgo the accuracy of LSBs to reduce power dissipation and circuit complexity [25]. However, note that approximate computing does not leverage the potential for non-determinism of the logic fabric itself. Low-power approximate full adder, constructed by transistor reduction in mirror-adder cells, and their application for signal processing were discussed in [25], [26]. A design and optimization framework for error and timing analysis of approximate circuits was proposed in [27]. Authors in [28] design an energy-efficient approximate neural network by selective replacement of least significant neurons in the network with imprecise versions.

Probabilistic computing relies on noisy gates that exploit the thermal randomness present in any computing system and, hence, implement an inherently stochastic behavior [18]. Due to the stochastic behavior, key design steps such as verification are naturally more challenging than they are for deterministic computing [29]. Probabilistic computing with CMOS logic is realized by using a noise source (often external), which introduces metastability and error in the binary CMOS switches [30]. Spintronic gates, such as the GSHE gate [14] and the all-spin logic (ASL) gate [31], are intrinsically random, without the need for external noise sources, due to their stochastic magnetization dynamics [32], [33]. A detailed study of probabilistic GSHE logic is presented in [17], which quantifies the energy savings for various operating accuracies and also outlines error models for complex logic gates.

B. Simple Case Study for Approximate Computing

Here, we present a simple case study illustrating the power-accuracy trade-off in the GSHE device [17]. We follow the fundamental rule of thumb for low-power approximate computing, i.e., the most significant bits (MSBs) shall have a higher priority than the LSBs while trading off computational accuracy for power savings. By assigning probabilistic behavior to the logic gates which only affect the LSBs, we restrict the loss in computational accuracy.

Consider a 32-bit adder. For the baseline, assume the adder is synthesized using GSHE gates that initially operate deterministically. Now, the logic gates that impact the computation of the LSBs—the lower 10 bits in this example—can be made to behave probabilistically by operating them at sub-critical input currents [17]. Considering an error rate of 10% for those gates, the worst-case error for the 32-bit addition is only about 0.000024%, whereas the power savings per gate are 50%, and in total about 5%. The GSHE gate in this work can be tuned to incur a power consumption of 0.2125 μW for the deterministic regime, whereas the same gate when operating with an output accuracy of 90%, consumes only 0.1071 μW [17].

Note that this discussion covers only accuracy and power, and the aspect of security will be introduced later.

C. Hardware Security

With the advent of globalization affecting the supply chain of integrated circuits (ICs), hardware security has emerged as a critical concern. The exposure to potential adversaries in any of the third parties tasked for design, manufacturing, and testing has escalated [10]. These adversaries may seek to (i) reverse engineer (RE) the ICs, (ii) counterfeit the ICs, (iii) steal the underlying intellectual property (IP), or (iv) insert some hardware Trojans. Regarding the IP-centric threats, it has been estimated that billions of dollars in revenue are lost every year [34]. To mitigate these threats, several schemes have been proposed; they are mostly based either on camouflaging, logic locking, or split manufacturing.

Camouflaging seeks to mitigate RE attacks; wherein the layout-level appearance of the IC is altered in a manner such that it becomes intractable to decipher its underlying functionality and IP [35]. For CMOS integration, various techniques have been proposed, e.g., look-alike gates [36], threshold-voltage-dependent (TVD) camouflaging [37], [38], and camouflaging of the interconnects [39]. Emerging devices have been recently considered for camouflaging as well, e.g., in [40]–[43]; see also Sec. II-E and Sec. III-B.

Logic locking (also known as logic encryption) obfuscates the IP functionality rather than the device-level layout [44]–[46]. Here, the designer obfuscates the netlist by inserting additional key gates such that the original functionality can only be restored once the correct key bits are applied. The key bits are programmed into a tamper-proof memory after fabrication; this is to hinder attacks during manufacturing time as well as in the field. However, realizing tamper-proof memories is a practical challenge [47], [48]; emerging spin devices can be promising here as well [11].

Analytical attacks targeting camouflaged (or locked) ICs were initially introduced in [2], [49]. Most analytical attacks are based on some notion of Boolean satisfiability (SAT) where a relatively small set of discriminating input patterns (DIPs) may suffice to resolve the functionality of camouflaged gates (or locking keys); see also Sec. II-D. Several SAT-attack resilient techniques were recently proposed, e.g., see [45], [46], [50]. These works seek to impose exponential computation complexity for the SAT solver. Still, most of these techniques are vulnerable to some degree when subject to tailored attacks such as [4], [51], [52]. Besides, we note that prior art on SAT attacks tender exclusively to deterministic computing and circuits. In this paper, among other contributions, we extend the scope of SAT attacks to probabilistic circuits.

Physical attacks range from non-invasive (e.g., power side-channel attacks) and semi-invasive (e.g., localized fault-injection attacks) to invasive attacks (e.g., RE, microprobing the frontside/backside) [53]. While such attacks require more sophisticated tools and know-how than analytical attacks, their potential is widely acknowledged to be more severe. Such attacks are also promising for extracting sensitive data at runtime, even from secured chips, e.g., [54], [55].

D. Boolean Satisfiability and Related Attacks

The problem of Boolean satisfiability (SAT) is an NP-complete problem that determines if a given propositional Boolean formula, usually expressed in its conjunctive normal form (CNF), can be satisfied by any combination of values assumed by the variables of the formula [56]. In case one or more such combinations result in a "true" evaluation of the Boolean formula, then it is termed satisfiable and otherwise, unsatisfiable. A SAT solver is based on an algorithm which heuristically sweeps through the solution space of the Boolean formula to check if any particular combination of variable assignments can satisfy the formula. Such SAT solvers have become quite prevalent for machine learning, artificial intelligence, combinatorial optimization, software verification, and cryptanalysis applications [57], [58]. More recently, SAT solvers have also been tailored for the field of hardware security, namely to resolve/attack logic-locked or camouflaged circuits [2], [49], [51], [59], [60]. Such SAT attacks are successful once the attacker can resolve the key bits of the locking scheme or the functionality of all the camouflaged gates. Note that the various possible, obfuscated functionalities for camouflaging can be modeled as key bits as well. In other words, camouflaging and logic locking are interchangeable in terms of analytical modeling [59], [61].

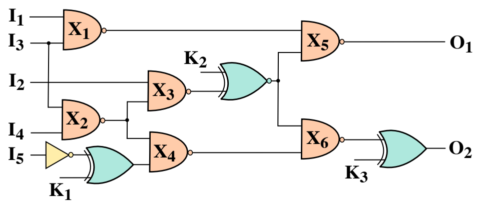


Fig. 2. The *ISCAS-85* benchmark *c17* with three key gates: K_1, K_2, and K_3.

Next, we provide a simple example that illustrates how SAT attacks decipher a locked netlist in general, namely by repeated iterations over the key space. Consider the benchmark circuit *c17* shown in Fig. 2, which has been locked with three key bits: K_1, K_2, and K_3. Now, the attack procedure is to stepwise fix a particular input pattern and then iterate over various possible keys, eliminating those whose variable assignments cannot satisfy the Boolean formula. For example, in Table I, the input pattern "00100" is chosen (either randomly or heuristically) in the first iteration. The corresponding output is "00", as obtained from an oracle (i.e., a working chip queried with the input). Accordingly, some keys cannot result in satisfiable assignments, namely k_0, k_2, k_3, k_5, k_6, and k_7. These keys are pruned, and in the same way, the second iteration prunes key k_4. This leaves k_1 as the last remaining key, which is returned as the attack solution.

It is important to note that the outlined attack flow remains purposefully generic and abstract. Actual SAT attacks all apply various heuristics and techniques to efficiently tackle the solution space, avoiding brute-force behavior. Interested readers are referred to [2], [49], [51], [59], [60].

E. Prior Art and Limitations

Now, we briefly review some prior art and their limitations. A more detailed comparison in terms of power and delay, and (lack of) resilience against SAT attacks are provided in Sec. III-B and Sec. IV-C (Table V), respectively.

In [40], Zhang *et al.* implemented a low-power and versatile gate using a GSHE-based magnetic tunnel junction (MTJ) as the basic switching element. However, this device is not explicitly tailored for security; it is unable to support logic locking by itself, as it is not polymorphic. More concerning is the limitation to only four possible Boolean functions, which renders this primitive weak against SAT attacks.

Alasad *et al.* [41] use ASL to design three different security primitives, supporting three sets of camouflaged functionalities: INV/BUF, XOR/XNOR, and AND/NAND/OR/NOR. The layouts of the three primitives are unique; they can be readily distinguished by imaging-based RE tools, which also eases subsequent SAT attacks. Besides, the primitives suffer from relatively high power consumption of $\sim 350\ \mu$W at ns delays.

Winograd *et al.* [42] introduced a spin-transfer torque (STT)-based reconfigurable lookup table (LUT), explicitly

TABLE I
SAT attack on the benchmark *c17*, locked as in Fig. 2. Labels k_0–k_7 represent all possible combinations of key bits, from 000 to 111, and columns denote the corresponding outputs, which are compared with the oracle output. k_1 is the correct key.

Input Patterns $I_1I_2I_3I_4I_5$	Oracle Output O_1O_2	Output for Different Key Combinations k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	Inference
00000	00	01	00	10	11	00	01	10	11	
00001	01	00	01	10	11	01	00	10	11	
00010	11	10	11	01	00	10	11	00	01	
00011	11	10	11	00	01	10	11	01	00	
00100	00	01	00	10	11	00	01	10	11	Iteration 1: $k_0, k_2, k_3, k_5, k_6, k_7$ are pruned
00101	01	00	01	10	11	01	00	10	11	
00110	11	10	11	01	00	10	11	00	01	
00111	11	10	11	00	01	10	11	01	00	Iteration 2: k_4 is pruned $\Rightarrow$ k_1 is inferred as correct key
...	...	...	...	...	...	...	...	...	...	
11111	10	11	10	10	11	11	10	10	11	

addressing hardware security. However, their approach falls short regarding resilience against SAT attack. Since the authors did not report on any SAT attack themselves, we conducted exploratory experiments ourselves. For example, we protect the *ITC-99* benchmark *s38584* according to their scheme and observe that the resulting layouts can be decamouflaged in less than 30 seconds (i.e., average SAT runtime over 100 runs of random gate selection according to [42]). This weak resilience stems from the limited use of their STT-LUT primitive to curb power, performance, and area (PPA) overheads.

Yang *et al.* [43] recently proposed an SOT-based design for reconfigurable LUTs. Their concept is tailored for obfuscation; it can support all 16 possible Boolean functions for two inputs, like ours. However, the authors neglect powerful SAT attacks. Based on overly optimistic assumptions regarding the attacker's capabilities (presumably to curb PPA cost as well), the authors limit their study to the obfuscation of 16/32/64 gates. In experiments similar to those we conducted for [42], we found that such small-scale obfuscation is easily resolved.

As for CMOS-centric camouflaging, most schemes are static (i.e., not polymorphic) and tend to incur a high layout cost. For example, the static look-alike NAND-NOR-XOR gate proposed by Rajendran *et al.* [36] induces overheads of 4$\times$ in area, 5.5$\times$ in power, and 1.6$\times$ in delay (compared to a regular two-input NAND gate). The TVD full-chip camouflaging as proposed in [38] still induces overheads of 14%, 82%, and 150% in PPA, respectively. As a result, such schemes are limited to a cost-constrained and selective application, which has severe implications for security (Sec. IV).

Finally, we acknowledge that Koteshwara *et al.* recently proposed dynamic obfuscation at the system level [62], albeit their work focuses only on CMOS implementation. We anticipate that inherently polymorphic gates, such as the GSHE device, can advance such a scheme. Furthermore, McDonald *et al.* [63] advocate runtime polymorphism for IP protection, albeit without any security analysis using SAT attacks and without any implementation details toward polymorphic devices.

III. Design of a Giant Spin Hall Effect (GSHE) Security Primitive

Protection schemes based on emerging devices can be competitive, even when compared to regular, unprotected CMOS circuits. Leveraging GSHE is one approach among many to realize SOT-based magnetic devices. The GSHE switch has

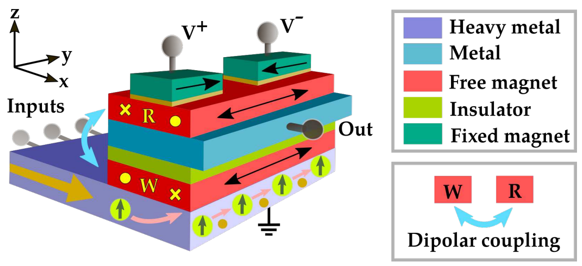


Fig. 3. Structure of the GSHE switch. The concept is derived from [17], but here we adopt a stacked integration to maximize the dipolar coupling.

been studied for a while, and its understanding is relatively mature. The SOT phenomena has been experimentally measured in several magnetic and non-magnetic bilayers at 300K [64], [65]. Likewise, the read-out mechanism in the GSHE device (see also Fig. 3) has been experimentally demonstrated in similar magnetic structures [66], [67]. Experiments are currently underway to integrate the read and write circuitry in the GSHE device to realize Boolean logic [15], [16].

Note that truly polymorphic gates such as the GSHE switch can inherently support both camouflaging and locking due to the following reasons. First, owing to their uniform device-level layout, the actual function of a polymorphic gate is hard to determine from its physical implementation, particularly when optical-imaging-based RE techniques are used. Second, the actual function is dependent on control currents and voltages, which can act as key inputs. Hence, the notions of locking and camouflaging are used interchangeably in the remainder of this work.

A. Structure and Operating Principle of the GSHE Switch

The GSHE switch, which is at the heart of the proposed primitive, is constructed by combining a heavy-metal spin-Hall layer, such as tantalum, tungsten, platinum or palladium, with a magnetic tunnel junction (MTJ) arrangement (Fig. 3). Above the heavy-metal layer are two nanomagnets for write and read modes (W-NM and R-NM, red). The W-NM is separated from the output terminal via an insulating oxide layer (green). On top of the R-NM sit two fixed ferromagnetic layers (dark green) with anti-parallel magnetization directions.

The switch relies on the spin-Hall effect [68] for generating and amplifying the spin current input, and the magnetic dipolar coupling phenomenon [69] to magnetically couple R-NM and W-NM, while keeping them electrically isolated. Thereby, the

R-NM and W-NM are coupled. A charge current through the bottom heavy-metal layer (along $\hat{x}$) induces a spin current in the transverse direction (along $\hat{y}$), which is used to switch the magnetization state of the W-NM. The dipolar coupling field then causes the R-NM to switch its orientation. That is because in the presence of magnetic dipolar coupling, the minimum energy state is the one in which the W-NM and the R-NM are anti-parallel to each other [14]. The final magnetization state of the R-NM is read off using a differential MTJ setup. The logic (1 or 0) is encoded in the direction of the electrical output current (+I or -I). The current direction depends on the relative orientations of the fixed magnets in the MTJ stack with respect to the final magnetization of the R-NM. The parallel path offers a lower resistance for a charge current passing either from the MTJ contact to the output terminal or vice versa (i.e., from the output terminal to the MTJ contact). Hence, depending on the polarity of the read-out voltage applied to the low-resistance path (i.e., either V$^+$ or V$^-$), the output current either flows inward or outward, representing the logic encoding of the GSHE switch operation.

This basic GSHE device can readily implement a BUF or INV gate (buffer or inverter operations). To realize more complex multi-input logic gates, a tie-breaking control signal X with a fixed amplitude and polarity is applied in addition to the primary input signals at the input terminal. That is, the input of the GSHE switch (or, more generally, any SOT-driven magnetic switch) is additive in nature. The polarities of the control signal and the MTJ voltage polarities are used to permute between different Boolean operations (Fig. 4). See also Sec. III-C and its Fig. 8 for all 16 possible Boolean gates.

The GSHE switch is a noisy polymorphic device, whose probability for output correctness depends on the input spin current's amplitude and duration, i.e., the outputs generated by the previous logic stages. In general, the control signal X is used to set the functionality for any current stage (n^{th} stage), and the output correctness probability for each next stage ($(n+1)^{th}$ stage) is as follows [17]:

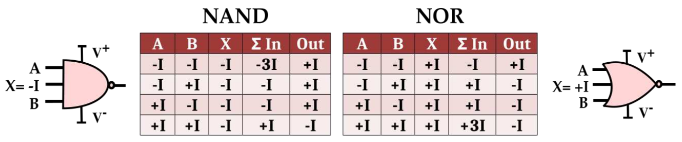
$$\mathcal{P}_{\text{correct}}^{\text{n+1}} = \mathcal{P}_{\text{flip}}\left(I_{\text{sX}} + \Sigma_{\text{i}}^{N_{\text{input}}} \frac{\beta \Delta G_{\text{i}}^{\text{n}} V_{\text{i}}^{\text{n}}}{1 + rG_{\text{i}}^{\text{n}}}\right)[f] + 1.\,[1-f] \quad (1)$$

where $\mathcal{P}_{\text{flip}}$ is the probability for flipping of the W-NM in the $(n+1)^{\text{th}}$ stage's GSHE switch. This probability itself is a function of the current supplied by the n^{th} stage and the magnitude of the control spin current I_{sX}. The relationship between the output current of a particular stage and the voltage supply in the MTJ arrangement of that stage is according to [17], wherein β is the spin-Hall current amplification factor, G^{n} is the MTJ conductance for the n^{th} stage, V^{n} is the MTJ voltage for the n^{th} stage, and r is the resistance of the spin Hall layers. The function f represents the Boolean function to be implemented, and establishes the condition for flipping of the magnetization state.

B. Characterization and Comparison of the GSHE Switch

The conceptual layout of the GSHE switch (Fig. 5) is drawn based on the design rules for beyond-CMOS devices [9], i.e., in units of maximum misalignment length λ. The area of the GSHE switch is accordingly estimated to be $0.0016\mu\text{m}^2$.

The material parameters for the GSHE switch considered are given in Table II. A spin current (I_S) of at least 20μA is required for deterministic computing, as compared to the sub-critical currents sufficient for probabilistic computing [17].



NAND

A	B	X	Σ In	Out
-I	-I	-I	-3I	+I
-I	+I	-I	-I	+I
+I	-I	-I	-I	+I
+I	+I	-I	+I	-I

NOR

A	B	X	Σ In	Out
-I	-I	+I	-I	+I
-I	+I	+I	+I	-I
+I	-I	+I	+I	-I
+I	+I	+I	+3I	-I

Fig. 4. The current-centric truth tables for NAND and NOR functionalities, with inputs A and B (X is a control signal). As always the case for our GSHE-based primitive, logic 1/0 is represented by an output current +I/-I.

TABLE II
MATERIAL PARAMETERS OF THE GSHE SWITCH

Parameter	Value
Volume of nanomagnets (NM)	$(28 \times 15 \times 2)$ nm^3 [17]
Saturation magnetization M_{S} of NM	10^6 A/m (W-NM) [17] 5×10^5 A/m (R-NM) [17]
Uniaxial energy density K_{u} of NM	2.5×10^4 J/m^3 (W-NM) [17] 5×10^3 J/m^3 (R-NM) [17]
Spin current I_{S}, determ. switching	20 μA [17]
Resistance area product RAP	1 $\Omega\mu$m^2 [70]
Tunneling magnetoresistance TMR	170% [70]
Parallel conductance G_{P}	420 μS
Anti-parallel conductance G_{AP}	155.6 μS
Resistivity of heavy metal (HM) ρ	$5.6 \times 10^{-7}\Omega$–m
Spin-Hall angle θ_{SH} of HM	0.4 [71]
Thickness t_{HM} of HM	1 nm
Internal gain β of HM $\beta = \theta_{\text{SH}} \times (w_{\text{NM}}/t_{\text{HM}})$	$0.4 \times$ (15 nm/1 nm) $= 6$
Resistance r of HM	≈ 1 $k\Omega$

The performance of the switch is determined by the nano-magnetic dynamics, which is simulated using the stochastic Landau-Lifshitz-Gilbert-Slonczewski equation [33]. Simulated delay distributions are illustrated in Fig. 6, and these distributions are computed using a CUDA-C model [17]. For the propagation delay for deterministic computing, we subsequently assume a mean delay of 1.55 ns as obtained for $I_S = 20$ μA. Further, this delay was then used to construct a behavioral Verilog model to obtain transient responses (see Fig. 7).

The power dissipation for the read-out phase is derived according to the equivalent circuit shown in Fig. 5 (inset). Using

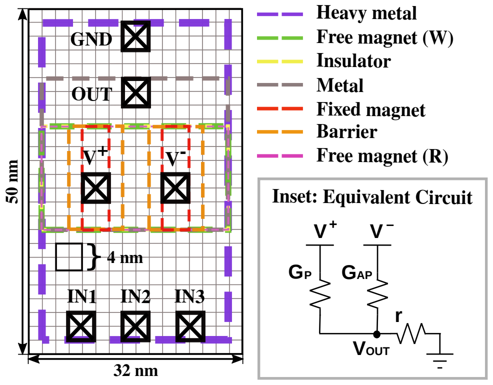


Fig. 5. The conceptual layout of the GSHE switch (main part), and the equivalent circuit (inset, derived from [14]). The power dissipation of the latter is dictated by the resistance r of the heavy metal as well as the conductances of the anti-parallel, high-resistance path (G_{AP}) and the parallel, low-resistance path (G_P) build up by the fixed ferromagnets.

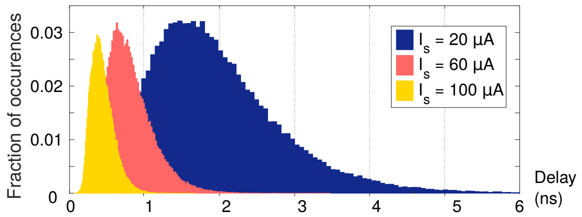


Fig. 6. Delay distributions for the GSHE switch at various spin currents (I_S), obtained from 100,000 simulations each. Note that the spread and mean delay diminish with increasing I_S, however, at the cost of higher power dissipation. Also note that for currents below 20 μA, probabilistic switching occurs.

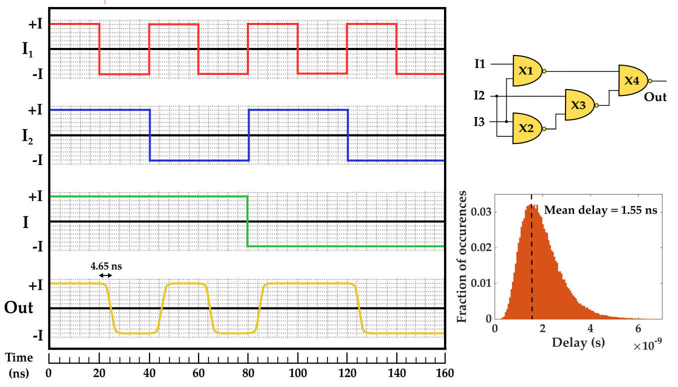


Fig. 7. Transient response for all input patterns applied for an example circuit (right top). The critical path comprises three GSHE gates, which each exhibit a mean delay of 1.55 ns (right bottom), hence the overall delay is 4.65 ns.

the following equations and the parameters listed in Table II, the power dissipation of the GSHE switch for deterministic computing, including leakage, is derived as 0.2125 μW.

$$P = \frac{V_{\mathrm{OUT}}^2}{r} + (V_{\mathrm{SUP}} - V_{\mathrm{OUT}})^2 G_{\mathrm{P}} + (V_{\mathrm{OUT}} + V_{\mathrm{SUP}})^2 G_{\mathrm{AP}} \tag{2a}$$

$$V_{\mathrm{SUP}} = \left|V^{+/-}\right| = \left(\frac{I_{\mathrm{S}}}{\beta}\right)\left(\frac{1 + r(G_{\mathrm{P}} + G_{\mathrm{AP}})}{G_{\mathrm{P}} - G_{\mathrm{AP}}}\right) \tag{2b}$$

$$V_{\mathrm{OUT}} = \frac{I_{\mathrm{S}}\ r}{\beta} \tag{2c}$$

$$\frac{G_{\mathrm{P}}}{G_{\mathrm{AP}}} = 1 + \mathrm{TMR};\ G_{\mathrm{P}} = \frac{A(\text{nanomagnets})}{\mathrm{RAP}} \tag{2d}$$

In Table III, we compare the GSHE switch against those of existing deterministically driven devices, including ones that are not necessarily security-oriented. The switch is superior in terms of energy/power but has a higher delay compared to CMOS devices. Nonetheless, the GSHE-based primitive can serve to protect industrial designs without inducing significant delay overheads, as we will outline in Sec. VI.

As for security in terms of obfuscation, the number of possible functions is the relevant metric—the GSHE switch significantly outperforms most prior art. For the recent work proposed by Yang *et al.* [43], we note the following. First, their study reports relative overheads, but no absolute device-level numbers (hence the “N/A” labels in Table III). Second, to support all 16 possible functions while implementing an LUT, their primitive requires multiple magnetic devices and related peripheral circuitry. In contrast, our switch can implement all 16 functions within one device, and we require peripheral circuitry only for transduction and obfuscation purposes

TABLE III
COMPARISON OF SELECTED EMERGING-DEVICE PRIMITIVES
(DETERMINISTIC REGIME, WITHOUT PERIPHERAL CIRCUITRY)

Publication	**Functions (2 Inputs)**	**Energy**	**Power**	**Delay**
[12] SiNW	NAND/NOR	0.05–0.1 fJ	1.13–1.77 μW	42–56 ps
[41, a] ASL	NAND/NOR/AND/OR	0.58 pJ	351.52 μW	1.65 ns
[41, b] ASL	XOR/XNOR	1.16 pJ	351.52 μW	3.3 ns
[41, c] ASL	INV/BUF	0.13 pJ	342.11 μW	0.38 ns
[72] DWM	AND/OR	67.72 fJ	60.46 μW	1.12 ns
[13] DWM	NAND/NOR/XOR/XNOR/ AND/OR/INV	N/A	N/A	N/A
[40] GSHE	AND/OR/NAND/NOR	N/A	N/A	N/A
[42] STT	NAND/NOR/XOR/ XNOR/AND/OR	N/A	N/A	N/A
[43] SOT	All 16	N/A	N/A	N/A
Ours GSHE	**All 16**	**0.33 fJ**	**0.2125 μW**	**1.55 ns**

(Sec. III-C). Third, the primitive in [43] is not inherently polymorphic; hence, it can only support static camouflaging.

C. GSHE Security Primitive: Protecting the Design IP

The GSHE switch is leveraged for a simple but versatile and effective security primitive—all 16 possible Boolean functions can be cloaked within a single device (Fig. 8). In other words, employing this primitive instead of regular gates can hinder RE attacks of the chip's design IP, without the need for the designer to alter the underlying netlist. For example, to realize NAND/NOR using this primitive, three charge currents are fed into the bottom layer of the GSHE switch at once (Fig. 4 and 8): two currents represent the logic signals A and B, and the third current (X) acts as the “tie-breaking” control input.

For some functions, the logic signals have to be provided as MTJ voltages, not as charge currents. To transduce voltage into charge currents (as well as obtain altering current polarities), magneto-electronic transducers can be used [73], [74]. Such transducers can be placed in the interconnects, and they are capable of charge current conversion (i.e., +I to -I) and voltage to charge current conversion (i.e., high/low voltages to +/-I) and vice versa.

Note that three wires must be used for the GSHE input terminals (Fig. 5). This renders the primitive indistinguishable for imaging-based RE by malicious end-users, irrespective of the actual functionality. As such, some gates will require dummy wires; these wires can be implemented either using RE-resilient interconnects in the BEOL [39] or with the help of logic locking (i.e., MUXes and key bits are used to seemingly switch between real/dummy wires). Similar protection is required for the assignment of the MTJ voltages. Figure 9 illustrates the concept for such a peripheral circuitry.

To hinder fab-based adversaries, we outline two equally promising options for secure implementation: (a) leverage split manufacturing [75], (b) provision for a tamper-proof memory. For option (a), the control inputs and the MTJ terminals shall remain protected from the untrusted FEOL fab. Hence, the related wires have to be routed through the BEOL, which is then manufactured by a separate, trusted fab [75]. Recent studies have demonstrated such guided routing within the BEOL for security considerations [76], [77]. For option (b), a tamper-proof memory holds a secret key that defines the correct assignment of control inputs and voltages (using some additional circuitry). The key is loaded into the memory post-fabrication by the IP holder or authorized parties. Both options

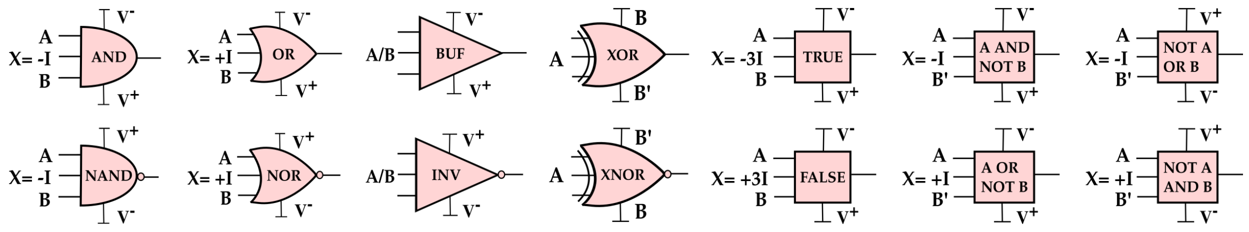


Fig. 8. All 16 possible Boolean functionalities for two inputs, A and B, implemented using the proposed primitive. If required, X serves as control signal, not as regular input. Note that BUF and INV capture two functionalities each.

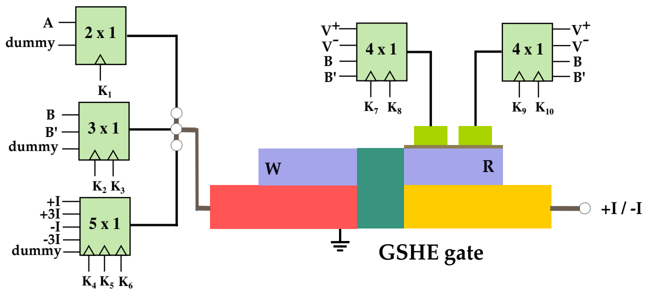


Fig. 9. Peripheral circuitry (the GSHE gate is laid out horizontally for clarity). For security reasons, the MUX control signals have to connect to a tamper-proof memory or leverage RE-resilient wiring.

represent a notable advancement over prior work related to camouflaging, where the IP holder *must* trust the fab because of the circuit-level protection mechanism. In the remainder of this paper, we focus on *malicious end-users.*

IV. Security Analysis

Here, we elaborate on the security promises of the GSHE primitive against various attacks. Most notably, a comprehensive study for analytical SAT attacks is conducted, where we benchmark our primitive against prior art. A key assumption for the SAT attacks is that the GSHE primitive is applied in the context of deterministic computing—we cover the security analysis for probabilistic computing separately in Sec. V.

A. Threat Model

The malicious end-user is interested in resolving the underlying IP implemented by the obfuscated chip. We consider that an attacker possesses the know-how and has access to RE equipment such as required for imaging-based RE. However, the attacker does not have access to advanced capabilities for invasive read-out attacks to, e.g., resolve the voltage and current assignments for individual GSHE primitives at runtime. (Even if so, such attacks seem practically challenging.)

In accordance with prior works, we assume that an attacker procures multiple copies of the chip from open market; she/he uses one for RE (which includes de-packaging, de-layering, imaging of individual layers, stitching of these images and final netlist extraction [78]), and another as an oracle to obtain input-output (I/O) patterns. These patterns are then utilized for SAT-based attacks. The attacker can also use the oracle chip to evaluate side-channel leakage at runtime.

B. On Reverse Engineering and Side-Channel Attacks

1) Layout Identification and Read-Out Attacks: Recall that the physical layout of the proposed primitive is uniform (Sec. III); hence it remains indistinguishable for optical-imaging-based RE. A more sophisticated attacker might, however, leverage electron microscopy (EM) for identification and read-out attacks. For example, Courbon *et al.* [55] used scanning EM, in passive voltage-contrast (PVC) mode, to read out memories in supposedly secured chips. While such attacks are yet to be demonstrated on switching devices at runtime, we believe that the proposed primitive can thwart them for three reasons. First, the dimensions of the GSHE switch are significantly smaller than CMOS devices, which is a challenge regarding the spatial resolution for EM-based analysis [55]. Second, the primitive readily supports probabilistic switching. This implies that once an attacker can read-out the switch at runtime, she/he still has to learn and account for the underlying error distributions. Third, the primitive is truly polymorphic, and its functionality may be switched at runtime; see also next.

2) Polymorphism at the System Level: Given the truly polymorphic nature of the GSHE switch and assuming some additional circuitry to switch their functionalities judiciously, one can implement *runtime polymorphism* at the system level. The gates' functionalities are not static anymore, possibly even for static input patterns, whereupon an attacker is bound to misinterpret some parts of the layout—it seems impossible to resolve all dynamic features on a full-chip scale at once.[1]

Besides hindering read-out threats, polymorphism at the system level is also powerful to thwart SAT attacks. In fact, we provide such a concept based on the GSHE switch in Sec. V-C.

3) Photonic Side-Channel Attacks: It is well known that CMOS devices emit photons during operation, which makes them vulnerable to powerful attacks [79], [80]. For example, Tajik *et al.* [79] successfully conduct an optical read-out attack against the bitstream encryption feature of a *Xilinx Kintex 7* FPGA. Contrary to CMOS, the GSHE switch itself does not emit any photons. The fundamentally different, magnetic switching principle thus renders the primitive inherently resilient against photonic side-channel attacks. We caution that a system-level assessment against such attacks shall be performed nevertheless (once such chips are manufactured) since additional circuitry may or may not remain vulnerable.

4) Magnetic- and Temperature-Driven Attacks: Ghosh *et al.* [11] consider and review attacks on spintronic memory devices using external magnetic fields and malicious temperature curves. As for the GSHE switch, note that it is tailored for robust magnetic coupling (between the W and R nanomagnets) [17], and this coupling would naturally be disturbed by any external magnetic fields. Hence, an attacker leveraging a

[1]For example, Courbon *et al.* [55] report that it took 50 ns to read-out one pixel of one memory cell; this is well above the 1.55 ns switching speed for the GSHE device for deterministic computing (recall Sec. III).

TABLE IV
CHARACTERISTICS OF SYNTHESIZED BENCHMARKS (ITALICS: *EPFL Suite* [83]; BOLD: *IBM Superblue Suite* [84])

Benchmark	Inputs	Outputs	Gates	Benchmark	Inputs	Outputs	Gates
aes_core	789	668	39,014	*log2*	32	32	51,627
b14	277	299	11,028	**sb1**	8,320	13,025	856,403
b21	522	512	22,715	**sb5**	11,661	9,617	741,483
c7552	207	108	4,045	**sb10**	10,454	23,663	1,117,846
ex1010	10	10	5,066	**sb12**	1,936	4,629	1,523,108
pci_bridge32	3,520	3,528	35,992	**sb18**	3,921	7,465	659,511

magnetic probe may induce stuck-at-faults which are, however, hardly controllable due to multiple factors: the very small size of the GSHE switch, accordingly large magnetic fields required for the probe, the state of the nanomagnets, the orientation of the fixed magnets, and also the voltage polarities for the MTJ setup. Temperature-driven attacks will impact the retention time of the GSHE switch. The resulting disturbances, however, are stochastic due to the inherent thermal noise in the nanomagnets; fault attacks are accordingly challenging as well. As a result, we believe that subsequent sensitization attacks to resolve the obfuscated IP (e.g., as proposed in [36]) will be difficult, if practical at all.

C. Study on Large-Scale IP Protection Against SAT Attacks

1) Experimental Setup: We model the GSHE primitive and selected prior art [12], [13], [36], [37], [40]–[42], [81] as outlined in [59]. More specifically, we model the GSHE primitive as follows. The logical inputs a and b are fed in parallel into all 16 possible Boolean gates, and the outputs of those gates are connecting to a 16-to-1 MUX with four select/key bits. As for other prior art with less possible functionalities, a smaller MUX with less key bits may suffice (e.g., for [36], a 3-to-1 MUX with two key bits is used). Although the GSHE primitive inherently supports locking as well, here we contrast it only to camouflaging primitives, without loss of generality. Besides emerging-device primitives, we also contrast to CMOS-centric primitives; this is meaningful since for any IP protection scheme the resilience against SAT attacks hinges on the number and composition of obfuscated functionalities [2], [39], [49], not their physical implementation.

For a fair evaluation, the same set of gates are protected; gates are randomly selected once for each benchmark, memorized, and then the same selection is reapplied across all techniques. We evaluate all techniques against powerful state-of-the-art SAT attacks [2], [4], [82], run on an Intel Xeon server (2.3 GHz, 4 GB per task allowed). The time-out (labelled as "t-o") is set to 48 hours. We conduct our experiments on traditional benchmarks suites, that is *ISCAS-85*, *MCNC*, and *ITC-99*, but also on the large-scale *EPFL suite* [83] (and on the industrial *IBM superblue suite* [84]; see Sec. VI for those experiments). The benchmarks are summarized in Table IV.

2) Results: In Table V, we report the runtimes incurred by the seminal attack of Subramanyan *et al.* [2], [82]. While there are further metrics such as the number of clauses, attack iterations, or number of remaining feasible assignments [59], runtime is a straightforward yet essential indicator—either an attack succeeds within the allocated time or not.

We observe that for the same number of gates protected, the more functions a primitive can cloak, the more resilient it becomes in practice. More importantly, the runtimes required for decamouflaging—if possible at all—tend to scale exponentially with the percentage of gates being camouflaged.[2]

In comparison with prior art, our primitive induces by far the largest efforts across all benchmarks. Except for *ex1010*, none of the benchmarks could be resolved within 48 hours once we protect 20% or more of all gates. To confirm this superior resilience of our primitive, we conducted further experiments running for 240 hours with full-chip protection (100% camouflaging)—the designs could still not be resolved. Moreover, we also observe some computational failures (e.g., "*internal error in 'lglib.c': more than 134,217,724 variables*"); this hints at practical limitations about the scalability of SAT attacks, as one can reasonably expect [49].

We also apply *Double DIP*, provided by Shen *et al.* [4]. The key advancement of their attack is that it rules out at least two incorrect keys in each iteration. Conducting the very same set of experiments as before, we observe that the runtimes are on average even higher across all benchmarks (Table VI). For example, decamouflaging the benchmark *aes_core* when 10% IP protection is applied using our primitive requires ≈7 hours for [2], but ≈15 hours for [4]. This implies that *Double DIP*, while successful for protection schemes such as *SARLock* [45], cannot cope well with our large-scale camouflaging scheme.

3) On Provably Secure Versus Large-Scale Schemes: Contrary to provably secure schemes such as [45], [46], [50], which are often backed by mathematical formulations, one may find it difficult to engage in "plain" but large-scale camouflaging. The reason is that the solution space C, covering all possible functionalities of the design after camouflaging, is hard to quantify precisely [2], [49], [50]. More specifically, C depends on (i) the number and composition of functions cloaked by each primitive, (ii) the number of gates protected, (iii) the selection of gates protected, and (iv) the interconnectivity of the design. Since all aspects are interacting, SAT attacks may or may not be able to prune C efficiently, but this can only be evaluated by running the attacks. As shown above for [4], some heuristics can, in fact, be counterproductive when tackling large-scale camouflaging.

Now, also recall that prior "plain" protection schemes are limited in both (i) and (ii) by cost considerations (Sec. II-E).[3] In contrast, thanks to the innate polymorphism of the proposed GSHE primitive, we are unbound toward large-scale camouflaging with all 16 possible functionalities cloaked within one device. As a result, we believe that our scheme is competitive against provably secure techniques. In this context it is important to note that provably secure schemes have to trade-off corruptibility and resilience against SAT attacks [51]: the larger the desired resilience, the lower the corruptibility, and vice versa. This trade-off also implies that a high-resilience scheme comes at the cost of effectively protecting only a small part of the IP. For our scheme, however, these concerns are inherently mitigated. That is, we can readily protect all of the IP, and the resilience of our schemes relies on incurring an excessive computational cost for the SAT solver

[2]Inducing prohibitive computational cost is also the primary objective for provably secure schemes as in [45], [46], [50]. We further elaborate on provably secure schemes versus our large-scale scheme in Sec. IV-C3.

[3]A massive interconnectivity may also impose a substantial cost, but more importantly, most prior studies ignore the potential of obfuscating the interconnects for IP protection to begin with. Patnaik *et al.* [39] proposed a dedicated design flow for low-cost and large-scale obfuscation of interconnects.

TABLE V
RUNTIME FOR SAT ATTACKS [2], [82], ON SELECTED DESIGNS, IN SECONDS (TIME-OUT "T-O" IS 48 HOURS, I.E., 172,800 SECONDS)

Benchmark	10% IP Protection							20% IP Protection						
	[36] (3)*	[37], [42] (6)*	[12] (4)*†	[41, c], [81] (2)*	[40], [41, a] (4)*	[13] (7+1)*‡	Our (16)*	[36] (3)*	[37], [42] (6)*	[12] (4)*†	[41, c], [81] (2)*	[40], [41, a] (4)*	[13] (7+1)*‡	Our (16)*
aes_core	610	4,710	890	132	536	6,229	25,890	4,319	41,844	11,306	407	9,432	t-o	t-o
b14	2,078	20,603	11,465	6,884	17,634	27,438	60,306	56,155	t-o	64,145	8,426	t-o	t-o	t-o
b21	7,813	162,324	45,465	3,977	24,035	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
c7552	37	210	74	12	66	371	2,289	169	14,575	1,153	110	1,327	172,548	t-o
ex1010	62	215	82	12	73	295	922	171	1,047	274	38	250	1,310	4,701
log2	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
pci_bridge32	1,119	t-o	9,011	1,325	2,690	t-o	t-o	54,577	t-o	t-o	t-o	t-o	t-o	t-o
	30% IP Protection							40–100% IP Protection§						
aes_core	17,148	t-o	31,601	2,020	26,498	t-o	t-o	t-o	t-o	t-o	8,206	t-o	t-o	t-o
b14	56,787	t-o	t-o	38,495	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
b21	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
c7552	1,786	t-o	t-o	766	t-o	t-o	t-o	t-o	t-o	t-o	41,721	t-o	t-o	t-o
ex1010	448	4,357	938	87	719	11,736	24,727	1,703	t-o	129,290	169—7,073§	1,950	t-o	t-o
log2	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
pci_bridge32	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o

*Number of cloaked functions; refer to Table III or the related publication for the actual sets of cloaked functions. Prior art covering the same set is grouped into one column. †Here we refer to their camouflaging primitive, not the polymorphic gate reported on in Table III. ‡Here we also assume BUF to be available. §The benchmark *ex1010* can be resolved even for 100% IP protection, but only for the primitives of [41, c], [81]. The related runtime range is for 40–100% protection, whereas all other runtimes are for 40% protection (50% protection or more ran into timeout).

TABLE VI
RUNTIME FOR *Double DIP* ATTACKS [4], ON SELECTED DESIGNS, IN SECONDS (TIME-OUT "T-O" IS 48 HOURS, I.E., 172,800 SECONDS)

Benchmark	10% IP Protection							20% IP Protection						
	[36] (3)*	[37], [42] (6)*	[12] (4)*†	[41, c], [81] (2)*	[40], [41, a] (4)*	[13] (7+1)*‡	Our (16)*	[36] (3)*	[37], [42] (6)*	[12] (4)*†	[41, c], [81] (2)*	[40], [41, a] (4)*	[13] (7+1)*‡	Our (16)*
aes_core	1,814	27,274	3,039	431	2,103	22,936	53,434	24,635	t-o	55,699	1,631	34,040	t-o	t-o
b14	4,866	47,303	8,197	344	8,299	52,657	t-o	62,698	t-o	138,809	4,757	t-o	t-o	t-o
b21	14,671	t-o	84,483	2,095	47,937	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
c7552	58	763	153	19	173	1,919	23,632	639	t-o	31,485	199	111,580	t-o	t-o
ex1010	126	470	194	27	153	627	1,897	396	3,280	628	94	560	4,361	11,660
log2	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
pci_bridge32	5,389	t-o	t-o	6,888	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
	30% IP Protection							40–100% IP Protection§						
aes_core	59,487	t-o	t-o	14,497	t-o	t-o	t-o	t-o	t-o	t-o	28,228	t-o	t-o	t-o
b14	t-o	t-o	t-o	39,128	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
b21	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
c7552	t-o	t-o	t-o	22,521	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
ex1010	1,247	17,143	3,305	192	1,842	60,970	t-o	8,226	t-o	102,512	396—42,543§	7,120	t-o	t-o
log2	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
pci_bridge32	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o

Refer to Table V for footnotes.

(as also discussed in [49]), not on low corruptibility. We have substantiated this claim with a comprehensive study above.

V. SECURITY ANALYSIS FOR THE PROBABILISTIC REGIME

So far we have leveraged the GSHE primitive in the context of classical, deterministic computation. In this section, we explore the implications of probabilistic computing for IP protection, which have been largely ignored until now.

A. *Conventional SAT Attacks on Probabilistic Circuits*

Consider an obfuscated, hybrid circuit in which some of the gates are probabilistic GSHE gates, while the rest are either implemented in CMOS or as deterministic GSHE gates (see also Sec. VI). Any probabilistic GSHE gate might function erroneously at any point in time, possibly corrupting the overall circuit output. When such a probabilistic circuit is leveraged as an oracle, it might not always faithfully produce the originally intended I/O patterns. Which of the individual patterns is erroneous, however, remains incomprehensible to an attacker who has yet to resolve the obfuscated functionality of the circuit. Without a deterministic oracle to prune only the incorrect keys, SAT attacks may observe conflicting hints or falsely prune the correct key—a conventional SAT attack inevitably tends to fail for probabilistic circuits.

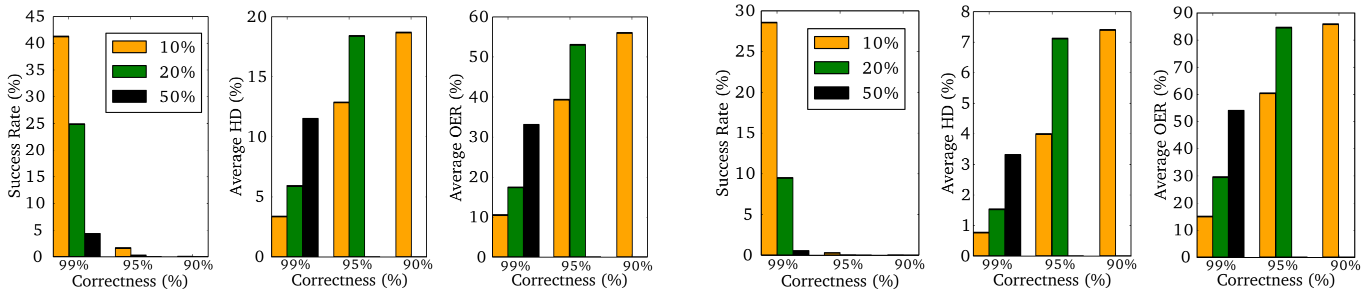
As for a simple example, consider *c17* of Fig. 2 again, but assume that the gate X_6 is being replaced by a probabilistic GSHE NAND gate with an error rate of 5%. Since X_6 impacts the primary output O_2, the oracle will produce correct outputs only for 95% of all inputs. Now, for a conventional SAT attack, such I/O errors can result in false pruning of keys (Table VII).

1) Experimental Setup: To verify this intuition, we prepare the following experiments (with the setup described in Sec. IV-C1 serving as a general baseline). Without loss of generality, we pick *c432* and *c880* from the *ISCAS-85* benchmark suite. On those circuits, we apply a simple obfuscation scheme, namely a random insertion of 32 XOR/XNOR key gates. Note that such simple obfuscation can be readily resolved for deterministic circuits when using any SAT attack, especially for such relatively small circuits. Next, we compile probabilistic versions for those obfuscated circuits. To do so, we randomly select, memorize, and replace fixed subsets of

TABLE VII

SAT ATTACK ON THE BENCHMARK CIRCUIT *c17*, LOCKED AS IN FIG. 2, BUT WITH GATE X_6 NOW ACTING PROBABILISTICALLY, WITH 5% ERROR RATE. AS IN TABLE I, THE LABELS k_0–k_7 REPRESENT ALL POSSIBLE COMBINATIONS OF KEY BITS, FROM 000 TO 111, AND THE COLUMNS DENOTE THE CORRESPONDING, MOST PROBABLE OUTPUTS, WHICH ARE COMPARED WITH THE ORACLE OUTPUT. THE ORACLE OUTPUT IS NOW PROBABILISTIC.

Input patterns	**Oracle Output 0.95%/0.05%**	**Most Probable Output for Different Key Combinations**								**Inference**
$\mathbf{I_1I_2I_3I_4I_5}$	$\mathbf{O_1O_2}$	k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	
00000	00/01	01	00	10	11	00	01	10	11	
00001	01/00	00	01	10	11	01	00	10	11	
00010	11/10	10	11	01	00	10	11	00	01	
00011	11/10	10	11	00	01	10	11	01	00	
00100	00/ 01	01	00	10	11	00	01	10	11	Iteration 1: probabilistic oracle assumes incorrect output $\Rightarrow k_1, k_2, k_3, k_4, k_6, k_7$ are (falsely) pruned
00101	01/00	00	01	10	11	01	00	10	11	
00110	11/00	10	11	01	00	10	11	00	01	
00111	11 /00	10	11	00	01	10	11	01	00	Iteration 2: probabilistic oracle assumes correct output $\Rightarrow k_1$ pruned $\Rightarrow k_5$ is inferred as (only seemingly) correct key
...	...	...	...	...	...	...	...	...	...	
11111	10/11	11	10	10	11	11	10	10	11	



(a) Circuit *c432* (b) Circuit *c880*

Fig. 10. Success rate, average HD, and average OER for 10,000 of the conventional SAT attack [3], [82] applied on probabilistic versions of the *ISCAS-85* benchmarks. The legend applies to all plots, and it represents the range of randomly selected probabilistic gates: 10%, 20%, and 50% of all gates, respectively. The random selection is re-applied for all setups (including those in Fig. 11), for a fair comparison. Correctness is the inverse of the gate error rate; for each setup, the respective correctness is constant for all probabilistic gates. For cases where the attack success rate is zero, HD and OER are not available.

gates (50%, 20%, and 10% of all gates) with probabilistic GSHE gates.

We model the probabilistic gate behavior directly within the conventional SAT attack by Subramanyan *et al.* [2], whose open-source framework [82] allows for such customization. Note that we provide our modification of [82], along with the probabilistic benchmark versions, as open source as well [3].

We run our modified but conventional SAT attack [3] 10,000 times each on three different setups for the probabilistic circuits, assuming error rates of 1%, 5%, and 10%, respectively. For simplicity, we assign identical error rates for all the probabilistic gates.[4] Whenever an attack is successful, we also evaluate the Hamming distance (HD) and output error rate (OER) between the outputs of the original but probabilistic circuit and the probabilistic circuit as resolved after the attack. We apply 10,000 random patterns for averaging the HD and OER (here and for all subsequent setups). It is important to note that HD and OER provide a combined measure of error for both the key inferred by SAT attacks and the probabilistic nature of the circuits under consideration.

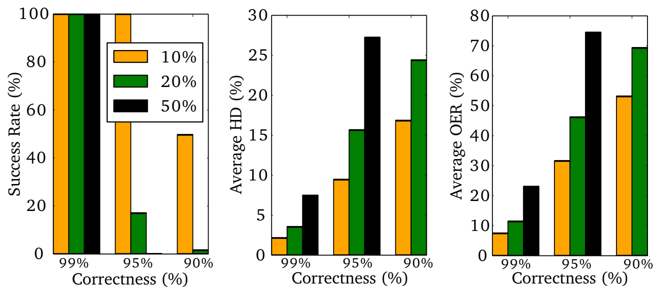
2) Results: The outcome of these experiments is presented in Fig. 10. Besides the aspects discussed in Sec. IV-C3, here we further note that the success rate for the conventional SAT attack depends on (i) the number of probabilistic gates, (ii) the type/function of those gates, and (iii) their error rates. As expected, the success rate decreases and the average HD inflates once the error rates are ramped up. In fact, for some cases with error rates of 5% or 10%, the success rate is even zero (and for such cases, HD and OER cannot be calculated). In short, when tackling obfuscated probabilistic circuits, the capabilities of a conventional SAT attack are drastically reduced once the correctness of the circuit is lowered.

This finding clearly illustrates the trade-off between correctness/accuracy and security. (Also recall Sec. II-B and Sec. III-B for the trade-off between accuracy and power.) Hence, whenever the designer seeks to forgo accuracy (typically to reduce power), the resilience against conventional SAT attacks is inherently boosted at the same time.

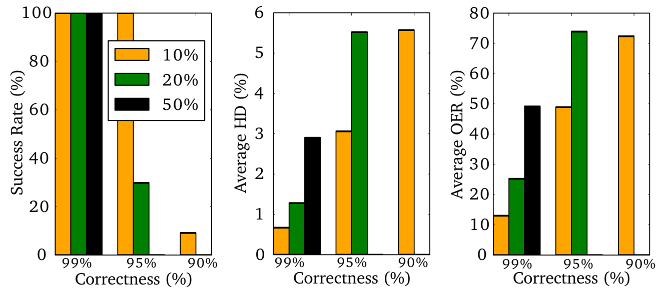
B. PSAT: Our Probabilistic SAT Attack

Having demonstrated the ineffectiveness of the conventional SAT framework against probabilistic circuits, here we present our advanced attack, called PSAT. As indicated above, when tackling probabilistic circuits, the main challenge for any SAT attack is to correctly prune the search space despite potential errors in the oracle's output. An attacker cannot know in advance which output patterns are erroneous (since the circuit is obfuscated and, hence, initially of unknown functionality). However, she/he can apply each input pattern multiple times, track the statistical distribution of the output patterns, and pick only the most prevailing outputs as *ground truths* while

[4]Using our setup [3], one can, however, apply different error rates for different gates, if considered useful for the design under consideration.



(a) Circuit *c432*



(b) Circuit *c880*

Fig. 11. Success rate, average HD, and average OER for 10,000 runs of our PSAT attack [3] applied on probabilistic versions of the *ISCAS-85* benchmarks. The legend applies to all plots, and it represents the range of randomly selected probabilistic gates: 10%, 20%, and 50% of all gates, respectively. The random selection is re-applied for all setups (including those in Fig. 10), for a fair comparison. Correctness is the inverse of the gate error rate; for each setup, the respective correctness is constant for all probabilistic gates. For cases where the attack success rate is zero, HD and OER are not available.

pruning the search space during the SAT attack. The basis for this assumption is that, in any probabilistic circuit, the overall error rate should be constrained such that the correct output patterns occur more frequently than incorrect ones. Otherwise, the circuit would become too approximate and might behave even arbitrarily at some point.

Accordingly, the fundamental concept of PSAT is *Monte Carlo sampling*. When querying the (probabilistic) oracle during the incremental SAT attack flow, we apply each input pattern 1,000 times, without loss of generality, and we track all resulting outputs. Once this sampling is done, we sort the various output patterns by their number of occurrence. In case a *dominant pattern* is established, we readily select this pattern as ground truth and proceed with the SAT attack. An output pattern is considered dominant if and only if it occurs at least as many times as the second and third most frequent patterns combined. Otherwise, in case no dominant pattern is observed, we randomly select among all observed patterns, but while considering their occurrence statistics. For example, if an output pattern has been observed for 27% of all oracle queries using the same input, this particular pattern has a 27% random chance to be selected as ground truth.

1) Experimental Setup: Similar as in Sec. V-A, we implement PSAT as an extension for the open-source framework of [2], [82], and also here we release our PSAT extension to the community as open source in return [3]. Along with the core techniques of PSAT, we provide supplementary features such as the conversion of regular gates to probabilistic/polymorphic ones in the *.bench* file format, simulation of probabilistic/polymorphic gates, the sampling of Hamming distances, etc. To enable a fair comparison between the conventional SAT attack and PSAT, we use the same probabilistic benchmark versions as in Sec. V-A, and we execute PSAT likewise for 10,000 times.

2) Results: As can be seen from Fig. 11, PSAT is notably more successful in resolving the obfuscation of the given probabilistic circuits than the conventional attack. For example, for 50% of all gates being probabilistic ones with an error rate of 1%, the conventional attack succeeded only for 4.3% and 0.5% of the 10,000 attack runs on *c432* and *c880* respectively. In contrast, PSAT holds a success rate of 100% for those cases. PSAT can resolve the underlying obfuscation with fewer errors, that is, the inferred key is more accurate, and the behavior of the recovered IP matches matches more closely the original IP. This is particularly the case for larger ranges of obfuscation using probabilistic gates. For example, for 50% of all gates being probabilistic ones with an error rate of 1%, the circuit *c432* as recovered by the conventional attack has an HD and OER of 11.5% and 33.1%, whereas the same circuit as recovered by PSAT has an improved HD and OER of 7.5% and 23.0%, respectively. However, also PSAT it is challenged once the error rate of the probabilistic gates increases to 10%. That is because, for such a large error rate, it is difficult to establish dominant patterns from the probabilistic oracle.

These results reinforce our earlier argument on the trade-off between accuracy, power, and security against SAT attacks. It is important to note that with excessive errors (of 10% or more), the computational accuracy is rather low, limiting the practicability of such overly imprecise circuits to begin with.

C. PSAT on Polymorphic Circuits

Consider an embedded, reconfigurable design with dynamic time-sharing circuitry. That is, a single circuit is tailored to perform all required operations serially on a time-sharing basis. Such "template circuitry" can be readily implemented when leveraging the runtime polymorphism of GSHE-based circuits. Moreover, operating the GSHE gates in the probabilistic regime would reduce the power consumption as well, rendering such circuitry a viable scheme for low-power and error-tolerant Internet-of-Things (IoT) applications [85].

For such a scenario, even an advanced attack like PSAT may become ineffective. That is because of the underlying principle of Monte Carlo sampling, which renders PSAT relatively slow, depending on how many times each input pattern is to be evaluated and how fast the oracle can be physically queried. Hence, for any iteration of the PSAT attack, the reconfigurable GSHE circuitry might have already morphed from one logic structure to another, resulting in an inconsistent oracle behavior. In turn, this is likely to induce unsatisfiable assignments for the SAT model, causing PSAT to fail.

1) Experimental Setup: Prior setups are used as a baseline here. We generate polymorphic versions of the benchmark circuits *c432* and *c880* as follows. First, to provide a fair baseline, we re-apply the same random selection of gates as when picking 10% for probabilistic gates in Sec. V-A and Sec. V-B. Next, we configure each of those gates as polymorphic GSHE gates, while also accounting for their true functionality, to avoid excessive overall errors. That is, any NAND, AND, NOR, OR, XOR, or XNOR gate is replaced

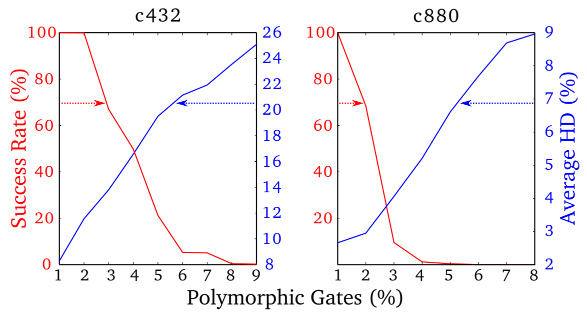


Fig. 12. Success rate (left axis, red) versus Hamming distance (HD, right axis, blue) for PSAT tackling circuits embedded with some polymorphic gates.

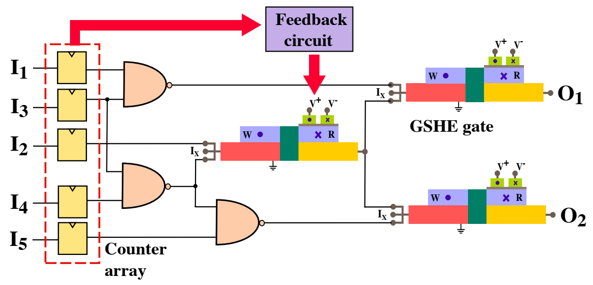


Fig. 13. Concept for a counter-based security mechanism. The input lines hold a counter to monitor suspicious pattern insertion. For clarity, the GSHE device has been laid out horizontally.

by a polymorphic NAND/AND/NOR/OR/XOR/XNOR gate, whereas the original functionality has a probability of 66.67% assigned, and all other functions a probability of 6.67%. Likewise, any INV or BUF is replaced by a polymorphic INV/BUF, whereas the original functionality has a probability of 66.67% assigned as well. We extend our PSAT framework [3] to account for such polymorphic behavior. Using these baseline benchmarks containing 10% polymorphic gates, we derive further benchmarks with 9% down to 1% polymorphic gates, in steps of 1%. We do so by randomly selecting polymorphic gates and reverting them to regular gates. Like in the prior setups, we run PSAT 10,000 times for each benchmark.

2) Results: As one would expect, PSAT can successfully resolve smaller scales of polymorphic obfuscation (Fig. 12). For larger scales, however, the success rate is limited. For those cases where PSAT is successful, we also observe larger HD values than for the probabilistic scenario in Sec. V-B. That is because polymorphic gates tend to induce larger overall errors.

Similar to the probabilistic scenario, there is a trade-off between accuracy, power, and attack resilience. While these experiments have shown a strong resilience against PSAT, we caution that once polymorphic gates were used in a more predictable manner (as for example envisioned above for some embedded and reconfigurable design), an attacker could tailor PSAT accordingly. Thus, once the designer seeks to employ polymorphic gates while keeping the error in bounds, there may be further protection measures required.

VI. Discussion

A. Error Control for Probabilistic Circuits

Here we envision a mechanism to detect repetitive sampling and to subsequently scramble the statistical properties of the GSHE circuit at runtime. For example, consider the modified circuit *c17* in Fig. 13, with three probabilistic NAND gates. The error rates for the primary outputs O_1 and O_2 can be altered by tuning the MTJ voltages of the GSHE gate in the previous stage (recall Sec. III-A). This feature is exploited in the conceptional feedback mechanism in Fig. 13, where a counter array placed at the primary inputs checks the applied input patterns for overly repetitive patterns (or any other deviation from the expected, application-specific inputs distribution, for that matter). In case such suspicious behavior is observed, the feedback mechanism shall dynamically adapt the MTJ voltage supply for the middle GSHE gate. This would alter this gate's output current and, in turn, impact the error rate for the following two GSHE gates. Those latter two gates then directly impact the overall circuit error rate, which can thwart advanced attacks such as PSAT.

The feedback mechanism itself should be implemented using deterministic but obfuscated GSHE gates which could help to obstruct removal attacks. Moreover, the feedback mechanism should be integrated into the circuit in such a way that even advanced removal attacks would result in fully stochastic circuit behavior. For example, the mechanism could be implemented such that removing it would also cut off the control current for the GSHE gates, which leads to unpredictable behavior of those gates.

B. Other Advanced Attack Schemes

Besides our PSAT framework [3], we acknowledge that other attack schemes might be extended as well toward resolving obfuscation in probabilistic and polymorphic circuits. Although tailored to attack SAT-resilient schemes, *AppSAT*, which was recently proposed by Shamsi *et al.* [51], is of particular interest. That is because AppSAT is based on the probably-approximately-correct (PAC) paradigm, which can tolerate some errors.[5] However, the attack is still relying on a consistent oracle behavior—an assumption which probabilistic circuits do not adhere to. At the time of writing, the source code of AppSAT has not been available to us; hence, we were unable to incorporate modeling of probabilistic and polymorphic circuit behavior into AppSAT.

More broadly, machine learning is increasingly being used for both developing new attacks (e.g., [86]) and to defend against attacks (e.g., [87]). We believe that machine learning can be used to augment SAT attacks, to make them even more powerful. Whether such attacks will be sufficiently robust and capable against probabilistic and polymorphic obfuscation schemes, however, remains to be seen. For our scheme, we would like to point out that (i) the GSHE device experiences thermally induced stochasticity, i.e., truly random behavior [17], (ii) the error rate for any device can be tuned individually, and (iii) those individual error distributions superpose with each other while they propagate throughout the entire circuit.

C. Secure and Delay-aware GSHE-CMOS Integration

Finally, we outline the prospects for securing industrial circuits using a hybrid GSHE-CMOS approach. While the

[5]Shamsi *et al.* tackle so-called compound schemes, e.g., [45], [46]. These schemes combine regular, large-error obfuscation techniques with provably secure, low-error obfuscation techniques. Shamsi *et al.* have shown that PAC can help to reduce these schemes to their low-error obfuscation component.

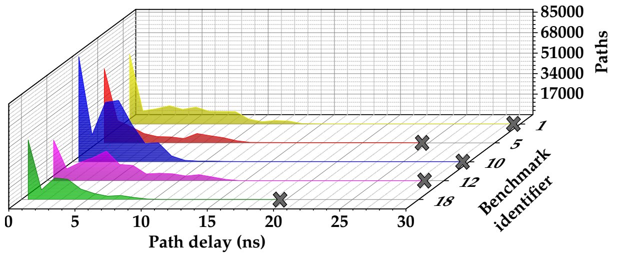


Fig. 14. Delay distributions of selected *IBM superblue* circuits. For clarity, the paths with the critical delays are marked with crosses.

manufacturing of spin devices is still in nascent stages [88]–[90], such a hybrid approach appears practical, given the CMOS-compatible processing of spin devices [13], [88], [91].

On the one hand, recall that the delay for the GSHE device is larger than for regular CMOS devices (Sec. III-B). On the other hand, note that large industrial circuits tend to exhibit a skewed distribution of timing paths, with most paths imposing short delays, and only a few paths inducing critical delays (Fig. 14). Here we explore a delay-aware approach for protecting the industrial *IBM superblue* circuits [84].[6] In short, we replace CMOS gates in the non-critical paths with an obfuscated, deterministic GSHE primitive, as long as no delay overheads are incurred. Doing so, we can obfuscate on average 5–15% of all the gates in the benchmarks.

Conducting the conventional SAT attacks [2], [82] on those GSHE-augmented (but fully deterministic) designs, we observe that they cannot be resolved within 240 hours. In fact, most attack trials incur computational limitations as previously indicated in Sec. IV-C. This suggests that the GSHE primitive can help protecting industrial circuits without excessive layout cost, once such GSHE-CMOS designs are fabricated.

VII. Conclusion

Imprecise computing is rapidly gaining traction due to its attractive low-power characteristics. In this paper, we present the first study exploring the hardware security prospects of imprecise computing systems, specifically for probabilistic and polymorphic circuits constructed with noisy GSHE gates. We design a GSHE-based primitive for IP protection by means of layout obfuscation. We provide not only a thorough security analysis for this primitive, mainly using conventional SAT attacks and our advanced attack PSAT, but we also discuss the inherent resilience of the GSHE device against side channel attacks. A key finding of this study is the following trade-off: the lower the accuracy of imprecise gates, the lower their power consumption, and the better their resilience. Since any design may have practical limitations on the error tolerance, we also promote large-scale obfuscation in the deterministic regime. That is underpinned by another key finding of this study, namely that large-scale obfuscation is competitive to provably secure schemes. Overall, we motivate any security-concerned designer to consider imprecise computing, magnetic devices, and/or large-scale obfuscation for their needs.

[6]To process the layouts for the *IBM superblue* circuits, we leverage scripts provided by Kahng *et al.* [92]. Being sequential circuits, we also have to pre-process them (to mimic access to the scan chains for the SAT attacks): the inputs (and outputs) of any flip-flop are transformed into pseudo-primary outputs (and inputs), whereupon the flip-flops can be removed.

VIII. Acknowledgements

This work was supported in part by the Semiconductor Research Corporation (SRC) and the National Science Foundation (NSF) through ECCS 1740136. The authors also thank the funding support from the MRSEC Program of the National Science Foundation under Award Number DMR-1420073.

References

[1] S. Patnaik *et al.*, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *Proc. Des. Autom. Test Europe*, 2018, pp. 97–102.

[2] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 137–143.

[3] (2018) PSAT by DfX Lab, NYUAD. The password to run the binary is ".stoch". [Online]. Available: https://github.com/DfX-NYUAD/PSAT

[4] Y. Shen *et al.*, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 179–184.

[5] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, vol. 34, pp. 43–98, 1956.

[6] (2007) International technology roadmap for semiconductor. ITRS. [Online]. Available: https://www.semiconductors.org/clientuploads/Research_Technology/ITRS/2007/Design.pdf

[7] J. Sartori *et al.*, "Stochastic computing: embracing errors in architecture and design of processors and applications," in *Proc. Compilers, Arch. and Synth. Emb. Sys.*, 2011, pp. 135–144.

[8] A. Bosio *et al.*, "Approximate computing: Design & test for integrated circuits," in *Proc. Lat.-Am. Test. Symp.*, 2017, pp. 1–1.

[9] D. E. Nikonov *et al.*, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proc. IEEE*, vol. 101, no. 12, pp. 2498–2533, 2013.

[10] M. Rostami *et al.*, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

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

[12] Y. Bi *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.

[13] F. Parveen *et al.*, "Hybrid polymorphic logic gate with 5-terminal magnetic domain wall motion device," in *Proc. Comp. Soc. Symp. VLSI*, 2017, pp. 152–157.

[14] S. Datta *et al.*, "Non-volatile spin switch for Boolean and non-Boolean logic," *Appl. Phys. Lett.*, vol. 101, no. 25, p. 252411, 2012.

[15] A. V. Penumatcha *et al.*, "Impact of scaling on the dipolar coupling in magnet–insulator–magnet structures," *IEEE Trans. Magn.*, vol. 52, no. 1, pp. 1–7, 2016.

[16] ——, "Spin-torque switching of a nano-magnet using giant spin hall effect," *AIP Advances*, vol. 5, no. 10, p. 107144, 2015.

[17] N. Rangarajan *et al.*, "Energy-efficient computing with probabilistic magnetic bits – performance modeling and comparison against probabilistic CMOS logic," *IEEE Trans. Magn.*, vol. 53, no. 11, pp. 1–10, 2017.

[18] J. Han *et al.*, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. Europe Test. Symp.*, 2013, pp. 1–6.

[19] A. Alaghi *et al.*, "The promise and challenge of stochastic computing," *TCAD*, vol. 37, no. 8, pp. 1515–1531, 2018.

[20] B. R. Gaines *et al.*, "Stochastic computing systems," *Advances in Information Systems Science*, vol. 2, no. 2, pp. 37–172, 1969.

[21] A. Alaghi *et al.*, "Survey of stochastic computing," *Trans. Emb. Circ. Sys.*, vol. 12, no. 2s, p. 92, 2013.

[22] ——, "Stochastic circuits for real-time image-processing applications," in *Proc. Des. Autom. Conf.*, 2013, pp. 1–6.

[23] S. Gaba *et al.*, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, pp. 5872–5878, 2013.

[24] B. Moons *et al.*, "Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies," *J. Emerg. Sel. Topics Circ. Sys.*, vol. 4, no. 4, pp. 475–486, 2014.

[25] V. Gupta *et al.*, "Low-power digital signal processing using approximate adders," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 32, no. 1, pp. 124–137, 2013.

[26] ——, "IMPACT: Imprecise adders for low-power approximate computing," in *Proc. Int. Symp. Low Power Elec. Design*, 2011, pp. 409–414.

[27] R. Venkatesan *et al.*, "MACACO: Modeling and analysis of circuits for approximate computing," in *Proc. Int. Conf. Comp.-Aided Des.*, 2011, pp. 667–673.

[28] S. Venkataramani *et al.*, "AxNN: energy-efficient neuromorphic systems using approximate computing," in *Proc. Int. Symp. Low Power Elec. Design*, 2014, pp. 27–32.

[29] N. Lee *et al.*, "Towards formal evaluation and verification of probabilistic design," *Trans. Comp.*, vol. 67, no. 8, pp. 1202–1216, 2018.

[30] S. Cheemalavagu *et al.*, "A probabilistic CMOS switch and its realization by exploiting noise," in *Proc. IFIP VLSI*, 2005, pp. 535–541.

[31] B. Behin-Aein *et al.*, "Proposal for an all-spin logic device with built-in memory," *Nature Nanotechnology*, vol. 5, no. 4, pp. 266–270, 2010.

[32] W. F. Brown Jr, "Thermal fluctuations of a single-domain particle," *Physical Review*, vol. 130, no. 5, p. 1677, 1963.

[33] M. dAquino *et al.*, "Midpoint numerical technique for stochastic Landau-Lifshitz-Gilbert dynamics," *J. Appl. Phys.*, vol. 99, no. 8, p. 08B905, 2006.

[34] R. Karri *et al.*, "Physical unclonable functions and intellectual property protection techniques," in *Fundamentals of IP and SoC Security*. Springer, 2017, pp. 199–222.

[35] A. Vijayakumar *et al.*, "Physical design obfuscation of hardware: A comprehensive investigation of device- and logic-level techniques," *Trans. Inf. Forens. Sec.*, vol. 12, no. 1, pp. 64–77, 2017.

[36] J. Rajendran *et al.*, "Security analysis of integrated circuit camouflaging," in *Proc. Comp. Comm. Sec.*, 2013, pp. 709–720.

[37] I. R. Nirmala *et al.*, "A novel threshold voltage defined switch for circuit camouflaging," in *Proc. Europe Test. Symp.*, 2016, pp. 1–2.

[38] B. Erbagci *et al.*, "A secure camouflaged threshold voltage defined logic family," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2016, pp. 229–235.

[39] S. Patnaik *et al.*, "Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging," in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 41–48.

[40] Y. Zhang *et al.*, "Giant spin hall effect (GSHE) logic design for low power application," in *Proc. Des. Autom. Test Europe*, 2015, pp. 1000–1005.

[41] Q. Alasad *et al.*, "Leveraging all-spin logic to improve hardware security," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 491–494.

[42] T. Winograd *et al.*, "Hybrid STT-CMOS designs for reverse-engineering prevention," in *Proc. Des. Autom. Conf.*, 2016, pp. 88–93.

[43] J. Yang *et al.*, "Exploiting spin-orbit torque devices as reconfigurable logic for circuit obfuscation," *TCAD*, 2018.

[44] J. A. Roy *et al.*, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[45] M. Yasin *et al.*, "SARLock: SAT attack resistant logic locking," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2016, pp. 236–241.

[46] Y. Xie *et al.*, "Mitigating SAT attack on logic locking," in *Proc. Cryptogr. Hardw. Embed. Sys.*, 2016, pp. 127–146.

[47] P. Tuyls *et al.*, "Read-proof hardware from protective coatings," in *Proc. Cryptogr. Hardw. Embed. Sys.*, 2006, pp. 369–383.

[48] S. Anceau *et al.*, "Nanofocused X-ray beam to reprogram secure circuits," in *Proc. Cryptogr. Hardw. Embed. Sys.*, 2017, pp. 175–188.

[49] M. E. Massad *et al.*, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. Netw. Dist. Sys. Sec. Symp.*, 2015, pp. 1–14.

[50] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. Int. Conf. Comp.-Aided Des.*, 2016, pp. 28:1–28:8.

[51] K. Shamsi *et al.*, "On the approximation resiliency of logic locking and ic camouflaging schemes," *Trans. Inf. Forens. Sec.*, 2018.

[52] X. Xu *et al.*, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Proc. Cryptogr. Hardw. Embed. Sys.*, 2017.

[53] H. Wang *et al.*, "Probing attacks on integrated circuits: Challenges and research opportunities," *J. Des. Test*, vol. 34, no. 5, pp. 63–71, 2017.

[54] S. Skorobogatov *et al.*, "In the blink of an eye: There goes your AES key," in *IACR Crypt. ePrint Arch.*, no. 296, 2012.

[55] F. Courbon *et al.*, "Direct charge measurement in floating gate transistors of flash EEPROM using scanning electron microscopy," in *Proc. Int. Symp. Test. Failure Analys.*, 2016, pp. 1–9.

[56] G. Weissenbacher *et al.*, "Boolean satisfiability: Solvers and extensions," *Software Systems Safety*, vol. 36, p. 223, 2014.

[57] M. W. Moskewicz *et al.*, "CHAFF: Engineering an efficient SAT solver," in *Proc. Des. Autom. Conf.*, 2001, pp. 530–535.

[58] I. Mironov *et al.*, "Applications of SAT solvers to cryptanalysis of hash functions," in *Proc. Theory Appl. SAT Test.*, vol. 4121. Springer, 2006, pp. 102–115.

[59] C. Yu *et al.*, "Incremental SAT-based reverse engineering of camouflaged logic circuits," *TCAD*, vol. 36, no. 10, pp. 1647–1659, 2017.

[60] X. Wang *et al.*, "A conflict-free parallelization framework for SAT-based de-camouflaging attack," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2018.

[61] M. Yasin *et al.*, "Transforming between logic locking and IC camouflaging," in *Proc. Des. Test Symp.*, 2015, pp. 1–4.

[62] S. Koteshwara *et al.*, "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.

[63] J. T. McDonald *et al.*, "Functional polymorphism for intellectual property protection," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2016, pp. 61–66.

[64] I. M. Miron *et al.*, "Current-driven spin torque induced by the rashba effect in a ferromagnetic metal layer," *Nature materials*, vol. 9, no. 3, p. 230, 2010.

[65] S. Fukami *et al.*, "Magnetization switching by spin–orbit torque in an antiferromagnet–ferromagnet bilayer system," *Nature materials*, vol. 15, no. 5, p. 535, 2016.

[66] H. Almasi *et al.*, "Enhanced tunneling magnetoresistance and perpendicular magnetic anisotropy in mo/cofeb/mgo magnetic tunnel junctions," *Appl. Phys. Lett.*, vol. 106, no. 18, p. 182406, 2015.

[67] P. Li *et al.*, "Electric field manipulation of magnetization rotation and tunneling magnetoresistance of magnetic tunnel junctions at room temperature," *Advanced Materials*, vol. 26, no. 25, pp. 4320–4325, 2014.

[68] J. Hirsch, "Spin hall effect," *Phys. Rev. Lett.*, vol. 83, no. 9, p. 1834, 1999.

[69] N. Kani *et al.*, "A model study of an error-free magnetization reversal through dipolar coupling in a two-magnet system," *IEEE Trans. Magn.*, vol. 52, no. 2, pp. 1–12, 2016.

[70] H. Maehara *et al.*, "Tunnel magnetoresistance above 170% and resistance–area product of 1 $\Omega(\mu\text{m})^2$ attained by in situ annealing of ultra-thin MgO tunnel barrier," *Appl. Phys. Express*, vol. 4, no. 3, p. 033002, 2011.

[71] Q. Hao *et al.*, "Giant spin hall effect and switching induced by spin-transfer torque in a W/$\text{Co}_{40}\text{Fe}_{40}\text{B}_{20}$/MgO structure with perpendicular magnetic anisotropy," *Phys. Rev. Appl.*, vol. 3, no. 3, p. 034009, 2015.

[72] K. Huang *et al.*, "Magnetic domain-wall racetrack memory-based nonvolatile logic for low-power computing and fast run-time-reconfiguration," *Trans. VLSI Syst.*, vol. 24, no. 9, pp. 2861–2872, 2016.

[73] S. Manipatruni *et al.* (2015) Spin-orbit logic with magnetoelectric nodes: A scalable charge mediated nonvolatile spintronic logic. [Online]. Available: https://arxiv.org/abs/1512.05428

[74] R. M. Iraei *et al.*, "Electrical-spin transduction for CMOS-spintronic interface and long-range interconnects," *J. Explor. Sol.-St. Comp. Dev. Circ.*, vol. 3, pp. 47–55, 2017.

[75] C. McCants, "Trusted integrated chips (TIC)," Intelligence Advanced Research Projects Activity (IARPA), Tech. Rep., 2011. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/tic

[76] S. Patnaik *et al.*, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2018, pp. 251–258.

[77] F. Imeson *et al.*, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," in *Proc. USENIX Sec. Symp.*, 2013, pp. 495–510.

[78] S. E. Quadir *et al.*, "A survey on chip to system reverse engineering," *J. Emerg. Tech. Comp. Sys.*, vol. 13, no. 1, pp. 6:1–6:34, 2016.

[79] S. Tajik *et al.*, "On the power of optical contactless probing: Attacking bitstream encryption of FPGAs," in *Proc. Comp. Comm. Sec.*, 2017, pp. 1661–1674.

[80] A. Schlösser *et al.*, "Simple photonic emission analysis of AES," in *Proc. Cryptogr. Hardw. Embed. Sys.*, 2012, pp. 41–57.

[81] J. Zhang, "A practical logic obfuscation technique for hardware security," *Trans. VLSI Syst.*, vol. 24, no. 3, pp. 1193–1197, 2016.

[82] P. Subramanyan. (2017) Evaluating the security of logic encryption algorithms. [Online]. Available: https://bitbucket.org/spramod/host15-logic-encryption

[83] L. Amarù. (2015) Majority-inverter graph (MIG) benchmark suite. [Online]. Available: http://lsi.epfl.ch/MIG

[84] N. Viswanathan *et al.*, "The ISPD-2011 routability-driven placement contest and benchmark suite," in *Proc. ISPD*, 2011, pp. 141–146.

[85] D. Blaauw *et al.*, "IoT design space challenges: Circuits and systems," in *Proc. VLSI Tech.*, 2014, pp. 1–2.

[86] L. Lerman *et al.*, "Side channel attack: an approach based on machine learning." Center for Advanced Security Research Darmstadt, 2011, pp. 29–41.

[87] M. Sabhnani *et al.*, "Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context," in *Proc. MLMTA*, 2003, pp. 209–215.

[88] S. Matsunaga *et al.*, "Fabrication of a nonvolatile full adder based on logic-in-memory architecture using magnetic tunnel junctions," *Applied Physics Express*, vol. 1, no. 9, p. 091301, 2008.

[89] S.-h. C. Baek *et al.*, "Complementary logic operation based on electric-field controlled spin-orbit torques," *Nature Electronics*, vol. 1, no. 7, pp. 398–403, 2018.

[90] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *TCAD*, vol. 35, no. 1, pp. 1–22, 2016.