Programmable Gates Using Hybrid CMOS-STT Design to Prevent IC Reverse Engineering

TED WINOGRAD, George Mason University
GAURAV SHENOY, George Mason University
HASSAN SALMANI, Howard University
HAMID MAHMOODI, San Francisco State University
SETAREH RAFATIRAD, George Mason University
HOUMAN HOMAYOUN, George Mason University

Abstract–This paper presents a rigorous step towards design-for-assurance by introducing a new class of logically reconfigurable design resilient to design reverse engineering. Based on the non-volatile spin transfer torque (STT) magnetic technology, we introduce a basic set of non-volatile reconfigurable Look-Up-Table (LUT) logic components (NV-STT-based LUTs). STT-based LUT with significantly different set of characteristics compared to CMOS provides new opportunities to enhance design security yet makes it challenging to remain highly competitive with custom CMOS or even SRAM-based LUT in terms of power, performance and area. To address these challenges, we propose several algorithms to select and replace custom CMOS gates with reconfigurable STT-based LUTs during design implementation such that the functionality of STT-based components and therefore the entire design cannot be determined in any manageable time, rendering any design reverse engineering attack ineffective. Our study conducted on a large number of standard circuit benchmarks concludes significant resiliency of hybrid STT-CMOS circuits against various types of attacks. Furthermore, the selection algorithms on average have a small impact on the performance of the circuit. We also tested these techniques against satisfiability attacks developed recently and show that these techniques also render more advanced reverse-engineering techniques computationally infeasible.

CCS Concepts: • **Security and privacy → Tamper-proof and tamper-resistant designs**; **Hardware attacks and countermeasures**; **Hardware reverse engineering**;

Additional Key Words and Phrases: Reverse Engineering, Hybrid CMOS-STT, Reconfigurable Gates, Hardware Security, Standard IC Design Flow

ACM Reference Format:
Ted Winograd, Gaurav Shenoy, Hassan Salmani, Hamid Mahmoodi, Setareh Rafatirad, and Houman Homayoun. 2010. Programmable Gates Using Hybrid CMOS-STT Design to Prevent IC Reverse Engineering. *ACM Trans. Web* 9, 4, Article 39 (March 2010), 21 pages.
https://doi.org/0000001.0000001

Author's addresses: T. Winograd, G. Shenoy and H. Homayoun, Electrical and Computer Engineering Department, George Mason University. H. Salmani, Electrical Engineering and Computer Science Department, Howard University. H. Mahmoodi, Electrical Engineering Department, San Francisco State University. S. Rafatirad, Department of Information Science and Technology, George Mason University.

1559-1131/2010/3-ART39
https://doi.org/0000001.0000001

1 INTRODUCTION

It is predicted that by 2020 there will be an estimated 50 billion [13] connected devices in the world. Internet of things (IOT) devices are being developed for a wide range of applications from home automation and personal fitness, to smart cities and defence. The economic opportunities offered by massive deployment of IoT devices are vast, but unfortunately IoT security threats grow exponentially along with the count and reach of these devices [1, 19, 43, 45, 47], and therefore threaten the economic opportunities. The severity and impact of a threat increases proportionally to the scale and reach of affected IoT devices. The majority of IoT devices are expected to be small and spatially distributed Systems on Chip (SoC), with power and computational constraints. Adapting complex software solutions for IoT security are not practical mainly due to the cost, power and computational constraints of the IoT devices. More importantly software solutions do not provide defense against manufacturing and post manufacturing physical security threats. The security therefore should be delegated to the underlying hardware, building a bottom-up solution for securing IoT devices rather than treating it as an afterthought. When it comes to IoT devices, hardware implementation of security objectives should incur low overhead in terms of design, area, as well as power consumption, corresponding to resource constrained nature of these devices. Integrated Circuits (ICs) are at the core of IoT devices; their security and trustworthiness ground the security of the entire IoT system. However, the security and trustworthiness of ICs deployed in IoT devices are exacerbated by the modern globalized, horizontal semiconductor business model and in particular for mass volume IoT device designs and fabrication. This model involves many steps performed at multiple locations by different providers and integrates various intellectual properties (IPs) from several vendors, which has become prevalent due to confluence of increasingly complex supply chains, time-to-market delivery, and cost pressures. This trend poses significant challenges to hardware security assurance in various forms and in particular affecting IoT devices. At the design stage, there is a chance of IP piracy and tampering with IP to change its intended functionality. Outsourcing design manufacturing provides significant opportunities for untrusted foundries for tampering, overproducing, and cloning, to name a few. Even after releasing design to the market, the design can be subject to non-invasive reserve engineering, such as side-channel attacks, to obtain secret information during design operation or invasive reserve engineering to obtain detailed design implementation. ICs may experience counterfeiting attacks even after being resigned in the forms of recycling and remarking as well as forging their documentation and selling defective ones.

To protect a circuit against the reverse engineering attack, new design techniques such as tamper detection [9, 37], obfuscation [8, 30, 32] 3D IC technology [10, 18], re-configurable logic [5, 24, 25] split manufacturing [17, 44] and camouflaging [31, 36] have been proposed. The split manufacturing makes it possible to manufacture a front-end-of-line piece of a semiconductor die in a first manufacturing line, and a back-end-of-line piece of a semiconductor die in a second manufacturing line. This technique hides interconnection between gates, so the first manufacturing would not be able to reconstruct the entire circuit. The camouflaging technique hides the logic functionality of gates as a set of camouflaged gates have an identical layout image and it is not possible to determine their functionality just by processing their images.

To reverse engineer circuits that integrate unknown gates such as camouflaged circuits, the RE attack based on satisfiability (SAT) has been also investigated [12, 26]. It has been shown that the SAT-based attack can effectively determine the unknown gates in a few minutes if the number of possible identities for an unknown gate is small (for the camouflaging technique the number is three).

All of these recently proposed techniques either incure significant design overhead or shown to be ineffective against recent machine learning or SAT based reverse engineering attacks [12, 26]. For instance obfuscation [8, 30, 32] , 3D technology [10, 18], and split manufacturing require significant change in the standard IC design flow. These solution while are effective for general purpose IC design, for IoT IC market which cost is a prime concern, they are not effective solutions. An effective solution in this domain should be low cost, requiring minimal changes to the standard IC design flow, and not breakable against recent SAT based attacks.

In response to all of these challenges we recently introduced Hybrid STT-CMOS vanishable logics [42] which combines custom and programmable gates to hide the functionality of certain number of critical gates. In this case, only a design house and trusted vendors know the functionality of programmable gates. The content of STT-based programmable gates is not accessible to an external probe engine as the programming bit are distributed within the combinational logic of the design.

To realize design assurance, Hybrid STT-CMOS vanishable logics leverage the concept of circuit design using reconfigurable logic based on hardware reconfiguration and transformation which recently proposed in [2] [23] and employ highly promising Spin Transfer Torque (STT) magnetic technology to build robust and reverse-engineering resilient Look-Up-Table (LUT) logic components with the goal of enhancing design security and using the STT technology to design on-die, run-time reconfigurable units such as ALU units and logic gates. Spin Transfer Torque (STT) is a new device technology that offers several advantages over conventional CMOS technology, including approximately 4 times higher integration density, non-volatility, near-zero leakage power, high retention time and high endurance, and most of all easy integration with conventional CMOS. Our approach is to design efficient and reconfigurable logic gates using STT LookUp-Tables (LUTs), and then employ them at the gate level to enhance design security. Our previous research has convincingly demonstrated that it is possible to design reconfigurable STT functional units that are competitive to or better than custom CMOS designs, at least for complex functional units (e.g. multipliers) [4, 27]. As shown in our earlier conference paper [42], the STT reconfigurable design is similar in functionality to an FPGA but with significantly higher speed running at GHz frequency, near zero leakage power, high thermal stability, highly integrative with CMOS and overall competitive with custom CMOS design in terms of performance and energy-efficiency. In addition, compared to SRAM-based LUT, STT-based LUT is non-volatile, that is, there is no need to another flash memory (which could be a source of vulnerability) to store the configuration bits to load from on every power up. However compared to SRAM-based LUT, STT-based LUT require significantly higher write current. This new set of characteristics provide new opportunities to enhance design security yet makes it challenging to remain highly competitive with custom CMOS or even SRAM-based LUT in terms of power, performance and area. To the best of our knowledge this is the first work that introduces the concept of reconfigurable STT-based LUT for enhancing design security and highlight the opportunities and challenges with the new technology and design and address these challenges to make it a deployable technology to enhance security yet remain competitive with custom CMOS. To protect a design from design reverse engineering attacks after final product release, depending on the required level of security, we propose several algorithms to select and replace custom CMOS gates in circuit netlist with reconfigurable STT-based LUTs during design implementation. While an untrusted foundry may still have access to the reconfigured design after its release to the market, the selection of custom CMOS gates for replacement ensures that the untrusted foundry cannot determine the functionality of reconfigurable LUTs, and therefore cannot reverse engineer the design in any reasonable time. The selection algorithm will ensure that original design parametric constraints such as design performance will be impacted only minimally. We have modified the tools used to select and analyze these techniques to better match the final results. Our current implementation leverages standard IC design flow such as Synopsys Design

Compiler to calculate necessary information before processing with a custom Python script. We believe Hybrid STT-CMOS is an effective solution in IoT domain, given its minimal overhead, such as power, area and timing implication, and most importantly cost as it require no change to standard IC design flow.

The main contributions of this paper are as follows:

- Introducing the concept of reconfigurable STT-based LUT for enhancing design security and highlighting the opportunities and challenges with the new technology and design to make it a deployable technology,
- Introducing a security driven design flow at gate level to prevent design reverse engineering,
- Proposing several algorithms for selection and replacement of custom CMOS gates with STT-based LUTs, and
- Analyzing performance, area and power overhead on standard circuit benchmarks.
- Analysis of these techniques against SAT attacks as described in [11].

The remainder of this paper is organized as follows. Section 2 presents preliminaries and surveys related work. In Section 3, we explain STT technology. Next, we describe how to employ CMOS-STT technology to realize design for assurance in Section 5. Experimental results are then presented in Section 6. Finally, Section 7 concludes this paper.

2 PREVIOUS WORK

Current techniques for hardware reverse engineering have raised serious concerns in the IC design community, particularly when facing a very high-tech adversary. Reverse engineering can be done at different levels of design abstraction and various phases of system on chip (SoC) design supply chain. An untrusted foundry may compromise the design security by inserting extra circuits as hardware Trojans [41], or extracting IPs used in a circuit and making profits by selling them without knowledge of IP owner [48], or overproducing the design and sell in the black market [15]. Many techniques to counter these attacks have been proposed and many are in active use. Examples of such countermeasures are mislabeling [35], shielding [3], tamper detection [9, 37], obfuscation [8, 30, 32] 3D IC technology [10, 18], re-configurable logic [5, 24, 25] split manufacturing [17, 44] and camouflaging [31, 36] and self-modification (reconfiguration) [22]. While several of these countermeasures are aimed at making an attack more difficult through obfuscation without providing any actual protection (mislabeling, potting, obfuscation, camouflaging), others remove or minimize side-channel leakage. The split manufacturing makes it possible to manufacture a front-end-of-line piece of a semiconductor die in a first manufacturing line, and a back-end-of-line piece of a semiconductor die in a second manufacturing line. This technique hides interconnection between gates, so the first manufacturing would not be able to reconstruct the entire circuit. The camouflaging technique hides the logic functionality of gates as a set of camouflaged gates have an identical layout image and it is not possible to determine their functionality just by processing their images.

All of these recently proposed techniques either incure significant design overhead or shown to be ineffective against recent machine learning or SAT based reverse engineering attacks [12, 26]. For instance obfuscation [8, 30, 32] , 3D technology [10, 18], and split manufacturing require significant change in the standard IC design flow. These solution while are effective for general purpose IC design, for IoT IC market which cost is a prime concern, they are not effective solutions. An effective solution in this domain should be low cost, requiring minimal changes to the standard IC design flow, and not breakable against recent SAT based attacks.

In response to all of these challenges, we propose a solution which relies on hardware reconfigurability. Hardware reconfigurability has been around for several years, primarily in the form

of FPGAs. In [23], using embedded SRAM-based reconfigurable logic for application specific integrated circuits (ASIC) design obfuscation is investigated. SRAM reconfigurable logic blocks provide reconfigurability and potentially enhance security, but they are not practical for use in embedded systems where power and performance are major constraints. Furthermore, SRAM require an external non-volatile memory to keep reconfiguration bitstream which becomes the source of vulnerability. Our idea in this paper of building reconfigurable logic using STT is different from the previous work in various aspects. Firstly, we introduce the non-volatile STT-based look up table design as a new method of realizing the hardware reconfiguration for security and integrating non-volatile STT-based LUTs and custom CMOS gates side by side on the same die. Secondly, we propose a security-driven hybrid STT-CMOS design flow to integrate design assurance with other design constraints and considerations. STT reconfigurable logic has several advantages over reconfigurable CMOS in terms of power, performance, area as well as security metrics. STT-based LUT has substantially lower leakage power compared to CMOS-based LUT [27]. In addition, STT-based LUT has advantage over CMOS-based LUT in terms of performance and area [27, 40]. Also it has a high thermal robustness. From security perspective, STT-based LUT brings two clear advantages over CMOS design: First, due to its non-volatility feature, it holds the reconfiguration bitstream, whereas CMOS-based LUT requires an external non-volatile memory which becomes the source of vulnerability. Second, STT-based LUT power consumption is almost insensitive to its input changes [27, 40], therefore compared to CMOS-based LUT, it is more robust against power-based side channel attacks.

3 STT TECHNOLOGY

In our proposed gate level solution to prevent reverse engineering, the run-time reconfiguration is realized by a LUT based design, which uses Spin Transfer Torque (STT) technology to store the LUT data. The new reconfigurable STT-LUT design not only enables run-time reconfiguration to vanish the logical property of the design, but offers added advantages for complex blocks in terms of power, performance, lifetime reliability and thermal stability compared to non-reconfigurable CMOS implementation.

STT provides i) approximately $4X$ higher integration density than conventional Static Random Access Memory (SRAM) [49], ii) high retention times (even more than 10 years [39]), iii) high endurance (10^{16} writes, or 10 years of operation as L1 cache) [14], iv) near-zero leakage [33] with close-to SRAM read performance, v) excellent thermal robustness 300^oC, vi) soft error resilience, and vii) above all, STT cells are easy to integrate with the conventional CMOS fabrication process.

STT technology, for the first time, provides us the opportunity to design reconfigurable logic gates that are on-die with CMOS logic and have low reconfiguration overhead. Existing Field Programmable Gate Arrays (FPGAs) cannot be used to design on-chip reconfigurable logic, as they are often flash devices that do not integrate well with the conventional CMOS fabric or SRAM-based reconfigurable units. Moreover, the reconfiguration time is long in existing technologies. For example, typical partial reconfiguration time on Virtex 6 FPGA is in the order of tens of milliseconds [20]. An alternative would be to use SRAM based reconfigurable units, but they suffer from problems of scalability, high leakage, high sensitivity to variations, and soft errors [28]. Moreover, SRAM based reconfiguration is volatile and needs to be re-programmed on every power up and this demands a separate non-volatile storage such as a flash memory to store configuration bits which becomes a source of vulnerabilities.

4 DESIGN LOOKUP TABLE BASED RECONFIGURABLE LOGIC IN STT TECHNOLOGY

In this paper we use the STT-LUT design proposed by Suzuki [40] and further improved in our recent work [27]. By loading different values in the LUTs, the reconfigurable fabric is able to implement various logic functions. Moreover, there is added security benefit because the content of the LUTs can be hidden to IC manufacturers or eliminated upon detection of a reverse engineering attempt. Moreover, the content of an LUT cannot be reverse engineered from its physical layout because of its generic and programmable nature. STT-NV technology utilizes Magnetic Tunnel Junctions (MTJ) to realize nonvolatile resistive storage. There have been several attempts to use MTJs for building logic circuits to exploit the leakage benefit of MTJs to reduce the circuit power [40]. However, due to the significant energy involved in changing the state of an MTJ, circuit styles that rely on changing the state of MTJs in response to input changes do not show any power and performance benefits [34]. An alternative to this approach has been to realize logic in memory by using LUTs built based on MTJs [16].

Based on current implementations [27], we show that for small logic gates, the STT-LUT style shows considerable overhead as compared to the custom CMOS implementation; however, as the circuit complexity increases this overhead reduces. The delay overhead is also less for high fan-in NOR gates as their static CMOS implementation would require a series connection of PMOSes in their pull-up networks. PMOS transistors tend to be slower than NMOS transistors and since the STT-LUT style uses less number of PMOS transistors, its benefit is more noticeable for implementation of such logic gates. Further, efforts have been underway to reduce this overhead over the past couple of years. More recently, we made several improvements by using alternative implementations of the STT-LUT, as shown in Figure 3. Previous iterations of this work have leveraged dynamic STT-LUTs, as shown in Figure 1. Our new schemes have since been developed that leverage a static design, as shown in Figure 2. The static STT-LUTs from 2 to 6 fan-ins are characterized in terms of delay, active power, leakage power, Power-Delay-Product (PDP), area and sensing failure rate; and compared with the dynamic STT-LUT. Unlike dynamic STT-LUT, in static STT-LUT, the active power consumption is dependent on the switching activities. So for obtaining a evaluation more realistic power comparison, the power of the STT-LUT designs were measured at 100%, 50% and 25% output switching activities. Figure 3(a) shows the active power comparison which is dependent on switching activities and figure 3(b) shows the leakage power which is measured in idle condition.

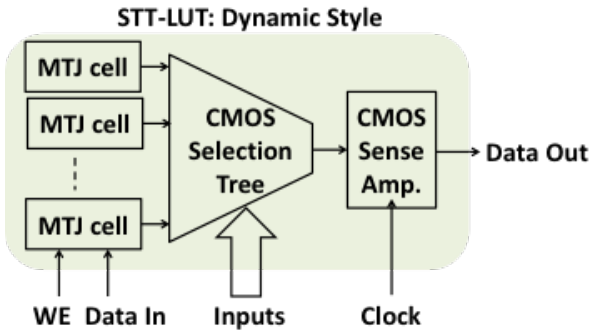

Fig. 1. *STT-LUT design concept: Existing dynamic style*

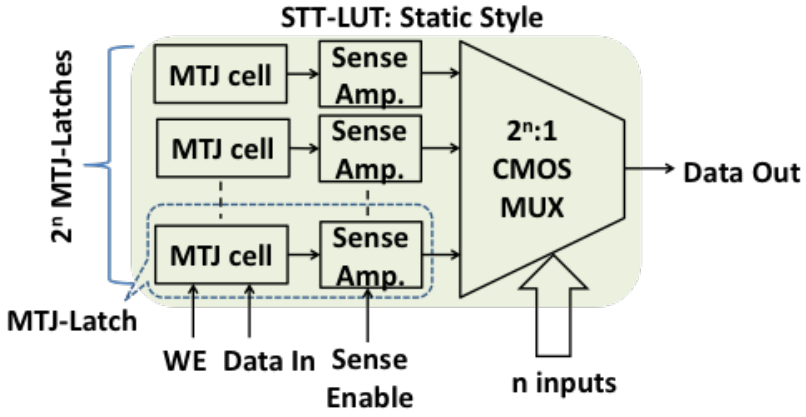

Fig. 2. *STT-LUT design concept: Proposed static style*

Another observation from our current implementations is that the LUT style shows less power overhead for higher data activity (α) (see Figure 4). This is due to the dynamic nature of the STT-LUT style that increases its switching activity making it a better fit for high data activity applications. Note that the power and delay of the STT-LUT is independent of the logic it is programmed to implement (i.e its data content) and also independent of its input data activity. The power and delay of the STT-LUT only depends on its fain-in (number of inputs).

The leakage power of the STT-LUT style is lower than the custom CMOS except for high fan-in NAND and NOR gates. In high fan-in static CMOS NAND (NOR) gates, there is a long chain of series connected NMOS (PMOS) transistors that suppresses leakage via the transistor stacking effect. However, this leakage advantage for such static CMOS gates will disappear if those gates are implemented using cascade of lower fan-in gates for performance reasons. Therefore we can argue that for low fan-in (4-input or less) standard logic gates, the STT-LUT style implementation offers less leakage.

5 SECURITY AND STT TECHNOLOGY

Figure 5 presents our novel security-driven design flow to prevent reverse engineering. While it is fully compatible with the common-practice VLSI design flow, the proposed flow aims to introduce security in the early design stages to prevent design reverse engineering with no or minimum impact on design parametric constraints. Along with the design constraints and the target CMOS technology node, the design security requirements and the STT technology library information are passed to the standard VLSI design flow.

The design flow is continued with circuit implementation and then the logic synthesis. Afterwards, an obtained gate-level netlist from the logic synthesis is passed to our novel *CMOS gate selection and replacement* stage. Depending on the design security requirements, one of our proposed algorithms described in the following section is chosen by the designer. The selected algorithm takes the synthesized gate-level netlist and carefully selects a number of CMOS gates and replace them with equivalent STT-based LUT implementation. In this context, we use STT-based LUTs, reconfigurable units and missing gates interchangeably. We refer to the obtained netlist as a hybrid netlist. After obtaining the hybrid netlist, the design flow is continued with the physical design, and then the design is signed-off.

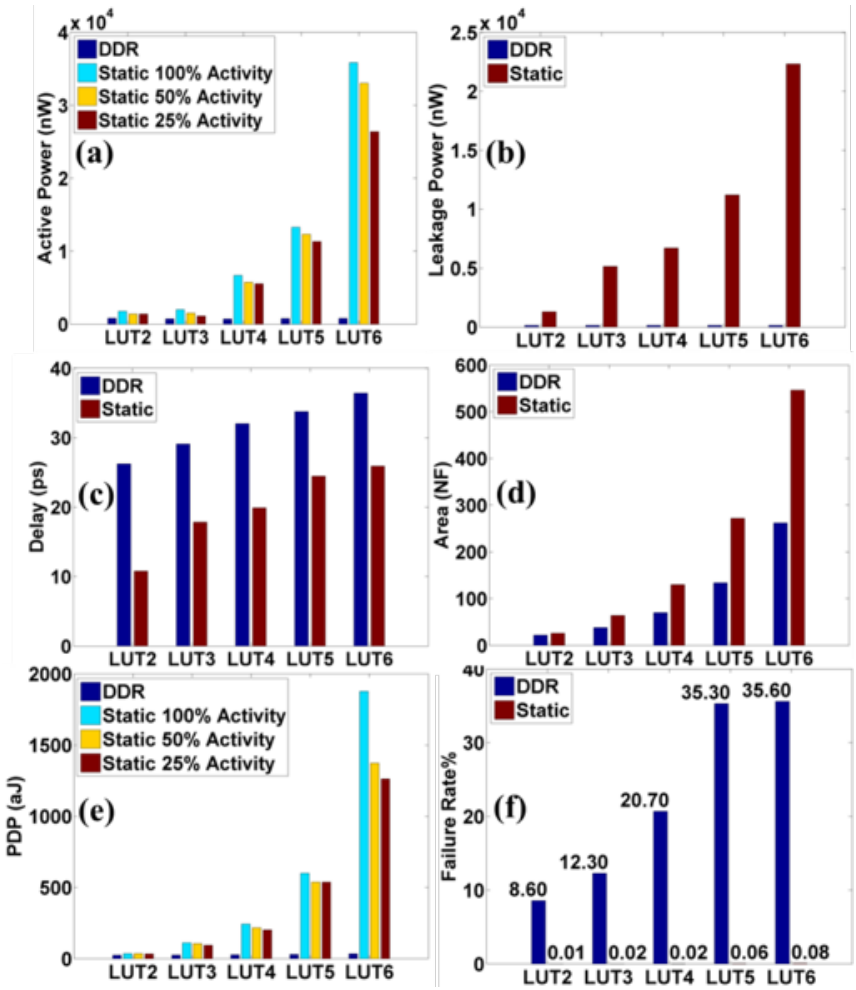

Fig. 3. *Performance comparisons: (a) Active Power (b) Leakage Power (c) Delay (d) Area (e) PDP (f) Failure rate*

By introducing design security requirements and introducing reconfigurability in the early stages of design, our novel security-driven STT-CMOS design flow effectively resists the design reverse engineering attacks. With a circuit consisting of missing gates, an untrusted foundry is not able to overproduce the design as each design finally should be configured by the design house or authorized vendors. Furthermore, selection and replacement of CMOS gates are such that it makes it impossible to determine missing gates in any reasonable time.

5.1 Algorithms

We propose two methods to select the CMOS gates in a netlist to replace with STT-based LUTs counterpart: *independent selection* and *dependent selection.*

5.1.1 Independent Selection . In this method, gates are selected such that they are not connected to each other (directly or indirectly) through any design path. From the security perspective, using the circuit netlist with reconfigurable units and an available configured counterpart, an attacker

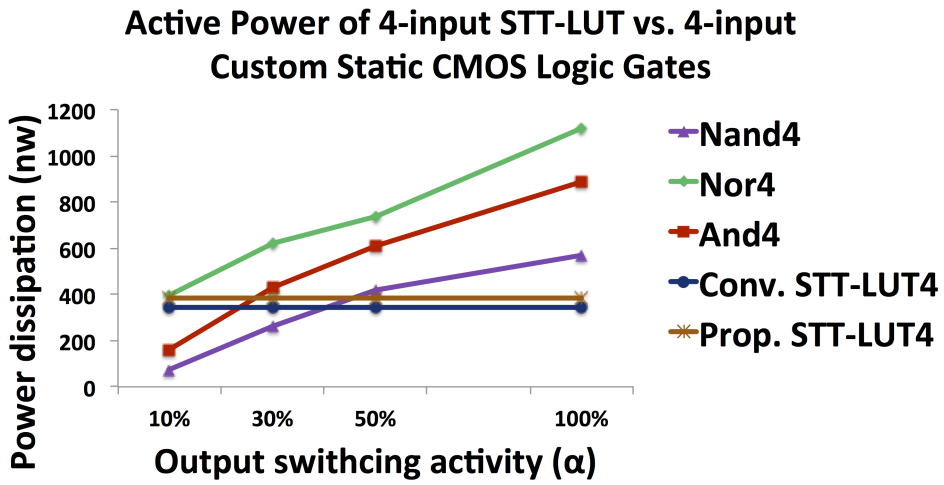

Fig. 4. *The performance of STT LUTs based on number of inputs*

can use a testing technique to justify and propagate the output of missing gates to the observation points. With this effort, the attacker can develop a partial truth table for each missing gate and then guess the functionality of the missing gate.

The independent selection provides some level of security; however, an attacker with adequate resources would be able to break and determine the functionality of STT-based LUTs. Assuming M the number of missing gates, and D the depth of circuit defined as the maximum number of flip-flops on a path from a primary input to a primary output in a circuit, the maximum possible number of required test clocks to determine all missing gates in the independent selection N_{indep} is equal to

$$N_{indep} = \sum_{i=1}^{M} \alpha_i \times D_i \tag{1}$$

where α is the number of required patterns to determine an independent missing gate. The value of α is determined based on the similarity of the output of the gates. For example, the similarity of 2-input AND gate and 2-input NOR gate is 2 since for two input combinations they produce the same output, and the similarity of 2-input AND gate and 2-input NAND gate is 0 as their output are completely opposite. If we assume all missing gates are 2-input gates, the average similarity of gates is 1.45, so the average required patterns to determine a 2-input missing gate (α) is 2.45. For 3-input gates and 4-input gates, α is equal to 4.2 and 7.4, respectively.

5.1.2 Dependent Selection. The second method is the dependent selection where there is dependency between reconfigurable units. Reconfigurable units are selected so that they are reachable from each other; in general, some inputs of a missing gate are driven by some output of other missing gates. This method significantly increases the efforts to determine the functionality of missing gates.

Algorithm 1 presents the implementation of dependent selection. In dependent selection, it first obtains the list of the longest I/O path that is between a primary input to a primary output and

ALGORITHM 1: Implementation of *dependent selection* algorithm.

Input: A gate-level netlist
Output: A hybrid netlist
Select the longest I/O path containing only non-critical timing paths;
foreach *timing path on the I/O path* **do**
    replace all gates with STT-based LUTs;
end
return *Hybrid netlist*;

contains timing paths beginning at and ending to flip-flops. Then all gates on composing timing paths are replaced with reconfigurable units.

Assuming M the number of missing gates that the input of one missing gate is driven by the output of another missing gate, D_i the depth of missing gate i is the number of flip-flops between the missing gate and a primary output, and P_i is the number of possible gates for the missing gate i, the maximum number of required test clocks to determine all missing gates in the independent selection N_{dep}, on average, is equal to

$$N_{dep} = \prod_{i=1}^{M} \alpha_i \times P_i \times D_i \tag{2}$$

where α_i is the number of required patterns to determine an independent missing gate. As discussed earlier, $\alpha = 2.45$ and $P = 2.5$ for 2-input missing gates.

5.1.3 The Parametric-aware Dependent Selection. In the dependent selection, all gates on selected timing paths are replaced with reconfigurable units. It is plausible that replacing all gates violates timing requirement of original circuit. Therefore, we introduce the parametric-aware dependent selection method to minimize the impact and possibly avoid violating timing requirement. The concept, shown in Algorithm 2, the parametric-aware dependent selection method selects only one gate on a timing path and replaces it with its reconfigurable counterparts. All source and sink gates into the timing path are also replaced with their reconfigurable counterparts.

ALGORITHM 2: Implementation of *parametric-aware dependent selection* algorithm.

Input: A gate-level netlist
Output: A hybrid netlist
Select the longest I/O path containing only non-critical timing paths;
foreach *timing path on the I/O path* **do**
    L1: Randomly select one gates and replace it with STT-based LUTs;
    Check design timing constrains;
    if *violated* **then** go to **L1** ;
    Push not-selected gates into UCL;
end
foreach *gate in UCL* **do**
    Replace immediate gates on the timing path driving or being derived by the gate but not belong to the current or longest I/O path;
end
return *Hybrid netlist*;

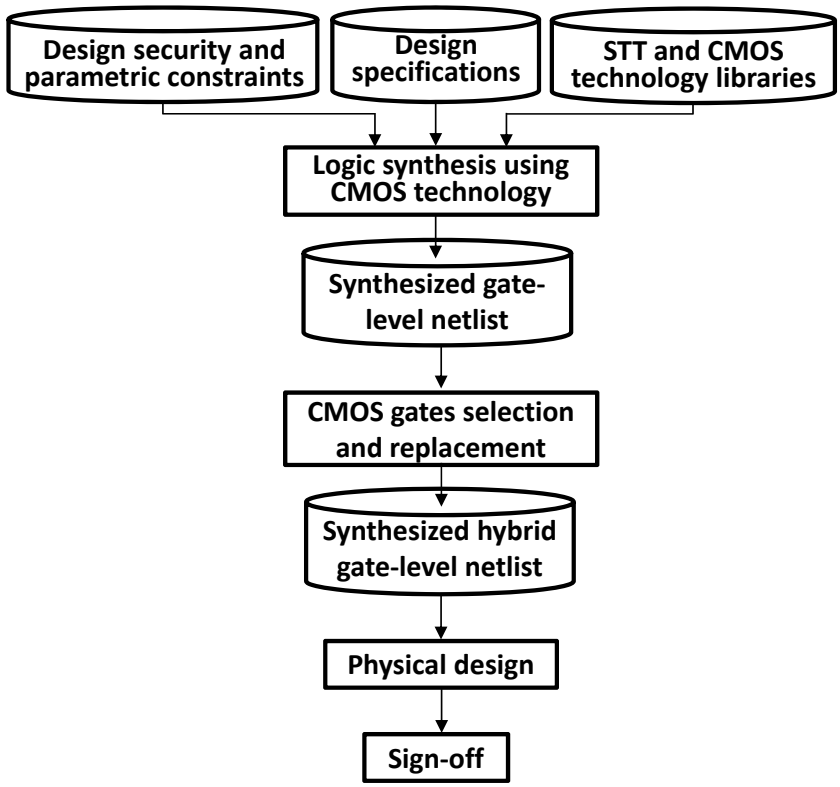

Fig. 5. Our novel STT-CMOS design flow.

Compared with the dependent selection method, the parametric-aware selection method indeed significantly increases the efforts to determine missing gates as it would make it impossible to create partial truth tables for missing gates. As a result, a more plausible approach for the attacker is to launch a brute force attack. Considering M the number of missing gates, I inputs accessible that drive missing gates, P the number of possible gates for each missing gate, and D the depth of circuit, the number of required clock cycles to determine the missing gates in a brute-force attack (N_{bf}) is equal to

$$N_{bf} = 2^I \times P^M \times D. \tag{3}$$

Equation 3 shows the exponential relationship between the number of required clock cycles and the number of missing gates and the number of inputs driving missing gates.

Selecting gates to be replaced with STT-based LUTs while meeting design security requirements and design constraints can be very challenging considering the huge number of timing paths in large circuits. To overcome this issue, first, we construct a graph representation of all of the components of the synthesized gate-level CMOS netlist. Using this graph representation, we randomly select a sample of 2% of the components within the circuit and perform a depth-first search in the graph to find the path to a primary input and a primary output of the circuit containing at least two flip-flops. Once all of the unique paths have been collected, we remove any paths that contain the critical path and sort the remaining paths by depth (e.g., the number of flip-flops between the primary

input and primary output). For independent selection, we select a pre-determined number of nodes for STT out of all nodes on the chosen paths. For dependent selection, we select a random timing path from flip-flop to flip-flop for a random path identified above. For parametric-aware selection, we randomly select a pre-determined number of timing paths and select a pre-determined number of random nodes within that timing path and then continue on the parametric-aware selection algorithm.

In addition to brute-force attack, a hybrid STT-CMOS circuit may undergo machine learning attacks similar to [29]. Contrary to similar works such as camouflaging [31], the possible candidates per STT-based LUT is not limited to a small number of gates. A 2-input STT-based LUT can realize 6 meaningful 2-input gates consisting of AND, NAND, OR, NOR, XOR, XNOR gates. 3-/4-input STT-based LUTs can also implement more than 12 meaningful gates. To exacerbate the situation for machine learning attacks, a 4-input STT-based LUT and a 3-input STT-based LUT can be also used to implement 3-/2-input gates and 2-input gates, respectively, with connecting unused inputs of STT-based LUTs to some signals in the circuit to expand search space for machine learning attacks. Furthermore, we can realize complex functions, such as $(A.(B \oplus C)) + D$, using a STT-based LUT instead of implementing only one simple gate. With incorporating these measures, the machine learning attack would render ineffective to determine the missing gates in any reasonable time as the size of search space is significantly large even with inserting a moderate number STT-based LUTs. While work, such as [29], significantly accounts on accessibility to scan architecture to reduce attack time, it is a common practice that the scan architecture is disabled or locked before releasing the design to raise bar against different attacks such as secret key extraction [21] [46].

5.2 Resistance to Attacks

One of the primary techniques to protect design against reverse-engineering attacks depends on making it more difficult to identify a given gate in the circuit. In fact, the techniques presented in this work aim to produce just this result. In general, if one were to partially extract the netlist and attempt to identify the remainder of the netlist using truth tables and the input/output of the circuit, it would require 1,000 years of processing time. In 2015, El MAssad, Garg, and Tripunitara showed that satisfiability (SAT) solvers can be configured to significantly reduce the amount of processing time required to determine the netlist [11]. Their work was able to show that naively replacing gates within the netlist with black boxes (e.g., STT-based LUTs) that were previously thought to require more than 1,000 years of processing time can be identified in minutes using complexity-theoretic characterization of the IC de-camouflaging problem. To analyze the results of our research, we developed an attack model that is an extension of the concepts proposed and developed in [11]. We use this technique against a subset of the circuits analyzed in this research to determine how it affects using STT-based LUTs.

The reverse engineering approach proposed in [31] suggests the use of two copies of the targeted IC. The first copy is delayered and used to obtain the partial netlist whereas the second copy acts as reference circuit or original circuit. The reference circuit is used to obtain the expected primary output by applying relevant primary input which might be later used for de-camouflaging. Consider the circuit in Figure 6. The two gates G1 and G2 are replaced by non-resolvable camouflaged gates. The gates can assume one of NAND, NOR, XOR gate types, hence there are $3^2 = 9$ possible combinations. Brute Force requires at least 9 input patterns to de-camouflage. On the contrary, it is possible to find the true identity of the gates by applying a minimal set of inputs i.e. three inputs here. This inference comes from the fact that the original circuit gives a unique combination of primary outputs. A true combination of the gate type should be able to match this combination when the same set of inputs are applied. Hence, by applying a particular set of inputs it is possible to eliminate the incorrect gate combinations.

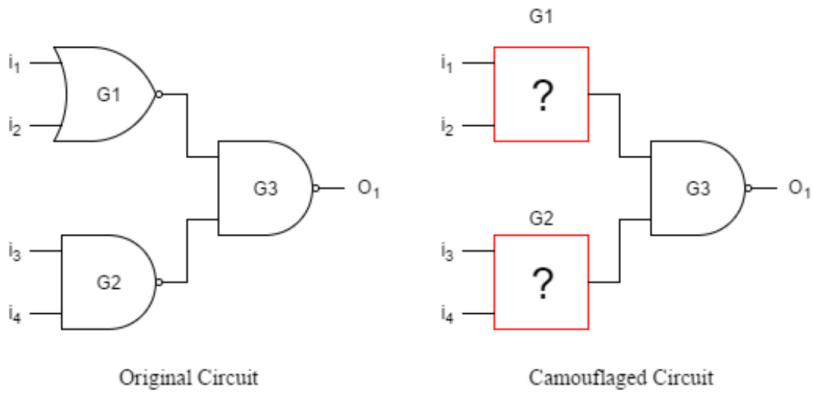

Fig. 6. Circuit depicting non-resolvable camouflaged gates

Table 1. Example of behavior of discriminating set of inputs

(Input, Expected Output) ->	(0000, 0)	(0001, 0)	(0100, 1)
(G1,G2)	**Output for a particular combination of (G1, G2)**		
(XOR, XOR)	1	-	-
(XOR, NAND)	1	-	-
(XOR, NOR)	1	-	-
(NAND, XOR)	1	-	-
(NAND, NAND)	0	0	0
(NAND,NOR)	0	0	1
(NOR, XOR)	1	-	-
(NOR, NAND)	0	1	-
(NOR, NOR)	0	1	-

Table 1 depicts how application of particular set of inputs can narrow down the search for right combination of the gates. As we see, a single input "0000" eliminates 5 incorrect gate combinations (indicated in red). This followed by inputs "0001" and "0100" reveal the true gate combination $(G1, G2) = (NAND, NOR)$. Thus, {fi0000fi, fi0001fi, fi0100fi} is a set of inputs which can de-camouflage the circuit and provide a combination which is functionally equivalent to original circuit. Such a set is referred to as a discriminating set of inputs [11].

Thus, the problem of predicting identity of gates can be split up into parts i.e. (1) Is a discriminating set of inputs able to differentiate among the given gate combinations and de-camouflage a camouflaged circuit C? (2) If so, what is the correct assignment for these gates which match the Boolean functionality of the original circuit? The work in [38] provides formal proofs for categorizing and resolving these decision problems using computational complexity tools.

For our research, we used the techniques introduced by [11] to attack circuits with missing gates chosen by the parametric-aware algorithm described in Secion 5.1.1. To show how these technique affect SAT-based attacks, we used partial circuits from ISCAS '85 [6] and ISCAS '89 [7]. The sequential bench circuits are assumed to have a scan chain flip- flops. This mechanism has been simulated with the use of pseudo-primary inputs and pseudo- primary outputs in place of the

Table 2. Benchmark Characteristics with corresponding % of missing gates

Circuit	Gates	Inputs	Outputs	% of Missing gates		
				1%	5%	10%
s832	292	18	19	2	14	29
s1423	731	17	5	7	36	73

nodes with D flip-flops. Hence, instead of random initialization the D flip-flop inputs are asserted by the Solver as any other primary input.

The gate selection for the partial netlists creation was done at random. The size of the circuits chosen for the security analysis are varied to observe the overall impact. The chosen netlist characteristics and the corresponding percentage and number of missing gates used for partial netlist formation are given in Table 4. The analysis is performed on the two input logic functions i.e. gate types ranging from 3 to 8 with the gate type 3 being equivalent to the camouflaged gates with functionalities *NAND*, *NOR*, *XOR*.

The analysis was performed on a 3.2 GHz quad-core desktop with 8 GB RAM and 500 GB disk space. MiniSAT tool configurations were set at default with resource allocations approximately 2147 TB for memory usage limit and CPU time of around 68 years.

The work was set out to test and analyze the security offered by the current protection techniques against layout level extraction using a SAT-based reverse engineering model. This model had presented the ability to resolve a camouflaged netlist with three gate types in minutes. The STT-based Lookup Tables (LUT) with their programmable nature significantly increase the number of functions that can be tried for any single missing gate making it very difficult to reverse engineer. This work analyzed the security mechanism for two-input missing gates which could have a maximum functionality of 16 gate types with each gate chosen randomly. The trends observed with the metrics selected for testing the vulnerability against reverse engineering i.e. the number of inputs required and time taken to identify the missing gates show significant increase in security compared to the camouflaged gates.

As shown in Figure 7, the claims by SAT-based reverse engineering model also becomes questionable with LUT based missing gates as we increase the input size of missing gates thereby delivering larger functionalities, size of the circuit being reverse engineered and the use of gate selection algorithms described earlier in this section.

6 RESULTS

To evaluate the effectiveness of logical reconfigurability against the design reverse-engineering attack, our proposed security-driven hybrid STT-CMOS design flow is applied to several ISCAS ‘89 [7] benchmarks. The benchmarks are first synthesized in 90nm technology node using Synopsys’s Design Compiler. While the STT technology library based on Suzuki [40] is provided, the CMOS gate selection and replacement is performed and the circuit power and performance parameters are evaluated up on any gate replacement. While all evaluation in this paper is performed in the gate selection and replacement step, the flow continues with the physical design to obtain the circuit layout. Finally, the design is signed off for fabrication.

By introducing security measures in the early stages of design flow, it is possible to effectively meet both design security requirements and parametric constraints. Table 3 shows the impact on performance, power, and area after introducing STT-based LUT units to the selected benchmarks. The second, third, and forth columns of the table present the relative performance degradation

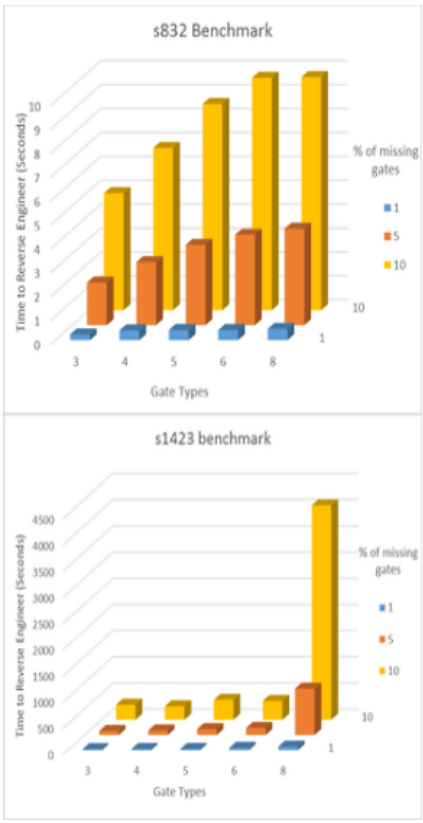

Fig. 7. Time to Reverse Engineer partial netlists when increasing the potential number of gates

after deploying the independent, dependent, and parametric-aware selection, respectively, on the original circuits.

To analyze the netlist and to determine the gates to be selected as STT LUTs, we developed a Python script that analyzes the netlist and creates a graph of the VLSI circuit with each gate and wire represented by nodes showing with inputs acting as the starting nodes of the graph and the outputs acting as the end nodes of the graph. To follow the algorithms described in Section 5, the Python script had to identify the longest path in the circuit as well as a number of input-to-output paths to meet the requirements of the algorithms. Originally, using a brute-force method, this was unsustainable and resulted in run-times of hours for even relatively small circuits with only 20,000 gates–and inherently infeasible in practice. To address this, we improved the methodology to use Synopsys Design Compiler to identify the input-to-output paths and critical path of the circuit. The Python script leverages the outputs of Design Compiler and determines which gates will be replaced with STT-based LUTs.

To achieve this end, as shown in Figure 8, the following steps are performed:

(1) Retrieve the ISCAS ‘89 benchmark
(2) Generate the netlist using Synopsys Design Compiler
(3) For each wire in the netlist, generate the Design Compiler command to calculate the longest path through the wire
(4) Run Design Compiler for each generated script

(5) Collect the input-output-paths and sort them
(6) Using the timing paths calculated, generate a graph of the netlist
(7) Based on the graph, randomly select LUTs not in the critical path
(8) Calculate the critical path, power and area using the graph

As a precursor to this script, a database is created by creating a one-gate netlist for each gate in the algorithm's netlist. The timing information, power and area for that one-gate netlist is used to generate approximate results using the graph.

The graph is created using a Python script that parses the netlist and traverses various nodes in the graph to identify appropriate gates to replace with LUTs.

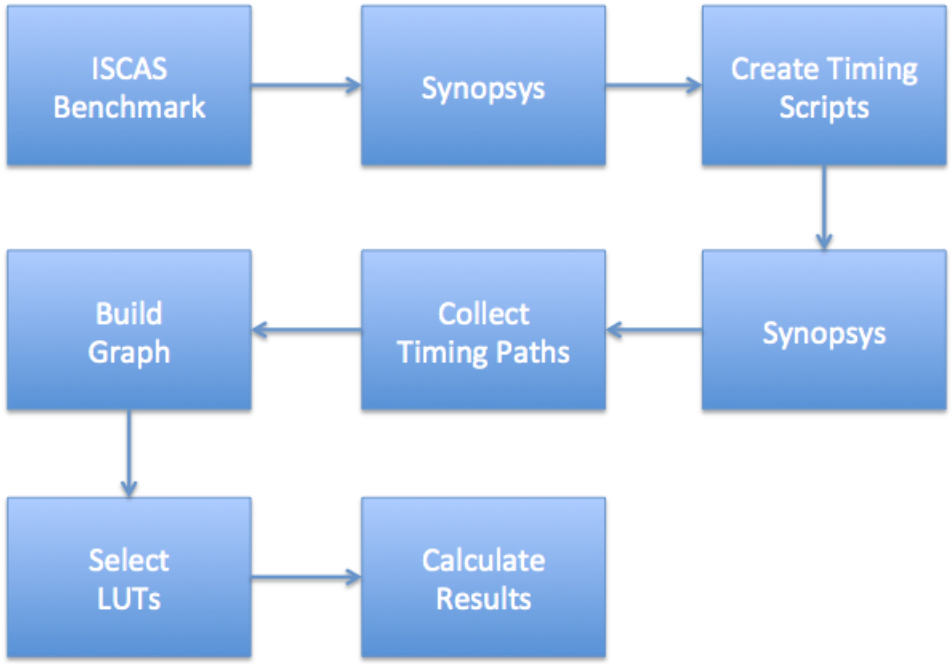

Fig. 8. *The LUT selection process*

For step 3, we generated a Python script that analyzes the netlist produced by Synopsys to collect the individual wires and generates a sequence of Design Compiler commands to perform the worst-case timing analysis through each wire and create a report file.

For step 5, we generated a Python script that analyzes the report files for each wire and generates an input-to-output path and timing information for each wire's worst-case path: this is passed to the script in Step 6 so that the script generating the graph has information for all input-to-output paths collected by Design Compiler, including the critical path. This also ensures that the solution is now deterministic.

Using this information, we are also able to determine the effect that changing a given percentage of nodes for a circuit will have on the overhead as a whole.

Table 3 shows the effect of replacing all gates in a given circuit with STT-based LUTs. As shown, the performance overhead associated with replacing gates with LUTs in a circuit has a maximum–albeit extremely expensive at 100%–cost. This gives us a maximum limit to which the security

Table 3. Effect of all-LUT implementation on each circuit

Circuit	Performance Overhead	Power Overhead	Area Overhead
s641	215%	371%	241%
s820	186%	578%	534%
s832	191%	572%	520%
s953	199%	447%	314%
s1196	168%	521%	243%
s1238	167%	526%	250%
s1488	166%	629%	359%
s5378a	88%	298%	597%
s9234a	94%	291%	1085%
s13207	92%	235%	240%
s15850a	102%	284%	178%
s38584	116%	328%	391%

metrics shown in Figure 9 will apply–as replacing *all* of the gates in a given circuit will render the implementation immune to reverse-engineering attacks. However, the power and area overhead in these cases will be infeasible. As s9234a shows, even a 10X area overhead is possible–particularly if the gates being replaced by STT-based LUTs are multi-output gates (e.g., adders) as these must be implemented with two LUTs, doubling the power and area overhead for that gate.

While the circuit sizes rages from about 300 to 20,000 gates, the results indicate that among the three selection algorithms, the dependent selection has considerable impact on design performance in terms of the delay of the longest path. This is attributed to replacing all gates of timing paths on selected I/O paths with STT-based LUT equivalent. The performance degradation is less or none using independent and parametric-aware selections as all STT-based LUTs are not placed on a single I/O path. Furthermore, with increasing the size of the circuit, both algorithms are provided a larger pool of gates and timing paths; therefore, STT-based LUTs are fairly distributed and a very few STT-based LUTs are located on a single timing path. The results in Table 4 signify that the relative performance degradation is almost zero for several benchmark circuits. The results imply that for large industrial circuits the impact of STT-based LUT units on circuit performance using the independent and parametric-aware dependent selections will be negligible.

In Table 4, we also present the relative power overhead and the number of replaced gates after applying the three selection techniques. For the independent selection, we always randomly select 5 gates for replacement. With increasing the size of the circuits, more number of gates are generally chosen for replacement in the dependent and parametric selection. On the other hand, the power overhead considerably reduces when the size of the circuit increases. For example, s641 benchmark only consists of 287 gates and 5, 4, and 18 gates are replaced with STT-based LUTs in independent, dependent, and parametric selections, respectively. Due to the small size of the circuit, the power overhead is relatively high, i.e. 9.4%, 5.6%, and 22.98% for independent, dependent, and parametric selections, respectively. On the opposite, s38584 benchmark consists of 19,253 gates, and 5, 5, and 26 gates are replaced with STT-based LUTs in independent, dependent, and parametric selections, respectively. While there is a considerable increase in the number of replaced gates, these incur only a small power overhead, 0.03%, 0.02%, and 0.1% for independent, dependent, and parametric selections.

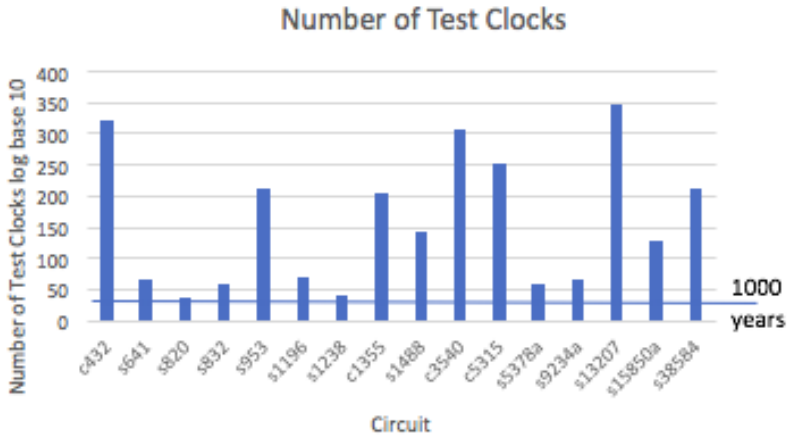

Fig. 9. Required clocks to determine the missing gates

Table 4. The effects of STT-based LUTs

	Performance degredation %			Power overhead %			Area overhead %			Number of STTs			
Circuit	Indep	Dep	Para	Indep	Dep	Para	Indep	Dep	Para	Indep	Dep	para	size
c432	47.6	52.7	51.7	7.3	4.1	16.1	20.3	41.7	0.0	5.0	6.0	24.0	104.0
s641	0.0	0.0	0.0	9.4	5.6	22.9	12.6	12.9	0.0	5.0	4.0	18.0	287.0
s820	36.0	78.3	0.0	2.9	0.3	6.0	11.2	18.9	0.0	5.0	5.0	10.0	289.0
s832	0.0	0.0	0.0	2.6	1.7	6.9	11.7	6.5	0.0	5.0	2.0	18.0	379.0
s953	0.0	0.0	0.0	3.1	1.9	9.0	8.3	6.0	0.0	5.0	3.0	32.0	395.0
s1196	3.2	0.0	0.0	1.3	0.3	4.6	6.7	1.4	0.0	5.0	1.0	21.0	508.0
s1238	17.1	0.0	0.0	2.2	1.0	0.9	7.3	5.9	0.0	5.0	5.0	10.0	529.0
c1355	22.5	89.4	36.3	-2.1	-1.6	1.1	7.4	18.7	0.0	5.0	11.0	45.0	218.0
s1488	0.0	0.0	0.0	0.8	0.2	8.1	8.4	3.4	0.0	5.0	2.0	46.0	657.0
c3540	61.5	70.4	51.2	-1.0	-1.3	1.1	4.0	9.0	0.0	5.0	11.0	22.0	444.0
c5315	7.9	80.4	28.3	-1.0	-1.4	-2.8	3.4	8.1	0.0	5.0	9.0	29.0	503.0
s5378a	0.0	0.0	0.0	0.6	0.0	-0.4	1.5	0.6	0.0	5.0	1.0	15.0	2779.0
s9234a	0.0	0.0	0.0	-0.3	0.2	-1.0	2.2	4.8	0.0	5.0	7.0	17.0	5597.0
s13207	8.4	121.4	6.4	-0.3	0.5	-0.6	0.6	2.3	0.0	5.0	13.0	39.0	7951.0
s15850a	17.5	0.0	0.0	-0.3	0.0	-0.01	0.6	0.2	0.0	5.0	2.0	23.0	9772.0
s38584	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.2	0.0	5.0	5.0	26.0	19253.0
Average	13.8	30.8	10.8	1.5	0.7	4.5	6.6	8.8	0.0	5.0	5.4	24.6	3104.0

The last column of Table 4 indicates the number of gates in the circuits excluding the number of flip-flops. Columns 8 to 10 of Table 4 presents the percentage of incurred area overhead. The results clearly indicate that the area overhead significantly reduces with increasing the size of the circuit. Collectively analyzing results in Table 4 reveals that with increasing the size of the circuit, it is possible to insert more number of STT-based LUTs with no or very negligible impact on performance, power, and area. Figure 9 shows the number of possible required test clocks to determine the missing gates using the machine learning attacks. The results signify that even for small circuits the number of required test clocks for the parametric-aware selection is significantly high so that it would take more than 1000 years assuming one billion pattern application per second to correctly resolve a hybrid STT-CMOS circuit using modern testing equipment. Figure 9 only shows the estimated time to brute-force the solution for sequential benchmarks, as the formula uses the number of flip-flops in a given path as an input.

With the new Synopsys-based analysis, the CPU time for each gate selection is less than the amount of time it takes to run Synopsys for the circuit. Based on this, it can be concluded that

selecting gates for replacement in large industrial circuits can be performed in a small fraction of time.

7 CONCLUSIONS

To prevent design reverse engineering, we introduced a novel security-driven hybrid STT-CMOS design flow that is very low cost and add almost minimal/no change to the standard VLSI IC design flow, making it a suitable solution for IoT market where cost is a prime concern. The flow does completely match the current in-practice industry standard design flow, using industry standard tools like Synopsys Design Compiler to aid in the selection of gates to replace with LUTS, and makes it possible to introduce security measure in the early stage of circuit design. With introducing three novel selection and replacement algorithms, i.e. independent, dependent, and parametric-aware dependent selections, a selected number of CMOS gates from a synthesized gate-level netlist are replaced with reconfigurable non-volatile STT-based LUTs counterparts based on the required security demands and design parametric constraints. Results on standard benchmarks showed significant resiliency of hybrid STT-CMOS circuits against the reverse engineering attack. Further, we showed that applying these techniques to relatively small combinational circuits renders satisfiability attacks, as described in recent work, computationally infeasible. Meanwhile, the impact of STT-based LUTs on design parametric constraints including area, power, and performance has shown to be negligible for large circuits. Furthermore, it has shown that the proposed methods are computationally inexpensive where selecting CMOS gates for replacement takes less than a minute even for large circuits.

REFERENCES

[1] Tigist Abera, N Asokan, Lucas Davi, Farinaz Koushanfar, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. 2016. Things, trouble, trust: on building trust in IoT systems. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 121.

[2] B. Alex and et al. 2010. Preventing IC privacy using reconfigurable logic barriers. *(IEEE Design & Test of Computers)*. 66–75.

[3] R. Anderson. 2008. Security Engineering: A Guide to Building Dependable Distributed Systems, Physical Tamper Resistance. (2008).

[4] Adarsh Reddy Ashammagari, Hamid Mahmoodi, and Houman Homayoun. 2014. Exploiting STT-NV technology for reconfigurable, high performance, low power, and low temperature functional unit design. In *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 335.

[5] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers* 27, 1 (2010).

[6] Franc Brglez, David Bryan, and Krzysztof Kozminski. 1985. Neutral netlist of ten conmbinational benchmark circuits and a target translator in FORTRAN. In *Special session on ATPG and fault simulatio, Proc. IEEE Int. Symp. Circuits and Systems*.

[7] Franc Brglez, David Bryan, and Krzysztof Kozminski. 1989. Combinational profiles of sequential benchmark circuits. In *Circuits and Systems (IEEE International Symposium)*.

[8] Rajat Subhra Chakraborty and Swarup Bhunia. 2009. HARPOON: an obfuscation-based SoC design methodology for hardware protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 10 (2009), 1493–1502.

[9] DARPA. 2014. Supply chain hardware integrity for electronics defense (SHIELD). Microsystems Technology Office/MTO Broad Agency Announcement Defense Advanced Research Projects Agency (DARPA).

[10] Jaya Dofe, Qiaoyan Yu, Hailang Wang, and Emre Salman. 2016. Hardware security threats and potential countermeasures in emerging 3D ICs. In *Great Lakes Symposium on VLSI, 2016 International*. IEEE, 69–74.

[11] Mohamed El Massad, Siddarth Garg, and Mahesh Tripunitara. 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes *(NDSS Symposium)*.

[12] Mohamed El Massad, Siddharth Garg, and Mahesh V Tripunitara. 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.

[13] D. Evans. 2011. he Internet of Things How the Next Evolution of the Internet Is Changing Everything. In *CISCO White Paper*.

[14] S. Giordano and et al. 2013. Thermal effects in magnetoelectric memories with stress-mediated switching. *J of Applied Physics* 46, 32 (2013).

[15] U. Guin and et al. 2014. Counterfeit Integrated Circuits: Detection, Avoidance, and the Challenges Ahead. Journal of Electronic Testing.

[16] X. Guo and et al. Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing. *ISCA 2015* (????).

[17] Benjamin Hill, Robert Karmazin, Carlos Tadeo Ortega Otero, Jonathan Tse, and Rajit Manohar. 2013. A split-foundry asynchronous FPGA. In *Custom Integrated Circuits Conference (CICC), 2013 IEEE*. IEEE, 1–4.

[18] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V Tripunitara. Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation.

[19] S. Koley and P. Ghosal. 2015. Addressing Hardware Security Challenges in Internet of Things: Recent Trends and Possible Solutions. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*.

[20] S. Lam and et al. Exploiting FPGA-aware merging of custom instructions for runtime reconfiguration. *IEEE ReCoSoC 2012* (????), 1–8.

[21] J. Lee and et al. A Low-Cost Solution for Protecting IPs Against Scan-Based Side-Channel Attacks *(VTS '06)*. 94–99.

[22] J. M. Lewis and et al. 2012. Self-modifying FPGA for anti-tamper applications. (April 17 2012). US Patent 8,159,259.

[23] B. Liu and B. Wang. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *DATE 2014*.

[24] Bao Liu and Brandon Wang. 2014. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 243.

[25] Bao Liu and Brandon Wang. 2015. Reconfiguration-based VLSI design for security. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 5, 1 (2015), 98–108.

[26] Duo Liu, Cunxi Yu, Xiangyu Zhang, and Daniel Holcomb. 2016. Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 433–438.

[27] Hamid Mahmoodi, Sridevi Srinivasan Lakshmipuram, Manish Arora, Yashar Asgarieh, Houman Homayoun, Bill Lin, and Dean M Tullsen. 2014. Resistive computation: A critique. *IEEE Computer Architecture Letters* 13, 2 (2014), 89–92.

[28] A. Makosiej and et al. CMOS SRAM scaling limits under optimum stability constraints. *IEEE ISCAS 2013* (????), 1460–1463.

[29] M. E. Massad and et al. 2015. Integrated Circuit (IC) Decamouflaging: Reverse Engineering Camouflaged ICs within Minutes.

[30] Carlos Tadeo Ortega Otero, Jonathan Tse, Robert Karmazin, Benjamin Hill, and Rajit Manohar. 2015. Automatic obfuscated cell layout for trusted split-foundry design. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 56–61.

[31] J. Rajendran and et al. Security analysis of integrated circuit camouflaging. In *ACM CCS 2013*. 709–720.

[32] Jeyavijayan Rajendran, Youngok Pino, Ozgur Sinanoglu, and Ramesh Karri. 2012. Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 83–89.

[33] M. Rasquinha and et al. An energy efficient cache design using Spin Torque Transfer (STT) RAM. *ACM/IEEE ISLPED 2010* (????), 389–394.

[34] F. Ren and D. Markovic. 2010. True energy-performance analysis of the MTJ-based logic-in-memory architecture (1-bit full adder). *IEEE T-ED* 57, 5 (2010).

[35] M. Rostami and et al. Hardware security: Threat models and metrics. In *ICCAD 2013*. 819–823.

[36] J.P. Baukus R.P. Cocchi, L.W. Chow and B.J. Wang. 2012. Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing. US Patent, App. 13/370,118.

[37] Davood Shahrjerdi, Jeyavijayan Rajendran, Siddharth Garg, Farinaz Koushanfar, and Ramesh Karri. 2014. Shielding and securing integrated circuits with sensors. In *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*. IEEE, 170–174.

[38] Gaurav Shenoy. 2016. Implementation and Evaluation of SAT-based Attacks on Hybrid STT-CMOS Circuits for Reverse Engineering. In *Master of Science Thesis, George Mason University, Fairfax Virginia*.

[39] C. Smullen and et al. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *IEEE HPCA 2011*. 50–61.

[40] D. Suzuki and et. al. Fabrication of a nonvolatile lookup-table circuit chip using agneto/semiconductor-hybrid structure for an immediate-power-up field programmable gate array. *Symposium on VLSI 2009* (????).

[41] M. Tehranipoor and et al. 2014. Hardware Trojans and Counterfeit Detection.

[42] Theodore Winograd, Hassan Salmani, Hamid Mahmoodi, Kris Gaj, and Houman Homayoun. 2016. Hybrid stt-cmos

designs for reverse-engineering prevention. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 88.
[43] Jacob Wurm, Khoa Hoang, Orlando Arias, Ahmad-Reza Sadeghi, and Yier Jin. 2016. Security analysis on consumer and industrial iot devices. In *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 519–524.
[44] Kan Xiao, Domenic Forte, and Mark Mohammed Tehranipoor. 2015. Efficient and secure split manufacturing via obfuscated built-in self-authentication. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 14–19.
[45] Teng Xu, James B Wendt, and Miodrag Potkonjak. 2014. Security of IoT systems: Design challenges and opportunities. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 417–423.
[46] B. Yang and et al. Secure Scan: A Design-for-Test Architecture for Crypto Chips. *IEEE TCAD 2006* 25, 10 (????), 2287–2293.
[47] Kun Yang, Domenic Forte, and Mark M Tehranipoor. 2015. Protecting endpoint devices in IoT supply chain. In *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 351–356.
[48] L. Yuan and et al. 2006. VLSI Design IP Protection: Solutions, New Challenges, and Opportunities. 469–476.
[49] W. Zhao and et al. Spin transfer torque (STT)-MRAM–based runtime reconfiguration FPGA circuit. *ACM TECS 2009* 9, 2 (????), 14.

March 2009June 2009

Received July 2007; revised February 2007