Hybrid STT-CMOS Designs for Reverse-engineering Prevention

Theodore Winograd
George Mason University
twinogra@gmu.edu

Hassan Salmani
Howard University
hassan.salmani@howard.edu

Hamid Mahmoodi
San Francisco State University
mahmoodi@sfsu.edu

Kris Gaj
George Mason University
kgaj@gmu.edu

Houman Homayoun
George Mason University
hhomayou@gmu.edu

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 of less than 3%, 8%, and 3% on design parametric constraints including performance, power and area, respectively.

I. Introduction

Integrated Circuits (ICs) are at the core of any modern computing system deployed in various industry sectors such as financial, information technology, smart electric power grids, and aerospace & defense; their security and trustworthiness ground the security of the entire system. However, the security and trustworthiness of ICs are exacerbated by the modern globalized, horizontal semiconductor business model. 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. 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 realize design assurance, we leverage the concept of circuit design using reconfigurable logic based on hardware reconfiguration and transformation which recently proposed in [1, 8] and employ highly promising Spin Transfer Torque (STT) magnetic technology to build robust and reverse-engineering resilient Look-Up-Table (LUT) logic components. 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

DAC '16, June 05-09, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4236-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2897937.2898099

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.

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.

The remainder of this paper is organized as follows. Section II presents preliminaries and surveys related work. In Section III, we explain STT technology. Next, we describe how to employ CMOS-STT technology to realize design for assurance in Section IV. Experimental results are presented in Section V. Finally, Section VI concludes this paper.

II. 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 [17], or extracting IPs used in a circuit and making profits by selling them without knowledge of IP owner [19], or overproducing the design and sell in the black market [5]. Many techniques to counter these attacks have been proposed and many are in active use. Examples of such countermeasures are mislabeling [14], shielding [2], obfuscation [3], camouflaging [12], and self-modification (reconfiguration) [7]. 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. Neither of these countermeasures require the detection of an attack, i.e., they are always on.

STT and Reconfigurable Logic for Security: Hardware reconfigurability has been around for several years, primarily in the form of FPGAs. In [8], 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 [9]. In addition, STT-based LUT has advantage over CMOS-based LUT in terms of performance and area [16, 9]. 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 [16, 9], therefore compared to CMOS-based LUT, it is more robust against power-based side channel attacks.

III. Design of STT-Based LUT

STT technology provides i) approximately $4X$ higher integration density than conventional Static Random Access Memory (SRAM) [20], ii) high retention times (even more than 10 years [15]), iii) high endurance (10^{16} writes, or 10 years of operation as L1 cache) [4], iv) near-zero leakage [13] 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 amazing opportunity to design reconfigurable logics that are on-die, comparable in performance to custom CMOS logic, and have low reconfiguration overhead. 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 [10]. 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. In this paper we use the STT-based LUT design proposed by Suzuki [16] and further improved in a recent work by Mahmoodi [9]. 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.

Figure 1 shows the simulation results of the STT-based LUT and static custom CMOS circuit styles for logic gates of various complexity implemented in a predictive 32nm technology. All the results are normalized to the corresponding results for a static CMOS implementation. It is clear from the results that for small logic gates, the STT-based 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-based LUT style uses less number of PMOS transistors, its benefit is more noticeable for implementation of such logic gates.

Another observation from Figure 1 is that the LUT style shows less power overhead for higher data activity (α). This is due to the dynamic nature of the STT-based 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

Gate	Metric	MTJ Based LUT	Static CMOS
NAND2	Delay	6.46	1
	Active Power(α=10%)	90.35	1
	Active Power(α=30%)	30.12	1
	Standby Power	0.48	1
	Energy per Switching	58.36	1
NAND4	Delay	4.49	1
	Active Power(α=10%)	76.73	1
	Active Power(α=30%)	25.57	1
	Standby Power	0.96	1
	Energy per Switching	34.45	1
NOR2	Delay	4.85	1
	Active Power (α=10%)	80.2	1
	Active Power(α=30%)	26.73	1
	Standby Power	0.51	1
	Energy per Switching	38.89	1
NOR4	Delay	3.06	1
	Active Power(α=10%)	24.25	1
	Active Power(α=30%)	8.08	1
	Standby Power	1.06	1
	Energy per Switching	7.42	1
XOR2	Delay	4.95	1
	Active Power(α=10%)	22.45	1
	Active Power(α=30%)	7.48	1
	Standby Power	0.13	1
	Energy per Switching	11.11	1
XOR4	Delay	4.18	1
	Active Power(α=10%)	90.06	1
	Active Power(α=30%)	30.02	1
	Standby Power	0.04	1
	Energy per Switching	37.64	1

Fig. 1. Comparison of circuit style alternatives (α: output switching activity).

STT-based 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-based LUT only depends on its fain-in (number of inputs). The leakage power of the STT-based 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-based LUT style implementation offers less leakage.

IV. Security and STT Technology

Figure 2 presents our novel security-driven design flow. 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 section IV-A is chosen by the designer. The selected algorithm takes the synthesized gate-level netlist and carefully select a number of CMOS gates to 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 physical design, and then the design is signed-off.

By introducing design security requirements and reconfigurability in the early stages of design, our novel security-driven

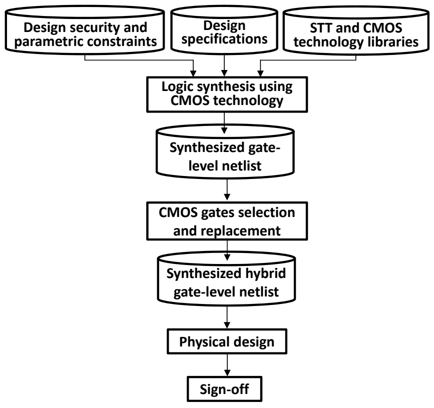
Design security and parametric constraints
Design specifications
STT and CMOS technology libraries
Logic synthesis using CMOS technology
Synthesized gate-level netlist
CMOS gates selection and replacement
Synthesized hybrid gate-level netlist
Physical design
Sign-off

Fig. 2. Our novel security-driven hybrid STT-CMOS design flow.

hybrid STT-CMOS design flow effectively resist 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 so that it makes it impossible to determine missing gates in any reasonable time.

A. 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.*

A.1 Independent Selection

In this method, gates are randomly selected such that they may or may 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 can use a testing technique to justify and propagate the output of missing gates to some observation points. With this effort, the attacker can develop a partial or complete truth table for each missing gate and then guess the functionality of those 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 using testing techniques to justify their inputs and propagate their outputs to some observation points. 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

$$N_{indep} = \sum_{i=1}^{M} \alpha_i \times D_i \quad (1)$$

where α is the average 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. For 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.

```
Input: A gate-level netlist
Output: A hybrid netlist
The list of the longest I/O paths ($lst$) containing only
  non-critical timing paths;
foreach timing path on lst do
    replace all gates with STT-based LUTs;
end
return Hybrid netlist;
```

Algorithm 1: Implementation of *dependent selection* algorithm.

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

Assuming M is the number of missing gates that the input of one missing gate is driven by the output of another missing gate, D_i is 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 dependent 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. For example, $\alpha = 2.45$ and $P = 2.5$ for 2-input missing gates.

A.3 Parametric-aware Dependent Selection

In the dependent selection, all gates on selected timing paths are replaced with reconfigurable units. It is possible 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 parametric-aware dependent selection method selects only a few number of gates on a timing path and replaces them with their reconfigurable counterparts. As untouched gates make determining missing gates possible, all gates driving and driven by the untouched gates that are not on the timing path are also replaced with reconfigurable units.

Algorithm 2 presents the implementation of parametric-aware dependent selection algorithm. Similar to the dependent selection algorithm, it first obtains the list of the longest I/O paths. It then replaces some of gates with STT-based LUTs. To increase the effort for reverse engineering, only gates with two or more number of inputs are considered for replacement. After replacement, design timing information is updated and compared against design timing constraints. If there is a violation, selection and replacement are repeated. Any unselected gate on a targeted timing path, is saved in a unselected list (USL). After this step, any gate that drives or is driven by any gate in USL is replaced with a STT-based LUT.

Compared with the dependent selection method, parametric-aware method indeed significantly increases the efforts to de-

```
Input: A gate-level netlist
Output: A hybrid netlist
The list of the longest I/O paths ($lst$) containing only
  non-critical timing paths;
foreach timing path on lst do
    L1: Randomly select some gates with two or more
      inputs and replace with STT-based LUTs;
    Check design timing constrains;
    if violated then go to L1 ;
    Push unselected gates into USL;
end
foreach gate in USL do
    Replace immediate gates driving or being derived by
      the gate but not belong to the longest I/O path;
end
return Hybrid netlist;
```

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

termine 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 or even a machine learning attack. Considering M is the number of missing gates, I is inputs accessible that drive missing gates, P is the number of possible gates for each missing gate, and D is 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 [11]. Contrary to similar works such as camouflaging [12], 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 [11], significantly accounts on accessibility to scan architecture to reduce attack time, it is a common practice that the scan architecture is disable or locked before releasing the design to raise bar against different attacks such as secret key extraction [6] [18].

V. 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 benchmarks. The benchmarks are first synthesized in 90nm technology node using Synopsys's Design Compiler. While the STT technology library based on Suzuki [16] 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 I 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 after deploying the independent, dependent, and parametric-aware selection, respectively, on the original circuits.

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 I signify that the relative performance degradation is almost zero for larger circuits. The results imply that for large industrial circuits the impact of STT-based LUT units on circuit performance using the independent and dependent selections will be negligible. It should be noted that as the selection of timing paths and gates is performed randomly, we observe that there is slightly larger overhead for a larger circuit in some cases where a slightly smaller circuit incurs smaller overhead, like between s1288 and s1488 benchmarks or s1238 and s1196 bechmarks. However, the trend clearly indicates that larger circuits result in smaller overhead.

In Table I, we also present the relative power overhead and

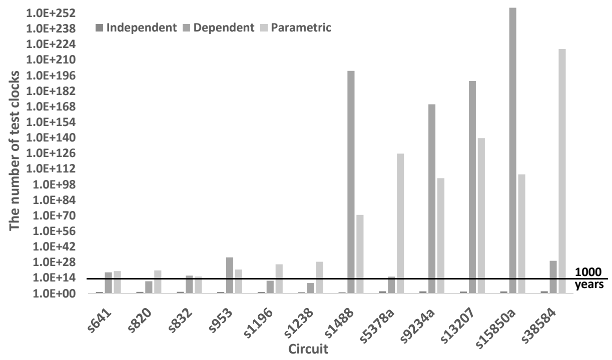

Fig. 3. The number of possible required test clocks to determine the functionality of missing gates.

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 is considerably reduces when the size of the circuit increases. For example, s641 benchmark only consists of 287 gates and 5, 39, and 9 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. 11.14%, 82.11%, and 8.45% for independent, dependent, and parametric selections, respectively. On the opposite, s38584 benchmark consists of 19,253 gates, and 5, 47, and 166 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, 0.21%, 1.86%, and 5.13% for independent, dependent, and parametric selections.

The last column of Table I indicates the number of gates in the circuits excluding the number of flip-flops. Columns 8 to 10 of Table I 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 I 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 3 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. For example, the analysis of s38584 benchmark shows that with introducing only 166 STT-based LUTs using the parametric-aware selection technique, about 6.07E+219 test clocks are required to determine their functionality while there is only about 5.13% increase in power consumption, 1.56% increase in area, and 0% performance degradation.

Table II presents the CPU time for selecting gates for replacement in independent, dependent, and parametric selections. The results obtained on a 1.7 GHz Intel Core i7 with 8 GB of RAM. As results show, it only takes about 44 seconds to select 166 gates for parametric-aware selection in s38584 benchmark with about 20,000 gates. It can be concluded that

TABLE I
The percentage of power, performance and area overhead after introducing STT-based LUT units.

	Performance degradation %			Power overhead %			Area overhead %			Number of STTs			
Circuit	Indep	Dep	Para	Indep	Dep	Para	Indep	Dep	Para	Indep	Dep	Para	size
s641	0	2	1	11.14	82.11	8.45	2.64	20.66	4.98	5	39	9	287
s820	10.82	14.77	2.37	11.45	18.72	5.08	3.02	5.63	1.34	5	9	2	289
s832	4.42	71.20	7.75	13.44	14.39	1.92	3.22	4.98	0.51	5	8	1	379
s953	0	28.42	4.55	11.02	33.49	8.03	2.32	7.14	2.38	5	15	5	395
s1196	0	0	0	7.83	12.54	7.95	1.97	3.94	2.64	5	10	7	508
s1238	0	8.76	4.45	8.32	14.39	8.13	2.02	4.38	2.73	5	11	7	529
s1488	0	45.45	6.7	4.43	15.49	8.18	1.60	6.83	3.47	5	21	11	657
s5378a	7.3	82.32	1.5	2.93	45.11	9.80	0.37	9.30	6.88	5	131	98	2779
s9234a	7.7	62.42	0	1.20	42.18	9.83	0.20	10.06	3.24	5	256	82	5597
s13207	2.07	0	0	0.73	9.82	8.21	0.12	2.19	2.60	5	92	111	7951
s15850a	0	25.39	0	0.70	9.41	6.04	0.10	1.88	1.78	5	89	85	9772
s38584	0	0	0	0.21	1.86	5.13	0.05	0.44	1.56	5	47	166	19253
Average	2.69	28.40	2.36	6.12	24.96	7.23	1.47	6.45	2.84	5.00	60.67	48.67	4033.00

TABLE II
The CPU time (min:sec)for selecting gates for replacement in various selection algorithms.

Circuit	Independent	Dependent	Parametric
s641	00:00.7	00:01.0	00:00.8
s820	00:00.1	00:00.1	00:00.1
s832	00:00.1	00:00.1	00:00.1
s953	00:00.1	00:00.2	00:00.2
s1196	00:00.1	00:00.2	00:00.2
s1238	00:00.1	00:00.1	00:00.1
s1488	00:00.1	00:00.1	00:00.1
s5378a	00:09.1	00:14.9	00:26.9
s9234a	01:15.5	01:07.4	01:30.2
s13207	00:25.4	00:25.4	00:27.1
s15850a	00:52.6	00:48.2	00:54.9
s38584	00:35.7	00:42.3	00:44.0

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

VI. Conclusions

To prevent design reverse engineering, we introduced a novel security-driven hybrid STT-CMOS design flow. The flow does completely match the current in-practice industry standard design flow 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. 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.

Reference

[1] B. Alex and et al. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers*, (1):66–75, 2010.

[2] R. Anderson. Security engineering: a guide to building dependable distributed systems, physical tamper resistance. 2008.

[3] R. Chakraborty and et al. HARPOON: an obfuscation-based SoC design methodology for hardware protection. *TCAD*, '09.

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

[5] U. Guin and et al. Counterfeit integrated circuits: detection, avoidance, and the challenges ahead. Journal of Electronic Testing, 2014.

[6] J. Lee and et al. A low-cost solution for protecting IPs against scan-based side-channel attacks. VTS '06, pages 94–99.

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

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

[9] H. Mahmoodi and et al. Resistive computation: A critique. *IEEE Computer Architecture Letters 2014*, 13(2):89–92.

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

[11] M. E. Massad and et al. Integrated circuit (IC) decamouflaging: reverse engineering camouflaged ICs within minutes. 2015.

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

[13] M. Rasquinha and et al. An energy efficient cache design using spin torque transfer (STT) RAM. *ACM/IEEE ISLPED 2010*, pages 389–394.

[14] M. Rostami and et al. Hardware security: threat models and metrics. In *ICCAD 2013*, pages 819–823.

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

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

[17] M. Tehranipoor and et al. Hardware Trojans and counterfeit detection. 2014.

[18] B. Yang and et al. Secure scan: a design-for-test architecture for crypto chips. *IEEE TCAD 2006*, 25(10):2287–2293.

[19] L. Yuan and et al. VLSI design IP protection: solutions, new challenges, and opportunities. pages 469–476, 2006.

[20] W. Zhao and et al. Spin transfer torque (STT)-MRAM–based runtime reconfiguration FPGA circuit. *ACM TECS 2009*, 9(2):14.