electronics

MDPI

Article

Logic Locking Using Hybrid CMOS and Emerging SiNW FETs

Qutaiba Alasad *, Jiann-Shuin Yuan and Yu Bi

Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA; Jiann-Shiun.Yuan@ucf.edu (J.-S.Y.); yubi@knights.ucf.edu (Y.B.)

* Correspondence: qutaibaeng@knights.ucf.edu; Tel.: +1-407-334-3964

Received: 10 July 2017; Accepted: 16 September 2017; Published: 20 September 2017

Abstract: The outsourcing of integrated circuit (IC) fabrication services to overseas manufacturing foundry has raised security and privacy concerns with regard to intellectual property (IP) protection as well as the integrity maintenance of the fabricated chips. One way to protect ICs from malicious attacks is to encrypt and obfuscate the IP design by incorporating additional key gates, namely logic encryption or logic locking. The state-of-the-art logic encryption techniques certainly incur considerable performance overhead upon the genuine IP design. The focus of this paper is to leverage the unique property of emerging transistor technology on reducing the performance overhead as well as preserving the robustness of logic locking technique. We design the polymorphic logic gate using silicon nanowire field effect transistors (SiNW FETs) to replace the conventional Exclusive-OR (XOR)-based logic cone. We then evaluate the proposed technique based on security metric and performance overhead.

Keywords: emerging technology; hardware security; logic locking; security metrics

1. Introduction

One of the main security challenges nowadays is the fact that the outsourcing of the chip industrialization and the consolidation of different third-party intellectual property (3PIP) due to the globalization of integrated circuits (ICs) design manufacture have made it simpler for unauthorized/untrusted users to compromise the integrity of once trusted IC processes [1,2]. Among all security threats, hardware Trojan attacks and IC piracy are the two most severe security concerns that the US government is encountering after more and more domestic IC companies started to go fabless. Following the booming of the merchant foundry industry, domestic IC design houses are able to access advanced-process capacity without the need for huge capital expenditure of constructing semiconductor foundries (note that a prediction of the cost of developing a semiconductor foundry is over $5.0 billion in 2015 [3]). At the same time, the reduced fabrication cost sacrifices the design security and leaves all IC designs in the hands of foundry. The International Chamber of Commerce (ICC) stated in their 2011 report that the total global economic and social impacts of counterfeiting and pirated products are as much as $650 billion every year. The figure more than doubled to $1.8 trillion in 2015 [4].

Although researchers try to solve this dilemma by developing new circuit structures, limited by the underlying complementary metal oxide semiconductor (CMOS) technology, the goal of achieving high security while still preserving low power consumption becomes a difficult task.

CMOS has been dominant technology on hardware security when it comes to design of circuit key generation, prevention of Differential Power Analysis (DPA) attacks, and/or hardware implementation

of encryption chip. However, there is a trade-off between security level and cost overhead. Interestingly, advancement of emerging technologies enables researchers to overcome the constraints of Moore's law by employing the unique features of emerging devices, such as spintronic devices (All Spin Logic (ASL) and Domain Wall Motion (DWM)), Tunnel Field-Effect Transistor (TFET), magnetoresistive random-access memory (MRAM), and silicon nanowire (SiNW). Therefore, that raises a question, "Can emerging devices help obtain higher performance with lower area"?

A state-of-the-art solution for logic obfuscation objectives is to leverage CMOS technology, but the challenge is to obtain a high level of chip protection without a high cost penalty. The required performance overhead for logic encryption purposes can exceed 25% when the number of inserted key-gates (XOR/XNOR) is about 5% of the total number of gates in the combinational International Symposium on Circuits and Systems (ISCAS)-85 benchmark [5]. In order to address this issue, we propose a technique based on the new characteristic of emerging technology for IP protection and hardware attack prevention. More specifically, we introduce the silicon nanowire (SiNW) FET based polymorphic logic gate to help obfuscate the netlist to further improve IP protections. Different from previous efforts [6–8], this paper presents an in-depth theoretical analysis and security evaluation with the proposed technique. The details of our contributions are listed as follows:

- We first present the polymorphic logic gate based on emerging SiNW polarity-contrallable FET and its advantages over conventional CMOS technology.
- We then incorporate polymorphic logic gates for encrypting combinational circuits. A polymorphic gate based logic encryption algorithm is further proposed with theoretical analysis.
- We evaluate the proposed SiNW FETs and CMOS hybrid logic encryption, achieving a hamming distance of 50% for most of the ISCAS'85 benchmark circuits.
- The performance penalty of the proposed technique has also been evaluated, where a much smaller overhead is incurred compared to the previous literature. A genuine energy-efficient logic locking is achieved.

The paper is structured as follows: Section 2 gives an overview of SiNW FET, where Device modeling is discussed. Conventional logic encryption methods are also included. Section 3 provides the concept of SiNW based polymorphic logic gate design and presents the detailed performance comparisons. In Section 4, SiNW based polymorphic logic gates are used to encrypt the integrated circuit. Theoretical analysis of proposed techniques are also included. The experimental results are illustrated in Section 5, which consists of both security evaluation and performance penalty. We conclude with Sections 6 and 7, which respectively represent a summary discussion and plans for future work with SiNW based logic encryption.

2. Background

In this section, we briefly review the device technologies (representative terminology), the problem that we aim to solve and the related work on the topic. In addition, we summarize the state-of-the-art work on logic encryption.

2.1. Introduction to Silicon NanoWire FET

One of the issues we should discuss first is the phenomenon of ambipolarity, which is defined as the placement of both positive and negative charge carriers under bias constraints. It enables a designer to change the polarity of the device. A good example of leveraging ambipolarity is found in silicon nanowire [9], graphene [10], and carbon nanotubes [11], which have already been fabricated [12,13]. Schottky barriers allow device functionality to be changed based on the external signal values. Among all of the above-mentioned devices, we concentrate on a vertically-stacked silicon nanowire FET, which includes two Gate-All-Around (GAA) electrodes [13].

The three-dimensional structure of Vertically-stacked GAA SiNWs is demonstrated in Figure 1. The benefit of using this structure is that it enhances electrostatic regulation. This device has two gates, namely control and polarity gates. In general, the transistor can be switched on and off based on the value of the supplied voltage at the control gate, while the polarity gate is used to swap the n and p channels, which is located between the Drain and Source junctions [13,14]. There are several emerging devices that have polarity control features, such as carbon nanotube CNT FETs, SiNW FETs, nanoelectromechanical (NEM) relays, Graphene SymFET, and so on. This work is focused on SiNW FET because it is compatible with the modern CMOS. It should be noted that designing reconfigurable logic gates are not limited to emerging transistors only, e.g., SiNW FETs, Graphene transistors, or ASL. One can get similar characteristics using only CMOS transistors, but it requires a larger number of transistors as discussed in [15–17].

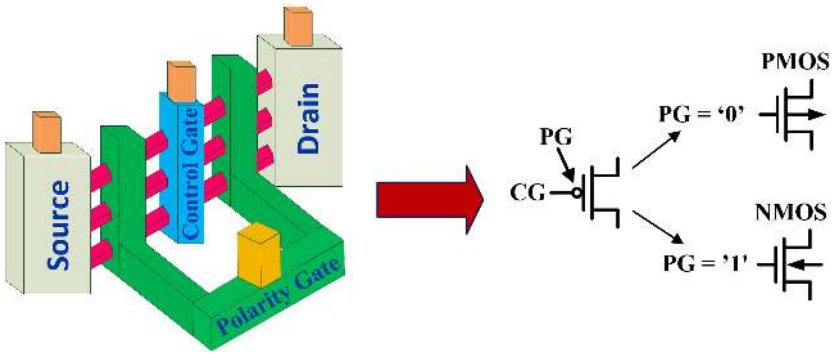

Figure 1. Three-dimensional scheme of the silicon nanowire field effect transistors (SiNW FETs) with the characteristics of two separate gates, namely, the control gate (CG) and polarity gate (PG) to form either a p-channel metal-oxide-semiconductor (PMOS) or a n-channel metal-oxide-semiconductor (NMOS) field effect transistor.

2.2. Logic Encryption Technique

Figure 2 explains the design flow of an IC through the designing, testing, and fabricating processes including a logic locking approach [18]. Since ICs might be imitated in an untrusted foundry due to the offshoring for fabrication process, they can be encrypted using a logic locking method with low cost. When chips come back after manufacturing, their correct functionalities can be revealed only via providing their valid keys.

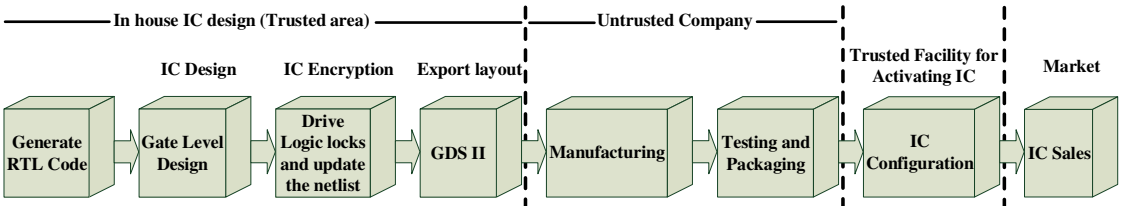

Figure 2. An integrated circuit design flow with logic encryption technique.

Logic encryption techniques are essentially able to prevent the untrusted overseas foundries from reverse engineering, injecting hardware Trojan and tampering with IP privacy. Even though an attacker in an untrusted foundry can get the layout and reverse engineer an IC, logic encryption can prohibit such an attack from obtaining the original design by encrypting the most important parts in the chip. Untrusted foundries cannot benefit from imitating ICs because they have been locked by designers who are the only ones that know the correct keys [18].

2.3. Prior Works

An IC could be protected from serious attacks by using either combinational or sequential logic technique. In combinational logic locking, XOR/XNOR key gates are introduced to mask the correct functionality of IP design [18–20]. Roy et al. [19] proposed a chip-locking system for active IC metering, while targeted to make physical tampering infeasible. The chip-locking framework inserts XOR/XNOR key gates with fan-ins connected to the bits of keys that activate the circuit. The insertion is achieved at randomly selected locations before physical synthesis but after logic synthesis. Similarly, Baumgarten et al. [21] used lookup table-based locking units that hinder attempts to reverse-engineer functionality from the mask prospectives. It demonstrates how logic encryption can be propagated to the field programmable gate array (FPGA) domain. Rajendran et al. [20] attempts to insert the key gates in a way that maximizes the relationship between correct and corrupt output patterns once wrong keys are applied. The fault analysis-based logic encryption formalizes the fault impact of a given netlist and incorporates XOR/XNOR gates at the selected locations. A wrong key ensures the corrupting of output values. A continuing work [18] includes a multiplexer as logic cone for the encryption. In [22], multiplexers are inserted in two ways, at the half of the output-bits and at each output bit, respectively, in order to assert 50% Hamming distance between the correct and incorrect outputs with less performance penalty. Moreover, a linear-feedback shift register (LFSR) random generator has been leveraged to change the output values on applying invalid keys.

The aforementioned techniques are vulnerable to most serious reverse engineering attacks, such as sensitization [5] and Boolean Satisfiability (SAT) based attacks [23]. In general, sensitization attacks use automatic test pattern generation (ATPG) tools to propagate the key-bits to the primary outputs of the encrypted design, while SAT attacks [23] can decrypt the locked circuit and reveal its secret key. A technique [24], namely strong logic locking (SLL) [24], was proposed to prevent propagating key-bits based attacks by inserting each two pairs of key-gates to a gate in the original circuit. They also incorporated Advanced Encryption Standard (AES) cryptography to prevent SAT attacks. In [25], the emerging technique, All Spin-Logic Device (ASLD) has been used to build secure combinational circuits that can prohibit sensitization attacks since ASLD provides a single key-bit for any simple logic gate without any extra hardware resources. Even though the ASLD offers strong protected IC against such attacks, it requires higher power dissipation compared to CMOS technology. In [26], Xie et al. introduced a technique, called Anti SAT, to protect an IC from SAT attacks. The simulation results indicate that the Anti SAT block can exponentially increase the number of attempts that the attacker needs to get the correct key; however, Anti-SAT is vulnerable to Signal Probability Skew (SPS) attacks [27]. In [28], a technique, namely SAT Attack Resistant Logic Locking (SARLock), was presented against SAT based attacks. SAR block is used to corrupt the output of the locked circuit unless the valid key is entered. As a consequence, an attacker's effort of finding the secret key increases exponentially. Unfortunately, the SAR technique is also broken by the development of SAT solver, namely Double Discriminating Input Pattens (DDIP), where the DDIP attack excludes more than one invalid key at each iteration. In [29], a method called tenacious and traceless logic locking technique (TTlock) is used to modify the logic cone circuit in a way to produce incorrect output when an invalid key is applied. To get the correct output of the locked circuit, Restore logic block has been incorporated. More details regarding SAT attacks and countermeasures will be given in Section 6.1.1.

In sequential logic locking, additional states are adopted in the state transition graph [30]. The design will not work correctly unless the correct key-bits sequence is provided on the obfuscated state transition graph. If the key is pulled out, the obfuscated circuit comes back in a logic block state. Rajendran et al. [31] gives a discussion on applying the logic encryption to the micro-architecture level. Process encryption selectively encrypts certain units of microprocessor to strengthen the detection of hardware Trojan attacks.

3. Designing Polymorphic Gates Using SiNW FETs

Polymorphic electronics, which were first introduced in [15], are based on the idea of having multiple functionalities built in the same cell and deciding the input–output relation by means of a controllable factor in the circuit. For instance, a polymorphic gate presented in [15] would be an AND gate when the supply voltage (VDD) is 3.3 V and it functions as an OR gate when VDD is lowered to 1.5 V. Such multi-functional gates would prove useful in a number of applications. Circuits that change functionality with temperature variation can find use in aerospace applications, or those that respond to VDD variation could be used to change functionality when the battery is low. In addition, polymorphic electronics could be beneficial in evolvable, intelligent or self-checking hardware [17]. For security objectives, adding polymorphic gates to a digital circuit can hide the real functionality of the circuit. Since the circuit functions correctly only in a certain configuration of the control signals known to the designer, even if the adversary knows the whole netlist (including the dummy and true contacts), he or she will not be able to utilize the circuit in his or her own design [20]. Carefully encrypting logic in this way can ensure that it will take too long for the adversary to find the key (a vector constructed from all the morphing signals of the polymorphic gates). Therefore, the polymorphic gate becomes a good candidate for integrated circuits protection against IP piracy.

Various polymorphic logic gates using CMOS technology are implemented via leveraging several techniques, such as external signals, different temperatures, and multiple VDD values. Table 1 shows a brief recapitulation of implementing different polymorphic logic gates. In [15], polymorphic logic gates were achieved using a smart algorithm. However, on applying an external signal, the designs encounter a problem through the simulation test, which is producing constant current at the output signal of the polymorphic gates, e.g., NOR/NAND. Moreover, connecting many stages of polymorphic gates in series causes another problem because, in some cases, their inputs might be connected to VDD or ground (GND). A more empirical technique is to use different VDD values, which has been already done [15]. However, employing many VDD values is not a feasible solution, especially with the new scaling technology, where the ranges of VDD are restricted. Designing XOR/NAND polymorphic gate with nine transistors [17] is considered as a good technique for emerging devices.

Table 1. A summary of developed polymorphic gates.

Polymorphic Gates	Techniques	# of Transistors	Publications
XOR/NAND	3.3/0V External signal	9	[17]
NOR/NAND	1.8/3.3V VDD	6	[15]
OR/AND	3.3/1.2V VDD	8	[16]
XOR/AND/OR	1.5/3.3/0.0V External signals	10	[16]
OR/AND	0.0/3.3V External signals	6	[16]
AND/NAND/XOR/NOR	1.8/0.0/1.1/0.9V External signals	11	[16]
OR/AND	125/27 C Temperatures	6	[16]
NOR/NAND	exchanging *key* and $\overline{key}$	4	Our Work

Now, we present our technique to implement different polymorphic gates for IP protection features employing the polarity control signal of the SiNW FET device. SiNW FET is very similar to CMOS except for the addition of the polarity gate between the drain and source junctions. As demonstrated in Figure 3, the structure of both a NAND and a NOR gate is not different in CMOS and SiNW devices. By only swapping the value of the control signal, denoted as $key/\overline{key}$ in Figure 3b,d, a designer can easily exchange the functionality of a gate with the same structure without any other extra resources. More precisely, in Figure 3b, if the $\overline{key}$ value is zero and the key value is one, the logic gate works as a NAND gate, while it works as a NOR gate when the values of $\overline{key}/key$ are interchangeable (see Figure 3d). Note that swapping the VDD and GND signals in any CMOS based

logic produces the complement of the original function at the output. However, full voltage level at the output will not be achieved due to the presence of PMOS in the pull-down network or NMOS in the pull-up network. Consequently, key-bits can be formalized via only gathering the $\overline{key}$ and *key* signals to a wire with an inverter. As a result, ICs can be encrypted by exchanging some logic gates in the original circuits with different polymorphic logic gates with much less area and Power and Delay Product (PDP) penalties, instead of incorporating XOR/XNOR gates or multiplexers as key-gates, which increase the performance overhead extensively as in [18]. Different functionalities with the same structure using CMOS could also be accomplished, but at the penalty of larger number of transistors as mentioned in Table 1.

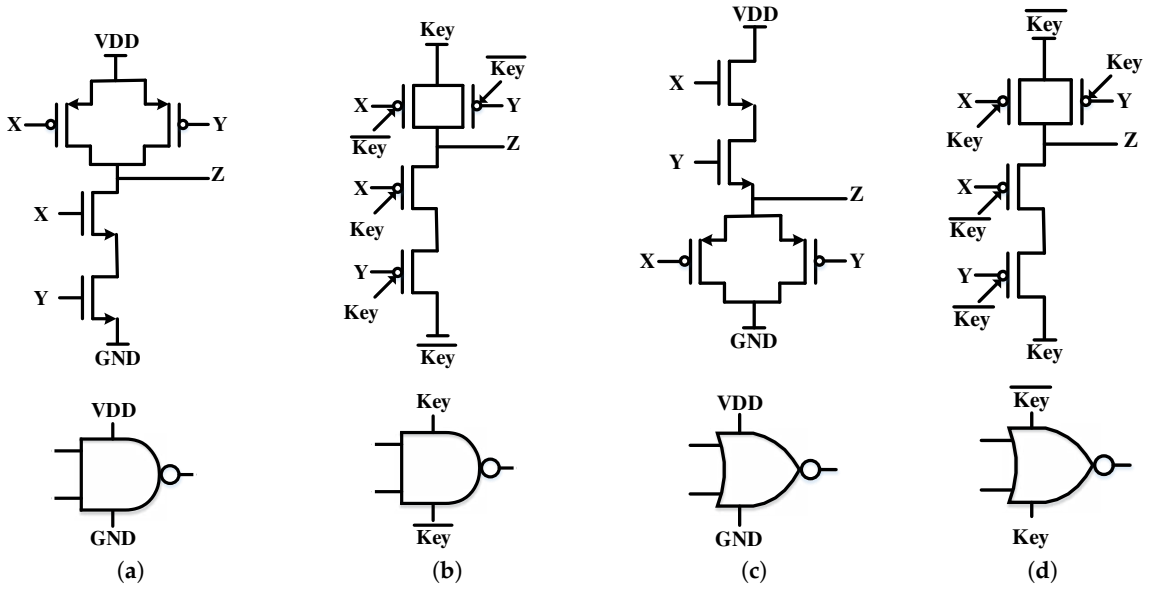

Figure 3. Different logic gates using complementary metal oxide semiconductor (CMOS) and silicon nanowire (SiNW) devices: (**a**) traditional CMOS NAND gate (**b**) silicon nanowire field effect transistors (SiNW FETs) NAND gate (**c**) conventional CMOS NOR gate (**d**) SiNW FETs NOR gate.

4. SiNW in Logic Encryption

A design could be encrypted via inserting different types of key-gates though different locations in an original circuit, such as look up tables (LUTs), multiplexers, XOR/XNOR and AND/OR gates. The locked chip with XOR/XNOR insertion is stronger against the most serious threats [23] than any other types of key-gates. However, building an XOR/XNOR gate requires a higher number of transistors than other gates, such as AND, OR, etc. As a result, the performance overhead will elevate significantly, especially for a small scale circuit (<800 gates) where the power and area overheads might override the original circuit size. For instance, by adding few XOR/XNOR key-gates (less than 5% of the total number of gates in an original circuit), the penalty of the power and area is approximately larger than 31% and 20% for the majority benchmark circuits, respectively [20]. It is worth mentioning that this amount of adding ratio is not enough to prevent the brute force attack, where the key-size should at least be larger than 64 bits [19]. With the scaling of CMOS technology, it becomes more expensive to achieve similar security level by compromising the performance. Due to the defects of existing work, we would like to present our improved method to implement logic locking using emerging transistors.

4.1. Fundamental of Logic Locking

A simple demonstration of logic locking is shown in Figure 4. The original logic gate is two-input AND gate. An exclusive-OR gate is further added to combine the original output f with a locking

enable signal k. Then, the locked netlist consists of two logic gates, AND gate and XOR gate, respectively. The locked Boolean logic function is given in Equation (1).

$$f_{lock} = f \cdot \overline{K} + \bar{f} \cdot K, \; f = ab. \tag{1}$$

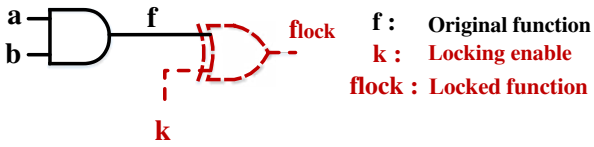

Figure 4. A simple example of logic locking.

When $K = 0$, it functions as the original AND logic gate. Meanwhile, when $K = 1$, it locks the original AND gate and works as a NAND gate. With triggered key ($K = 1$), the output will report all the four input vectors as failing patterns. It is important to note that $K = 1$ is not dedicated to lock the function. For instance, when an XNOR gate is incorporated, the locking key is switched to $K = 0$. The choice of either XOR or XNOR gate relies mainly on the definition of key value, where normally $K = 1$ is more favorable. Furthermore, the key-bit (K) could be configured as one or zero (based on the designer's desirability). For instance, if the inserted key-gate is XOR, K should be set as zero to recover the correct functionality. However, one can configure such key to one for the correct functionality by only adding an inverter before or after the inserted XOR key-gate. Chakraborty et al. [30] introduced a methodology of defining logic cone, in which more logic elements are included so that the number of failing input patterns will increase accordingly. This scenario will not be covered in our work due to the larger area overhead.

4.2. Encrypted Logic Circuit Leveraging Polymorphic Logic Gates

Since inserting key-gates that are designed using traditional CMOS technology for logic encryption purposes leads to a high performance overhead, our technique is to select gates in an original circuit that have a high impact on output and then exchange them with polymorphic logic gates designed using SiNW.

A simple example of obfuscating a circuit using our proposal is shown in Figure 5. The original design has two 2-AND, 2-NAND, and 2-OR gates with five primary input and two primary output signals as demonstrated in Figure 5a. To encrypt this circuit, a designer can replace one OR and one NAND gate with AND/OR and NAND/NOR polymorphic logic gates, respectively, as shown in Figure 5b. The two polymorphic gate keys, referred to as ($K1$) and ($K2$) in Figure 5b, are specified as "00" to recover the correct functionality. For any other $K1$ or/and $K2$ value(s) (incorrect key values), the encrypted design will produce the wrong output. An attacker cannot know what the original gates are before the replacements since the original gates before the exchanged AND/OR polymorphic gates could be either AND or OR gates, and they could be either NAND or NOR gates before the exchanged NAND/NOR polymorphic gates. Note that each of the two incorporated polymorphic gates has an inverter (to create a uniform key-bit as mentioned in Section 3). As a result of using this approach, the performance penalty should be much less than inserting XOR key-gates.

The locked design should produce corrupt outputs for most of the combination incorrect key values. Otherwise, the encryption technique will be vulnerable [20] to an attacker who might figure out the correct functionality. Consider the same encrypted circuit in Figure 5b. On applying input pattern "00110", the correct output of the circuit, which is "10", will be revealed once the correct value "00" of $K1$ and $K2$ is supplied . In contrast, the design will produce incorrect outputs "01" at F1 and F2, respectively, if both $K1$ and $K2$ values are "11", and therefore the Hamming distance between

the correct and corrupt outputs will be 100%. In this case, the first polymorphic gate switches from original OR to AND gate, and the second one switches from NAND to NOR gate. Moreover, if the value of either *K*1 or *K*2 is '1', the output signal of F1 and F2 will be either "00" or "11", respectively, where for each case the Hamming distance will be 50%.

In additional to these two polymorphic gates, another XOR/XNOR polymorphic gate is designed as shown in Figure 5c. The XOR/XNOR polymorphic gate could be swapped to XNOR/XOR gate. Adding more reconfigurable gates is important to increase the ambiguity of an attacker from identifying or comprehending the structure of the original circuit. The three possible aforementioned polymorphic gates have been leveraged for the encryption purposes. The detailed security evaluation will be discussed further in the following section.

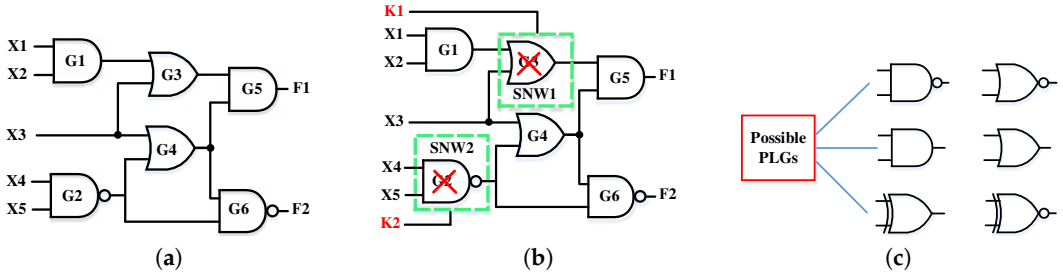

Figure 5. An example of encrypted a circuit using polymorphic logic gates designed using SiNW FETs (**a**) an original circuit (**b**) encrypted circuit via exchanging some gates in the original circuit with polymorphic logic gates, where both of AND/OR and NAND/NOR polymorphic gates are incorporated) (**c**) three possible polymorphic logic gates produce six different logic gates.

4.3. Security Metrics

Before the discussion of the detailed implementation, it is essential to explain the security metrics on evaluating the proposed logic locking technique.

As expected, the attacker is not aware of the key values for encryption and decryption. An extensive test plan might be launched in order to retrieve the correct keys from the attackers' perspective, thereby decrypting the protected IP. Certainly, increasing the key size can increase the effort of an attacker. By applying the wrong key values on the encrypted design, the attacker will get wrong outputs.

To further formalize the security metric, we assume, as the authors in the fault impact analysis assumed [18], that the IC design consists of T primary input bits, Y primary output bits and M encryption key bits. Let $N = \{0, 1\}$. Assume that a valid input $x \in N^T$ and a corresponding correct output $z \in N^Y$. Let $k \in N^M$ be the correct key values. A function f with encryption variables should be defined as two scenarios:

- On employing the valid secret key k, the function produces correct outputs for all input test patterns.
- On employing the incorrect secret key values, the function generates wrong outputs correspondingly:

$$f(x,w) = \begin{cases} z \forall \times \in N^T, z \in N^Y, & \text{when } w = k, \\ z' \forall \times \in N^T, z' \in N^Y, \text{but } z' \neq z, & \text{when } w \neq k. \end{cases} \quad (2)$$

To define the security metrics, Hamming distance (HD) has been commonly adopted. The definition of Hamming distance is a number used to denote the difference between two binary strings. By that means, the wrong output z′ can be quantitatively differentiated from the correct output z by applying HD measurement. For instance, when HD(z, z′) = 0, the corner case shows that the outputs of encrypted netlist function independently of the locking key. It indicates that the applied encryption is drastically

weak. If HD(z, z′) = Y (the number of output bits), z′ is complementary to z, which is also weak in case an attacker tries to reverse the output value [18].

Consequently, it is substantial for the defender to identify the system and define the encryption mechanism such that the attacker is unable to recover the correct functionality. With the minimized correlation between the wrong and the correct outputs, a maximum ambiguity can be generated for the attacker. Let B be the number of output-bit combinations corresponding to certain HD between the correct and wrong outputs. If HD(z, z′) = P, then B is calculated as $\binom{Y}{P}$.

Similar to cryptography, a larger B would imply greater ambiguity, thereby improving the robustness. Clearly, B is maximum when $P = Y/2$ (or HD(z, z′) = $Y/2$). Therefore, the security metric for logic locking/encryption technique should be defined in a way that the Hamming distance is evaluated between the output bits by employing the correct key values and the wrong key values. A Hamming distance of half of the output-bit number (HD = $Y/2$ or 50% of Y) indicates the most robust implementation.

4.4. Algorithm for Insertion of Polymorphic Logic Gates

It is substantial to formalize the previous analysis into a universal method. Algorithm 1 is proposed to choose the optimized locations for incorporating polymorphic logic gates.

Algorithm 1 Logic Locking Algorithm

Input: Netlist, keySize
Output: Encrypted netlist with key
for $i \leftarrow 1$ *to KeySize* **do**
  for *each* $gate_I$ *at the output of the netlist* **do**
    Call the ***GATE*** and update the netlist;
    if *Corrupted Output* $\geq$ (*Threshold*) **then**
      Call the ***CAL_HD***;
    else
      Call the ***GATE***;
    end if
  end for
  for *each* $gate_k \in$ *netlist* **do**
    Call the ***GATE*** and compute the corrupted output;
  end for
  Select the highest impact gate on output;
  if (*KeySize* = *MAX*) or (*Inc HD* = *smallest*) **then**
    Call the ***GATE***;
    Terminate;
  end if
  Call ***CAL_HD***
end for
function ***CAL_HD***:
  Increment i;
  Accumulate the corrupted output;
  if $HD == 50\%$ **then**
    Terminate;
  else if $HD > 50\%$ **then**
    Compare the HD at exchanging $gate_{I-1}$ with $gate_I$;
    Select the exchanging gate that is closer to 50 % HD;
    Terminate;
  end if
end function
GATE: case (gate):

{
$NAND \iff NOR$;
$OR \iff AND$;
$XOR \iff XNOR$; }

In general, the algorithm has two inputs-netlist and keysize, while the output is the locked netlist with inserted key.The algorithm starts with inputting one key bit into an original netlist. Each selected gate close to the output will be calculated regarding certain test patterns. If incorrect output bits are 50% different from correct output bits, i.e., HD = $Y/2$, the algorithm will terminate and output the encrypted netlist. When HD = 50% is not satisfied for gates close to output, the selection will iteratively go over the remaining gates in the netlist and calculate the highest impact (HD = 50%).

Note that two conditions are required, increasing-rate HD $\leq$ 0.01% and KeySize == MAX (MAX is 128 bits in this paper), respectively. When HD is increased by 0.01% every iteration, we will terminate the program. The reason is because HD almost hits the limit, and it merely adds extra overhead by incorporating more encryption key. The for loop continues incrementing the key size until a desired HD is satisfied.

Two functions CAL_HD and GATE are also attached following the abstracted main pseudocode. CAL_HD enables the computation of Hamming distance, while GATE selects the potential exchanging gates. As mentioned, three different polymorphic logic gates are employed, resulting in six various cases.

5. Results

5.1. Experimental Setup

In this section, we provide empirical results regarding the implementation penalty and the security level of the proposed approach. The effectiveness of our proposal has been evaluated using combinational benchmark circuits from ISCAS'85 benchmark suites [32]. We leverage the Synopsys Hailey Simulation Program with Integrated Circuit Emphasis (HSPICE) for the circuit simulation to design and simulate the SiNW based polymorphic logic gates. Afterwards, the Java language is utilized to implement the algorithm of the proposed logic locking technique. One thousand random input patterns are applied to the encrypted netlist to further evaluate the Hamming distance. The Synopsys Design Compiler, including both silicon nanowire 20 nm and CMOS 20 nm technologies, is used to further evaluate the performance overhead of all ISCAS'85 benchmark circuits.

5.2. Security Evaluation

To evaluate the security of logic locking, a Hamming distance based metric is mostly applied in [18,20–22]. Figure 6 shows the Hamming distance analysis of ISCAS'85 benchmark circuits using our proposed algorithm. Approximately 50% Hamming distance is achieved for all benchmark circuits. The slope of the traces implies the effectiveness of logic locking technique. If the slope is steeper, a smaller amount of key gates is required for encryption purposes, thereby reducing the performance overhead.

The majority of benchmark circuits hits the 50% mark in less than 40 key gates, except for one outliner C5315, which needs 95 key gates. Furthermore, as shown in Figure 6, when an encrypted circuit reaches 50%, its Hamming distance value does not swerve more by incorporating more key gates. In other words, the minimum number of key gates for achieving 50% HD is defined as the encryption threshold. The defender can intentionally increase the key gates for extra obfuscation without changing the robustness of the logic locking.

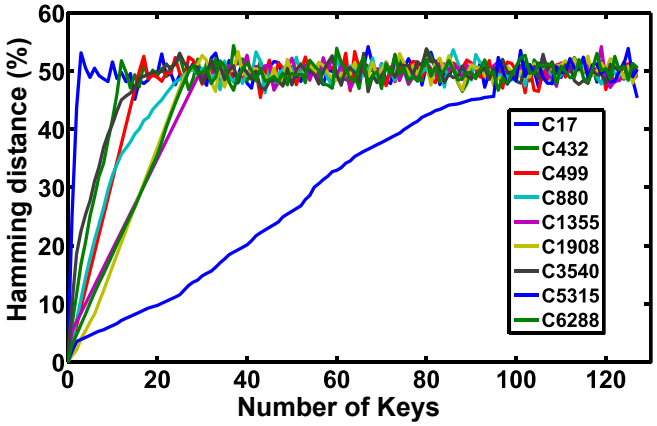

Figure 6. Hamming distance of ISCAS'85 (International Symposium on Circuits and Systems) benchmark circuits.

Table 2 shows the detailed results of security evaluation. The previous random and fault analysis-based logic encryptions are included for the comparison. The number of required key gates using a polymorphic logic gate is listed between the second and fourth columns. The fifth column shows the number of required XOR/XNOR gates used in previous random and fault analysis works. It is apparent that our proposed technique embraces more variants for key gates besides XOR/XNOR gates. NAND/NOR and AND/OR based polymorphic gates virtually are more favorable for most benchmark circuits. It can be seen that the required number of the polymorphic logic gates is less than the conventional XOR/XNOR based key gates, which implies the effectiveness of our proposed technique. The last column of Table 2 shows the achieved Hamming distance using our technique, where 50% HD is mainly accomplished. Only benchmark circuit C5315 with 45.6% HD is better than both random and fault analysis based methods.

Table 2. The number of polymorphic logic gates to achieve 50% Hamming distance using our proposed scheme compared to previous techniques.

Benchmark Circuits	# of XOR PLGs	# of NAND PLGs	# of AND PLGs	# of XOR/XNOR Gates [19,20]	Hamming Distance (%)		
					Ran [19]	FA [20]	PLGs
C17	-	3	-	6	42	51	53
C432	-	10	1	17	29	50	50.06
C499	16	-	-	40	26	50	50
C880	-	18	13	28	19	50	48.3
C1355	-	32	1	42	26	50	50
C1908	-	23	4	28	26	50	49.9
C3540	-	8	13	22	23	50	49.9
C5315	-	4	91	97	15	44	45.6
C6288	-	26	1	27	32	50	50

5.3. Performance Overhead

As we discussed, our proposed polymorphic gates mechanism should display a dramatic advantage in less performance overhead, i.e., area and power-delay product overheads, mainly resulting from the polymorphic gates not adding additional logic gates into original circuits. However, it is expected that our technique would incur certain performance overhead, since SiNW FET is more energy-hungry than its CMOS counterpart due to the unique polarity controllable feature of the emerging device.

Figure 7 shows the area overhead of all benchmark circuits with logic locking technique. The number of logic gates corresponds to the results listed in Table 2. Similar to the previous work [20], we do not include the overhead of peripheral circuits, such as key-bit generator. Apparently, the polymorphic gate based logic locking has drastically lower area consumption than the other two techniques. When the circuit scale increases, the overhead is merely negligible for our proposed technique. C499 circuit has almost zero area overhead, mainly because the key gate is an XOR/XNOR polymorphic gate, which has less area for SiNW FET than for CMOS.

Figure 8 shows the power-delay product (PDP) penalty of all benchmark circuits. It maps to the number of gates added for encryption listed in Table 2. Except for the C499 circuit, all benchmark circuits are more favorable to NAND/NOR and AND/OR polymorphic gates. It is obvious that the polymorphic gate based logic locking hardly provokes any overhead on power-delay product, where <1% overhead applies to every benchmark circuit. On the other hand, random and fault analysis encryption techniques display a considerable power-delay overhead upon original circuits, where >25% penalty occurs at the majority of benchmark circuits.

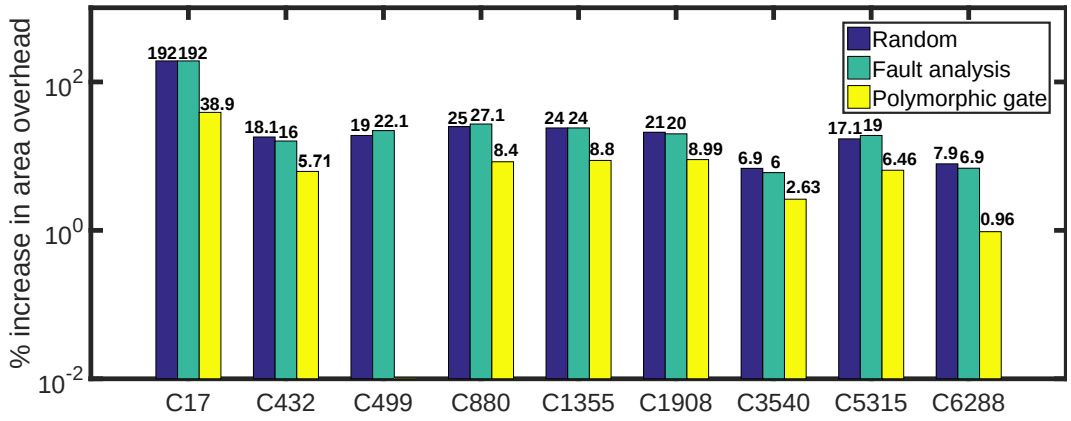

Figure 7. Area overhead of random, fault analysis and polymorphic gate based logic locking.

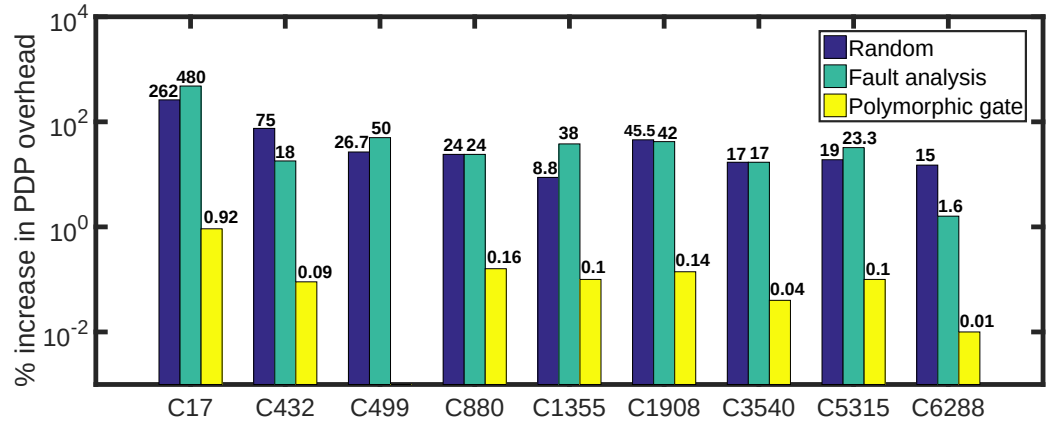

Figure 8. Power-delay product overhead of random, fault analysis and polymorphic gate based logic locking.

6. Discussion

6.1. Attacker's Perspectives

The goal of an attacker is to expose the secret key of an encrypted circuit. Once the key is revealed, there is no meaning for the encryption since with the correct key the attacker can copy an IC, insert

a hardware Trojan, and/or overbuild an IC illegally without designer's license. The most serious attacks and remedies on logic encryption techniques are discussed below.

6.1.1. SAT Based Attack

SAT attack is the most severe one among all existing threats. This attack effectively disputes the secret key from all presented logic encryption methodologies. The attack records all the discriminating input–output patterns of an active IC. Afterwards, the discriminating inputs are applied to the encrypted IC with different key-bits and the corresponding output patterns are compared with the recorded outputs that are obtained from the active IC. After supplying all discriminating inputs, the correct key of the encrypted IC will be abstracted from the SAT formula [23].

All of the current defending techniques against SAT-attack require incorporating an additional circuit. Figure 9 shows the more resilient techniques against SAT-attack that we discuss in detail. Instead of directly connecting all of the key-bits to the key-gates in a locked circuit, some of the key-bits are fed as inputs to 32-bit AES cryptography [24]. Then, the outputs of AES are considered as the actual part of the key-bits, as illustrated in Figure 9a. Inserting such cryptography makes the SAT-attack execution time increase exponentially with the number of fed key-bits. However, the performance penalty will be massive, which is practically infeasible. Subramanyan, et al. [23] emphasized that SAT attack is vulnerable to any design has a structure of "Tree-AND". Since there is no guarantee that the encrypted circuit has a Tree of AND structure, it is possible to incorporate a circuit that has XOR/XNOR key-gates (between part of the valid key and the input-bits) and AND/NAND gates to prohibit SAT attack with smaller overhead compared to AES. Based on this fact, three different techniques have been proposed, namely Anti-SAT [26], SARLock [28], and TTLock [29].

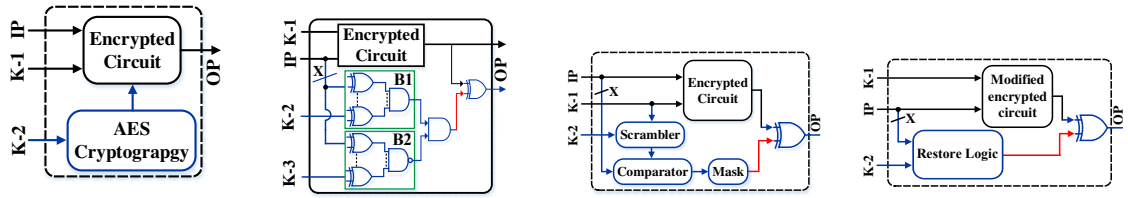

Figure 9. Prohibition Satisfiability (SAT) attack methodologies (specified by a blue color) (**a**) Advanced Encryption Standard (AES) cryptography: conceals part of the correct key-bits (**b**) Anit-SAT: produces incorrect output at a single primary output for all incorrect combination keys unless the correct one is applied (**c**) SAT Attack Resistant Logic Locking (SARLock): inverts the outputs of the encrypted circuit unless the correct key is provided (**d**) tenacious and traceless logic locking (TTLock): modifies the logic cone circuit by flipping its outputs unless the secret key is provided.

A simple example of Anti-SAT resilience is shown Figure 9b. Two complementary blocks (B1,B2) were incorporated. The distinguishing input (X) is connected to XOR gates with key-bits K-2 for B1 and K-3 for B2, where $|K-2| = |K-3|$, and the outputs of each set of the XOR gates are connected to AND and NAND gates, respectively. The outputs of these AND and NAND gates are fed to another AND gate, which is connected with an output-bit in the encrypted circuit to an XOR gate. Anti-SAT is broken by Signal Probability Skew (SPS) attack [27] because the outputs of B1 and B2 are inputs to a gate that has maximum different signal values. SARLock technique is demonstrated in Figure 9c. A bunch of XOR gates between input X and K-2 with AND gate are added, namely comparator. Then, the output of this AND gate with the output of the locked design are connected to XOR gates. The primary outputs of the locked design will always produce wrong values unless the combined correct key (K-1 and K-2) is provided. If an attacker can supply random K-2 equal to an input pattern, the correct functionality will be revealed. Therefore, the authors added a scrambling block to mix K-1 with K-2

and hence prohibit such attacks. Unfortunately, a double DIP based attack [33] successfully breaks SARLock. Double DIP is the development of SAT-attack, which allows for excluding at least two wrong key-bits during each iteration. When the Double DIP completes, the SAT solver returns a key that is the correct logic encryption key ($K1$) plus random SARLock key (K-2). TTLock [29] is the more practical technique. TTLock has an XOR gate and a Restore logic block. The encrypted circuit is modified in a way to produce incorrect functionality for a certain input pattern, which is specified by the designer. The Restore logic circuit is used to correct the functionality only when the valid key is inserted, as shown in Figure 9d. In the last three above mentioned techniques, K-2 must be longer than or equal to 64 bits in order to successfully prevent SAT-attack. As a consequence, the output value of these methodologies (Anti-SAT, SARLock, and TTLock) could be tracked, signified in a red color in Figure 9b–d. This happens because this output signal should be constant for most of the wrong supplied key values. If an attacker removes the tracked signal in both Anti-SAT and SARLock, he or she will get the encrypted circuit alone, which is vulnerable to SAT-attack. However, if the tracked signal in TTLock is removed, the offender will get the modified locked design that is different from the original one. Therefore, TTLock provides strong protection against SAT and tracked signal attacks. A hardware engineer can add the TTLock technique to the proposed logic locking based CMOS-SiNW FETs to obtain a robust logic encryption against all of the existing attacks.

A recent work was proposed [34] to prevent SAT-attack without the need to add a tree structure by creating a logic loop in combinational circuits. The loop requires adding extra dummy gates and wires. Even though this technique is strong against traditional SAT-attacks, it has been efficiently broken by cycle SAT (CycSAT) algorithm based attack using different acyclic constraints [35].

It is worth noting that the two reversed engineering attacks (propagated/isolated secret key [5] and SAT [23] attacks) assume that an attacker can get a functional IC from the market and obtain the encrypted chip by either IC design or reverse engineering in an untrusted foundry [5,23,36]. Therefore, an adversary can have access to the primary input–output pairs and can also reveal the structure of the circuit. By getting the circuit's structure and applying input–output pairs, he or she is able to use one of the above-mentioned techniques to reveal the secret key. To thwart such attacks, a defender can use our proposal + TTLock to get robust IC protection.

6.1.2. Sensitization of the Secret Key-Bits Based Attack

In general, the value of the secret key-bits can be propagated to the primary outputs of the locked circuit via supplying certain input patterns and/or muting some other key-bits unless there is a relationship between the inserted key-gates or the insertion is not done randomly through the original design [5]. This happens due to the fact that each inserted key-gate is either an XOR or an XNOR gate. One of its primary inputs is the key-bit, and the second one comes from an internal net in the circuit. For example, the value of a key-bit will be revealed on a primary output-bit if the second input value of the inserted XOR gate (coming from the internal net of the encrypted netlist) is zero, which can be achieved by a supplying special input pattern. Rajendran et al. [5] proposed a technique, namely Smart logic Obfuscation (SO), to prohibit sensitization attack via maximizing the interference graph among the injected key-gates, where the attacker needs many years to break a locked design that has a sufficient number of key-bits. In our proposed encryption-based polymorphic gates, the attacker may sensitize a path from the output of a polymorphic gate to an output on the working device, and, therefore, the logic function can be determined by applying different patterns to the polymorphic gate's input (using ATPG). Once the logic function is determined, a key bit guess can be made on an unprogrammed device and the same vectors run again. If the output remains the same, then the key bit guess is correct; otherwise, the opposite value must be the correct assignment. Therefore, one needs to employ 'interference graph' to make the task difficult [5].

6.1.3. Applying Brute Force Attacks

An assailant could expose the valid key of an encrypted circuit by applying all possible cases of the key-bits unless the key-size is long enough. A defender can prevent the brute force attacks via increasing the key length. In the XOR/XNOR insertion approaches, enlarging the key-size means increasing the number of the injected key-gates since each new key-bit is an input of each added key-gate leading to an increase in the performance overhead substantially. In our proposal, besides the smaller performance penalty due to the exchanging gates, a hardware engineer can freely enlarge the key-size to a maximum of two times if the key-bits are not gathered to a line with an inverter for each exchanged gate, as demonstrated in Figure 3.

6.2. Key Generations

Previously, Rajendran et al. [20] applied a Physically Unclonable Function (PUF) circuit and Rivest–Shamir–Adleman (RSA) encryption unit to generate the keys for logic encryption. However, area consumption of the two cryptographies might override original netlist with only hundreds of logic gates. For instance, ISCAS'85 C17 to C1355 circuits have less than 400 logic gates. To tackle this issue, we adopted the common encryption technique in system on chip (SoC) design, called dynamic scrambling [37]. The encryption and decryption mechanisms for the key generation are presented in Figure 10.

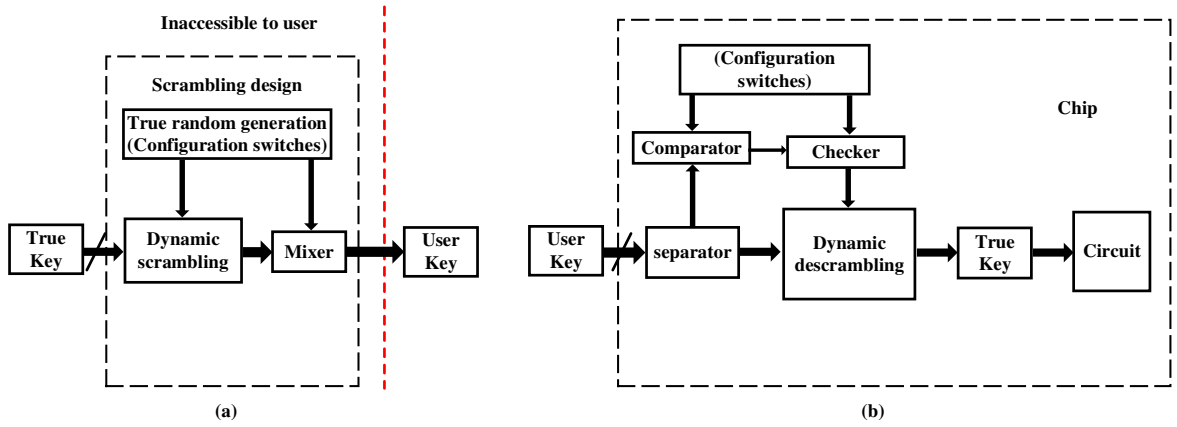

Figure 10. Scrambling technique used for (**a**) scrambling and (**b**) descrambling.

On the rising edge of a Fetch operation (i.e., for a new instruction), the random generator sets a new scrambler configuration. This configuration is saved in a new segment of memory given by a first input first output (FIFO) and the address used is saved with the scrambled data in random memory. Concurrently, each time a new configuration is requested to unscramble data, the configuration is read in the memory at the address given by the random memory and the data is unscrambled. This address is saved in a FIFO that stores all empty memory addresses. When a configuration value is read, the memory block that holds the value should be overwritten to avoid risk of reuse.

More specifically, the designer provides the input of the dynamic scrambling by the secret key of the circuit, and the random generation is used to generate new configuration bits. The scrambling output bits are randomly combined with the configuration bits via a Mixer to produce the user key. The end-user uses this key to decrypt the circuit, where the user key will be separated into the configuration bits and the scrambling output bits. The incoming configuration bits will be compared with the configuration bits in the chip that the designer already burns in a non-volatile memory and the rest of the incoming bits (scrambling output bits) will be fed as inputs to the dynamic unscrambling.

In this case, the encrypted chip will only be activated by the secret key if both of the comparator output and the unscrambling key are correct.

6.3. Testing in an Untrusted Foundry

Since the complexity of the design becomes very large and needs different types of equipment as well as fabricating process involvement [24], many companies design the ICs and then fabricate them at other companies. Thus, ICs might be imitated by the untrusted company so the company could sell them in the markets illegally or insert a Trojan inside the chips [38]. As a result, the developers have no control over untrusted foundries to protect their designs, where they are susceptible in the face of several attacks [39]. IP owners can protect their design from counterfeit ICs and other attacks in a company during the test using the Secure Split-Test (SST) method before sending the ICs to the trusted facility for configuring its functionality [40]. SST protocol is based on communicating and exchanging generated keys between the foundry and the designer, where only the IP owner can know whether the IC is passing the test successfully or not. An improvement on SST is achieved, namely CSST, which gives a simple communication between the IP owner and the foundry as well as providing more protection than the traditional SST. In this technique, the designer has full control over the chip, and only he or she can understand and analyze the result of the locked chip [41].

6.4. Beyond SiNW FETs

Besides the proposed SiNW FETs, other emerging transistors might also be employed to protect IP designs. For instance, the recently proposed negative capacitance FET (NCFET) [42] is embedded with the property of tunability. By adding a ferroelectric layer in the gate stack of a MOSFET, NCFET is able to reduce the switching slope to a value less than 60 mV/dec, which shows potential for ultra low-power design. Meanwhile, since it can be configured in a way that may or may not have hysteresis loop, one NCFET can virtually function in two modes: memory cell and Boolean logic cell. The difference between two modes is determined by the thickness of ferroelectric layers, which is on the sub-nanometer scale. Once the NCFET fabrication is done, it is extremely difficult to distinguish which mode NCFET stays because the reverse engineering cannot have the advanced SEMs to identify the devices.

7. Conclusions

In this paper, we have demonstrated that the usage of emerging transistor, i.e., SiNW FETs, can help improve the logic locking design by preserving lower power and area consumption compared to conventional CMOS technology. Specifically, the SiNW-based polymorphic gates work as logic key units to encrypt the combinational circuits. A smart placement algorithm is formalized and it shows that 50% of Hamming distance between the correct and wrong output bits can be achieved through a security assessment. We showed that, besides the traditional criteria for emerging devices such as area, power, delay and non-volatility, security may serve as a new criterion to thoroughly judge the pros and cons of any emerging devices. Using this new standard, we plan to revisit existing emerging transistors to have a full comparison between emerging technologies and CMOS technology. Meanwhile, we believe that more research outcomes are expected in this area where unique properties of emerging transistors can help in enhancing the circuit security.

Acknowledgments: This work was supported in part by the Higher Committee for Education Development (HCED) in Iraq scholarship.

Author Contributions: Qutaiba Alasad proposed the ideas, implemented the designs including obtaining the experimental results, and wrote the manuscript. Jiann-Shuin Yuan proved the idea, reviewed the manuscript and gave technical feedback. Yu Bi evaluated the performance overhead for the proposed technique and discussed the writing. The final manuscript has been read and confirmed by all authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Computing Research Association (CRA). Defense Science Board (DSB) Study on High Performance Microchip Supply. Available online: http://www.cra.org/govaffairs/images/2005-02-HPMS_Report_Final.pdf (accessed on 28 March 2016).
2. Adee, S. The Hunt for the Kill Switch. *IEEE Spectr.* **2008**, *45*, 34–39.
3. Yeh, A. *Trends in the Global IC Design Service Market*; DIGITIMES Research: Taipei, Taiwan, 2012.
4. International Chamber of Commerce (ICC). *Estimating the Global Economic and Social Impacts of Counterfeiting and Piracy: A Report Commissioned by Business Action to Stop Counterfeiting and Piracy (BASCAP)*; Frontier Economics Ltd.: London, UK, 2011.
5. Rajendran, J.; Pino, Y.; Sinanoglu, O.; Karri, R. Security Analysis of Logic Obfuscation. In Proceedings of the DAC'12 49th Annual Design Automation Conference, San Francisco, CA, USA, 3–7 June 2012; pp. 83–89.
6. Bi, Y.; Shamsi, K.; Yuan, J.-S.; Gaillardon, P.-E.; De Micheli, G.; Yin, X.; Hu, X.S.; Michael, N.; Jin, Y. Emerging Technology based Design of Primitives for Hardware Security. *J. Emerg. Technol. Comput. Syst. ACM* **2016**, doi:10.1145/2816818.
7. Bi, Y.; Shamsi, K.; Yuan, J.S.; Standaert, F.X.; Jin, Y. Leverage Emerging Technologies for DPA-Resilient Block Cipher Design. In Proceedings of the 2016 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 14–18 March 2016; pp. 1538–1543.
8. Bi, Y.; Shamsi, K.; Yuan, J.S.; Jin, Y.; Niemier, M.; Hu, X.S. Tunnel FET Current Mode Logic for DPA-Resilient Circuit Designs. *IEEE Trans. Emerg. Top. Comput.* **2016**, *5*, 340–352.
9. Colli, A.; Pisana, S.; Fasoli, A.; Robertson, J.; Ferrari, A.C. Electronic transport in ambipolar silicon nanowires. *Phys. Status Solidi (b)* **2007**, *244*, 4161–4164.
10. Geim, A.K.; Novoselov, K.S. The rise of graphene. *Nat. Mater.* **2007**, *6*, 183–191.
11. Martel, R.; Derycke, V.; Lavoie, C.; Appenzeller, J.; Chan, K.K.; Tersoff, J.; Avouris, P. Ambipolar Electrical Transport in Semiconducting Single-Wall Carbon Nanotubes. *Phys. Rev. Lett.* **2001**, *87*, 256805.
12. Appenzeller, J.; Knoch, J.; Tutuc, E.; Reuter, M.; Guha, S. Dual-gate silicon nanowire transistors with nickel silicide contacts. In Proceedings of the 2006 IEDM'06 International Electron Devices Meeting, San Francisco, CA, USA, 11–13 December 2006; pp. 1–4.
13. De Marchi, M.; Sacchetto, D.; Frache, S.; Zhang, J.; Gaillardon, P.; Leblebici, Y.; De Micheli, G. Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs. In Proceedings of the 2012 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 10–13 December 2012; pp. 8.4.1–8.4.4.
14. Gaillardon, P.-E.; Bobba, S.; De Marchi, M.; Sacchetto, D.; De Micheli, G. NanoWire Systems: Technology and Design. *Philos. Trans. R. Soc. Lond. A* **2014**, *372*, doi:10.1098/rsta.2013.0102.
15. Stoica, A.; Zebulum, R.S.; Keymeulen, D.; Ferguson, M.I.; Duong, V. Taking evolutionary circuit design from experimentation to implementation: Some useful techniques and a silicon demonstration. *IEE Proc. Comput. Digit. Tech.* **2004**, *151*, 295–300.
16. Stoica, A.; Zebulum, R.S.; Keymeulen, D. *Polymorphic Electronics*; Springer: Berlin, Germany, 2001; pp. 291–302.
17. Ruzicka, R. New polymorphic NAND/XOR gate. In Proceedings of the 7th WSEAS International Conference on Applied Computer Science, Venice, Italy, 21–23 November 2007; Volume 2007, pp. 192–196.
18. Rajendran, J.; Zhang, H.; Zhang, C.; Rose, G.S.; Pino, Y.; Sinanoglu, O.; Karri, R. Fault Analysis-Based Logic Encryption. *IEEE Trans. Comput.* **2015**, *64*, 410–424.
19. Roy, J.A.; Koushanfar, F.; Markov, I.L. EPIC: Ending Piracy of Integrated Circuits. In Proceedings of the 2008 DATE '08 Design, Automation and Test in Europe, Munich, Germany, 10–14 March 2008; pp. 1069–1074.
20. Rajendran, J.; Pino, Y.; Sinanoglu, O.; Karri, R. Logic encryption: A fault analysis perspective. In Proceedings of the Conference on Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 12–16 March 2012; pp. 953–958.

21. Baumgarten, A.; Tyagi, A.; Zambreno, J. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Des. Test Comput.* **2010**, *27*, 66–75.
22. Alasad, Q.; Bi, Y.; Yuan, J. E^2LEMI:Energy-Efficient Logic Encryption Using Multiplexer Insertion. *Electronics* **2017**, *6*, 16.
23. Subramanyan, P.; Ray, S.; Malik, S. Evaluating the security of logic encryption algorithms. In Proceedings of the 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Washington, DC, USA, 5–7 May 2015; pp. 137–143.
24. Yasin, M.; Rajendran, J.; Sinanoglu, O.; Karri, R. On Improving the Security of Logic Locking. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *35*, doi:10.1109/TCAD.2015.2511144 .
25. Alasad, Q.; Yuan, J.; Fan, D. Leveraging All-Spin Logic to Improve Hardware Security. In Proceedings of the GLSVLSI'17 on Great Lakes Symposium on VLSI 2017, Banff, AB, Canada, 10–12 May 2017; pp. 491–494.
26. Xie, Y.; Srivastava, A. Mitigating SAT Attack on Logic Locking. Cryptology ePrint Archive, Report 2016/590, 2016. Available online: http://eprint.iacr.org/2016/590 (accessed on 17 January 2017).
27. Yasin, M.; Mazumdar, B.; Sinanoglu, O.; Rajendran, J. Security analysis of Anti-SAT. In Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, 16–19 January 2017; pp. 342–347.
28. Yasin, M.; Mazumdar, B.; Rajendran, J.J.V.; Sinanoglu, O. SARLock: SAT attack resistant logic locking. In Proceedings of the 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 3–5 May 2016; pp. 236–241.
29. Yasin, M.; Sengupta, A.; Schafer, B.; Makris, Y.; Sinanoglu, O.; Rajendran, J. What to Lock?: Functional and Parametric Locking. In Proceedings of the GLSVLSI'17 on Great Lakes Symposium on VLSI 2017, Banff, AB, Canada, 10–12 May 2017; pp. 351–356.
30. Chakraborty, R.S.; Bhunia, S. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2009**, *28*, 1493–1502.
31. Rajendran, J.; Kanuparthi, A.K.; Zahran, M.; Addepalli, S.K.; Ormazabal, G.; Karri, R. Securing Processors Against Insider Attacks: A Circuit-Microarchitecture Co-Design Approach. *IEEE Des. Test* **2013**, *30*, 35–44.
32. Hansen, M.C.; Yalcin, H.; Hayes, J.P. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Des. Test* **1999**, *16*, 72–80.
33. Shen, Y.; Zhou, H. Double DIP: Re-Evaluating Security of Logic Encryption Algorithms. In Proceedings of the GLSVLSI'17 on Great Lakes Symposium on VLSI 2017; Banff, AB, Canada, 10–12 May 2017; pp. 179–184.
34. Shamsi, K.; Li, M.; Meade, T.; Zhao, Z.; Pan, D.Z.; Jin, Y. Cyclic Obfuscation for Creating SAT-Unresolvable Circuits. In Proceedings of the GLSVLSI'17 on Great Lakes Symposium on VLSI 2017, Banff, AB, Canada, 10–12 May 2017; pp. 173–178.
35. Zhou, H.; Jiang, R.; Kong, S. CycSAT: SAT-Based Attack on Cyclic Logic Encryptions. In Proceedings of the International Conference on Computer-Aided Design (ICCAD), Irvine, CA, USA, 13–16 November 2017.
36. El Massad, M.; Garg, S.; Tripunitara, M.V. Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ics within minutes. *NDSS* **2015**, doi:10.14722/ndss.2015.23218.
37. Dubeuf, J.; Hély, D.; Karri, R. Run-time detection of hardware Trojans: The processor protection unit. In Proceedings of the 2013 18th IEEE European Test Symposium (ETS), Avignon, France, 27–30 May 2013; pp. 1–6.
38. Yasin, M.; Sinanoglu, O. Transforming between logic locking and IC camouflaging. In Proceedings of the 2015 10th International Design Test Symposium (IDT), Amman, Jordan, 14–16 December 2015; pp. 1–4.
39. Rajendran, J.; Sinanoglu, O.; Karri, R. Regaining Trust in VLSI Design: Design-for-Trust Techniques. *Proc. IEEE* **2014**, *102*, 1266–1282.
40. Contreras, G.K.; Rahman, M.T.; Tehranipoor, M. Secure Split-Test for preventing IC piracy by untrusted foundry and assembly. In Proceedings of the 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), New York, NY, USA, 2–4 October 2013; pp. 196–203.

41. Rahman, M.T.; Forte, D.; Shi, Q.; Contreras, G.K.; Tehranipoor, M. CSST: Preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly. In Proceedings of the 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Amsterdam, The Netherlands, 1–3 October 2014; pp. 46–51.
42. Dasgupta, S.; Rajashekhar, A.; Majumdar, K.; Agrawal, N.; Razavieh, A.; Trolier-Mckinstry, S.; Datta, S. Sub-kT/q Switching in Strong Inversion in $PbZr_{0.52}Ti_{0.48}O_3$ Gated Negative Capacitance FETs. *IEEE J. Explor. Solid State Comput. Devices Circuits* **2015**, *1*, 43–48.