The ISC Int'l Journal of Information Security

July 2016, Volume 8, Number 2 (pp. 93–104)

http://www.isecure-journal.org

ASIC Design Protection against Reverse Engineering during the Fabrication Process using Automatic Netlist Obfuscation Design Flow☆

Sharareh Zamanzadeh [1,*], and Ali Jahanian [1]

[1] *Computer Science and Engineering Department, Shahid Beheshti University, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 20 November 2015
First Revised: 27 April 2016
Last Revised: 30 Jun 2016
Accepted: 13 July 2016
Published Online: 26 July 2016

Keywords:
Hardware Security, Netlist Encryption, Obfuscation, Reverse Engineering, IP Piracy.

ABSTRACT

Fab-less business model in semiconductor industry has led to serious concerns about trustworthy hardware. In untrusted foundries and manufacturing companies, submitted layout may be analyzed and reverse engineered to steal the information of a design or insert malicious Trojans. Understanding the netlist topology is the ultimate goal of the reverse engineering process. In this paper, we propose a netlist encryption mechanism to hide the interconnect topology inside an IC. Moreover, new special standard cells (Wire Scrambling cells) are designed to play the role of netlist encryption. Furthermore, a design flow is proposed to insert the WS-cells inside the netlist with the aim of maximum obfuscation and minimum overhead. It is worth noting that this mechanism is fully automated with no need to detail information of the functionality and structure of the design. Our proposed mechanism is implemented in an academic physical design framework (EduCAD). Experimental results show that reverse engineering can be hindered considerably in cost of negligible overheads by 23% in area, 3.25% in delay and 14.5% in total wire length. Reverse engineering is evaluated by brute-force attack, and the learned information is 0% and the Hamming distance is approximately 50%.

1 Introduction

1.1 Motivation, Contribution and Organization

Decreasing feature size and time to market, accompanied by demands for lower power and higher performance ICs imposed a new paradigm shift in semiconductor business model, called globalization. With the advent of globalization, IC backend design and fabrication processes are outsourced and the IC supply chain is flattened and driven to specialize. IP vendors are established to create functional units which they license to IC designers to apply in their ASIC designs. Also, contract foundries are emerged to manage economies of scale related to their contracted design houses and fabrication companies. Thus the whole IC supply chain is not under authority of the designer companies anymore [1]. On the other hand, master engineers in the foundries and fabrication companies have full access to whole technology libraries, purchased IPs, and layout of the designs. Therefore, this change in semiconductor industry has caused un-

☆ This article is an extended version of an ISCISC'12 paper.

* Corresponding author.
Email addresses: sh_zamanzadeh@sbu.ac.ir (S. Zamanzadeh), jahanian@sbu.ac.ir (A. Jahanian)

ISSN: 2008-2045

trustworthy environment with remarkable security concerns such as reverse engineering.

Reverse engineering is practical during all stages of design flow to extract the functionality of the chip /IP. Adversaries may use this technique to steal the design ideas and make fake IC/IP or to insert hard-to-detect Trojans by obtaining comprehensive information about the victim hardware to camouflage Trojans [2]. In this situation, by preventing reverse engineering and hiding circuit functionality, adversaries may not be able to access into critical confidential information of the circuit or determine the best location for Trojan insertion.

In this paper, our suggested technique hinders the reverse engineering against theft and trust during the chip fabrication. In the proposed technique, special standard cells (Wire Scrambling cells) are designed andinjected into the std-cell library. Then these cellsare inserted on the design netlist at physical level. The duty of wire scrambling cells (WS-cells) is to encrypt net interconnections. In this scenario, circuit will be incomplete without applying key to WS-cells. In other words, original netlist topology can be realized only when the correct key is applied to WS-cells. Otherwise any incorrect key makes a different netlist topology and causes inaccurate functionality in the circuit.

It is obvious that reverse engineers are practically impossible when a circuit is incomplete or it has incorrect netlist. As will be shown, the probability of extracting the right topology and therefore accurate functionality with the absence of WS-cells' correct key is practically zero in a scrambled design. Our experiments and analyses show that this technique can be automated easily at the physical design flow with the negligible overhead in area, power consumption and EDA execution time. The main contributions of this research are as follows:

- Wire scrambling cells are proposed as efficient element for encrypting the routing topology of netlist.
- A new netlist encryption design flow is proposed to prevent GDSII files against IP cloning and reverse engineering.
- An intelligent net selection criteria and heuristic algorithm is provided to improve hardware immunity with the minimum cost of overhead.

The rest of this paper is organized as follows. Threat model is discussed in Section 2. Section 3 describes the proposed WS-cells and in Section 4 the proposed netlist scrambling algorithm is delineated. The strength of netlist encryption mechanism against information leakage is analyzed in Section 5. Experimental results are presented in Section 6 and finally, Section 7 concludes the paper.

1.2 Literature Review

Extensive research has been reported to improve IC supply chain security in the fields of theft and trust. Most of these ideas are directly or indirectly related to Obfuscation techniques. Obfuscation is a technique to immunize the design against cruel reverse engineering while its functionality remains unchanged [3] and [4].

Hardware encryption is a recent approach to prevent against manufactured attacks and has been focused on logic/structural encryption techniques. One group of these techniques is specially designed for sequential logic. They are mainly implemented in FSM locking at RTL/gate level [4],[5],[6] and [7]. In [4], Charkraborty *et al.* utilized the FSM locking as a way of Trojan prevention. Also, in a similar manner, Koushanfar *et al.* introduced an FSM locking mechanism which is accompanied by unclonable random unique function (RUB) to remotely control ICs activation and overcome piracy [7]. In [8] Rajendran *et al.* proposed a metric (search space) to quantify the strength of designs. Also, they have provided a secure high-level synthesis design flow that adds decoy connections to maximize the search space. This research is dedicated to protect sequential circuits (controller unit) and the implemented techniques require too much information about the design. The automation process has a great complexity; it is not easily automatable.

Some approaches suggested adding XOR/XNOR, MUX extra gates to conceal the functionality of a design and corrupt the outputs [9], [10], [11] and [12]. In the [10] and [11], logic encryption approach is used to encrypt the design functionality. In these papers, the output data is protected and guaranteed maximum invalidation in output vectors, when the wrong key is applied to the circuit. However, the whole netlist and layout of the circuit is available and open in the foundry/fab. Therefore, netlist level analysis and functional dependency checking for wires are still possible to find the best place for camouflage malicious logics such as backdoors. In order to protect the design from the adversaries, more complicated protection mechanism is required than logic encryption technique in [9], [10] and [11]. Therefore, in the present paper, we proposed a geometric encryption approach in which netlist topology of the circuit is encrypted to provide higher security level to protect the design against reverse engineering during the fabrication process. Moreover, the proposed process in [10] and [11] is a heavy pre-process and it is handled with a discrete tool from synthesis/physical design tools. Automating this method is not that much easy and is

time-consuming. In [12] the author has used a PUF circuit to generate the key for the logic obfuscation. The PUF circuit guarantees a specific key for each individual IC; therefore this mechanism provides IC-metering as well, but at peculiar costs that should be balanced with the value of its applications.

Authors of [13] presented a logic obfuscation which allows the designer not to send the entire schematic of the IP to the foundry. They divide the whole functionality into two parts; fixed and configurable in which fixed parts are fabricated as ASIC and configurable parts are implemented using LUTs and configured at runtime. Therefore, real circuit cannot be understood before usage. However, this partitioning needs deep information about the design and its features such as circuit functionality, critical nodes of the design and whole design netlist. This information would draw the EDA execution out and it is not automated easily. Also, this technique needs a re-synthesize phase to map the selected function to the LUTs that leads to a complicated and time-consuming design flow. In addition, manufacturing of LUTs will increase the cost of the chip.

Netlist encryption is a novel approach in hardware encryption as a solution against three main concerns of IC fabrication: metering, theft and trust. As far as we know, netlist topology encryption concept is proposed for the first time in [14] in which the netlist topology is obfuscated in such a way that the correct netlist of the design cannot be generated without applying the correct key. However, the proposed algorithm in [14] is a uniform random selection path (partition the circuit based on cut size and insert one WS-cell in each partition). The security analysis also is limited to the search space enlargement that is caused by WS-cells insertion. However, netlist topology encryption in the present paper is done via an intelligent algorithm. This algorithm chooses best wires for netlist encryption to cause minimum overheads and maximum obfuscation. Moreover, inserting the WS-cells at the physical design can be done with a simple netlist graph processing (BFS) without any information about the functionality of circuit.

Authors of [15] has combined the idea in [14] and [13] and provide a protection scheme in which the search space and the attack cost increase exponentially, while the overhead imposed on the designer's side grows only linearly. However, the authors have confined to the novel scheme. It is not clear, how the obfuscation primitives are inserted, which part of the circuit and to what extent is going to be chosen to be obfuscated in LUTs. In other word, this paper is lacking in automated design flow and it is not easy automatable.

2 Threat Model

In this paper, Trojan insertion and IP cloning with the use of reverse engineering are considered as the threats for security. These malicious activities are supposed to be carried out when the layout-level geometry is generated, during fabrication process or at foundries. It is worth noting that our approach targets IP-protection instead of IC-protection. In each tape-out, the design is encrypted with a different key which is the same for all IC instances. In other word the proposed encryption technique guarantees the trust of fabricated chips for each tape-out without aim of controlling overbuilding.

The attacker is neither IP designer nor IP providers, but competitors or ill-wishers who are skill full in circuit design at foundries or manufacturing companies. The opponents reach valuable information like layout detail information and test vectors with expected responses. They also possess advanced technical knowledge of IC design and fabrication and reach to the reverse engineering, test and design tools and also they have ability to fabricate ICs. We assume no computational, financial and temporal limitation for the attackers. The aim of the attacker may be knowledge acquisition, taking the control of ICs or making profit from piracy. Moreover, hard-to-detect Trojans are considered in this paper whose insertion requires the information gained by reverse engineering of the circuit. In other words, the Trojans whose insertion does not involve to functionality information of the circuit are not target of this paper.

In this paper, proposed technique known as *Netslist topology encryption* allows designer to send incomplete netlist to the foundry/fab and IP/IC will not be activated or understood in the absence of legal key. Correct key will be applied at personalization stage, similar to personalization process of smart cards in issuing stage. Personalizing chip by correct key is carried out after all supply chain stages: foundry, fabrication and shipment, right before sending the chip to the market to be used.

3 Proposed Wire Scrambling Cells

As mentioned in Section 1, a new netlist obfuscation technique is proposed to hinder the reverse engineering in ASIC design flow. In the proposed technique, special standard WS-cells (Wire Scrambling cells) are designed and inserted to the netlist. Nets connected to these key-based WS-cells are scrambled at physical level such that the circuit connections are shuffled. The original topology will be made just after applying the correct key to WS-cells. In the scrambled design, probability of extracting the right topology and therefore true functionality with the absence of correct key

is practically zero. Figure 1 illustrates a simple diagram of the proposed technique. Figure 1-a shows original topology of 4 wires (n0, n1, n2, and n3) in the netlist. Figure 1-b represents the symbolic view of WS-cell and Figure 1-c represents the incorrect topology of the scrambled nets after applying an incorrect key.

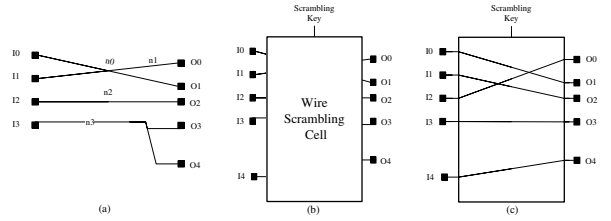

Figure 1. (a) Original connection (b) Symbolic view of WS-cell (c) False topology resulted by an incorrect key

.

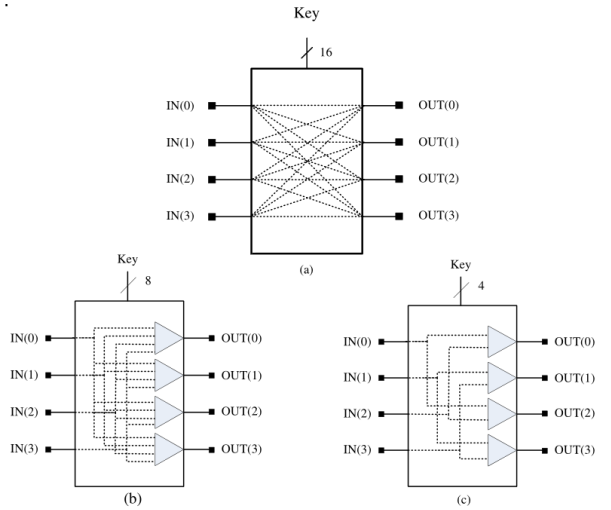

Figure 2. a) Structure of a generic WS-cell b) Mux base WS-cells structure, Full shuffler c) Partial shuffler

A generic WS-cell can be implemented by using multiplexers or pass transistors as shown in Figure 2-a. In an n-bit wire scrambler, $n \times n$ bit scrambling key is required for encoding/decoding its inputs/outputs. However, with the same connection space, MUX-based WS-cell shown in Figure 2-b requires $(n \times log_2 n)$ bit key signals; which is a beneficial alternative. Each WS-cell may be full or partial shuffler. In the former, each input can be connected to each output pins (Figure 2-b) while in the latter each input can be connected to only one sub-set of outputs (Figure 2-c). It is obvious that full shuffling provides higher security due to the larger provided connection space.

In WS-cells, repetition of wire connections is allowable. It means each input wire of WS-cell can be connected to more than one output wires. For example, if we select multi-terminal net(s) as the inputs of WS-cells, the correct key shall connect one input to more than one output (repetitive connection). In this case some inputs of WS-cell would not drive any outputs of the WS-cell when the correct key is applied. However, these inputs of WS-cells are connected to other standard cells through dummy wires to let WS-cells seems full connected. We call these wires, dummy wires, since they are connected to the netlist, but they are not expected to have any sink terminals through the WS-cell. Dummy wires are useful in keeping the level of ambiguity. Figure 3 shows a simple example of a multi-terminal net connection in a WS-cell. In this fig, input IN(0) is connected to the multi-terminal net and correct key connects this input to all outputs of the WS-cell and other input pins of the WS-cell are connected to some dummy wires. Dummy wires are those design wires that are connected to input pins of WS-cell and are not expected to be routed to the WS-cell output pins. Using the dummy wires extends the ambiguity and search space of the solution up to n^n, practically.

The WS-cells are standard cells and the internal connections would be discovered whenever the key is applied to them. As a result, for each output there are N possible input choices to be connected. In the case of full-shuffling architecture, all inputs are connected to any output through an instance of MUXnx1. Since, each MUX has $(log_2 n)$ bit selector signals, WS-cell would have $(n \times log_2 n)$ bits selector key.

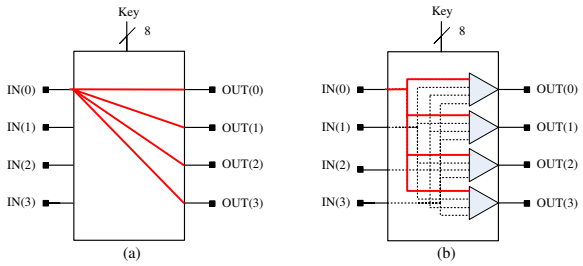

Figure 3. The Multi-terminal nets connection in WS-cells

An *n-input and n-output full shuffler* WS-cell has n^n different cases for connecting the input pins to the output pins. For example, in a 4-input and 4-output WS-cell, $4^4 = 256$ states of scrambling can be occurred. In a MUX-based full shuffler WS-cell, the probability of finding the correct topology (*ideal probability of reverse engineering*) is as:

$$IPRE = \underbrace{\frac{1}{n} \times \frac{1}{n} \times ... \frac{1}{n}}_{n} = \frac{1}{n^n} \qquad (1)$$

with m number of $n \times n$ full-shuffler WS-cell in a design, this probability in whole design is as:

$$IPRE = \underbrace{\frac{1}{n^n} \times \frac{1}{n^n} \times ... \frac{1}{n^n}}_{m} = \frac{1}{n^{n \times m}} \qquad (2)$$

As can be seen in Equation (2), the probability of finding the correct netlist using 16, 4×4 WS-cell is $(\frac{1}{4^{64}})$ that is close zero.

We designed the WS-cells with different sizes. Table 1 shows the designed WS-cells in which columns I/O width shows the number of input/outputs in a WS-cell, respectively. The columns labeled pattern space and guess prob. indicate the number of possible scrambling states and probability of successful guess in a WS-cell, respectively and finally, Power and Area represent the estimated dynamic power (with FO4 load) and area of the cells, respectively. It should be noted that all the standard cells in the used library are of the same height ($7.92\mu m$).

As can be observed in Table 1, large WS-cells have considerable amount of area and power overheads that may not be feasible in real applications. Moreover, larger WS-cells have more number of key bits. On the other hand, pattern space of the moderate cells is large enough to be used in secure applications. Therefore, it is recommended to use 4×4 full-shufflers WS-cells.

Table 1. Characteristics of designed wire scrambling cells

WS Name	I/O width	Search Space	Guess Prob.	Power $(\frac{nW}{MHZ})$	Area (μm^2)
WS2	2x2	4	0.25	8	125.45
WS4	4x4	256	3.90E-03	48	501.80
WS8	8x8	16.77E+6	5.96E-08	224	2007.25
WS16	16x16	1.845E+19	5.42E-20	480	8028.98

4 Proposed Wire Encryption Process

In the proposed technique, netlist topology is encrypted using WS-cells and during the foundry process, manufacturing and shipment the design is transferred without key transmission. This incomplete tape out of design (tape-out without key transmission) has two advantages as follows:

- First no key management is required during the fabrication process to avoid the complicated process and extra implementation costs of key management. It is worth noting that on-chip key storage techniques such as fuse-based ROMs, tamper-resistant memories, unreadable VLSI masks, using special sensors and so on lead to huge complexity and cost in design, manufacturing and test process of the chip. Moreover, key transmission protocols between design house and foundry/fab have considerable security side-effect and costs that are avoided in the proposed methodology.
- Second, there is lower chance to leaks design information to adversaries at untrusted foundry and manufacturing companies.

In our strategy, key is applied at personalization stage which is after foundry, fabrication and shipment process, thus the attackers at foundry stage do not reach to the key and have no information about the circuit functionality. This mechanism is similar to the personalization process of smart cards at issuing stage in which chips become active by chip-provider/end-user before using the chip. Consequently, customers would be sure that each fabricated design party is protected during fabrication. In high-secure applications, netlist encryption keys can be changed for each tape-out. It is worth to note that the key can be applied serially (similar to test scan chain) to meet pin constraints.

An important point is that post-wafer testing at chip fabrication stage does not require the correct key, necessarily. Because the chip designers knows the behavior of the chip when an incorrect key is applied. Hence, they can give the benchmarks to fab for testing the chip comprising an incorrect key. Consequently, no information about the key and real topology of design transmits/leaks to the fab. The rest of this section describes the proposed algorithm which selects the suitable nets to be scrambled through WS-cell resources. The metrics and conditions of the nominated nets are mentioned as well.

4.1 Wire selection metrics and Netlist encryption algorithm

Net Selection for assigning to the WS-cells is very important in increasing the efficiency of WS-cells and reducing the overheads.

4.1.1 Wire selection metrics

There are some significant criteria in net selection:

- Number of terminals: multi-terminal nets are more suitable than two terminal nets. Since the more terminal numbers a selected net has, the more pins would be disconnected from the netlist and ambiguity will spread in the design.
- Toggle rate: Toggle Rate is the rate at which a net or logic element switches, in comparison with its input(s). Toggle rate is expressed as a percentage [16] and [17]. In case that, the nets involved in encryption have high toggle rate, wrong key (wrong connection) may cause more output variation in the WS-cell.
- Observability and controllability: controllability is defined as possibility of setting the value of a certain netlist node and observability is defined as possibility of observing the changed value of a net on a primary output [17]. Due to the less probability of Trojan activation and detection, nets with low observability/ controllability are

more desirable for insertion of hard-to-detect Trojans [2]. On the other hand, low controllable/observable nets are less testable and discovering the key of WS-cells would be more difficult. The nets near the primary outputs are highly observable and nets near the primary inputs are highly controllable. The nodes in the middle of the network blend both controllability and observability, helping to increase the attacker's burden. Some simulation and analyses have been reported in the literature as [13] and [17] that supports this claim. Therefore, the nets at the middle of the DAG (Directed Acyclic Graph) graph are best candidates to be used in the net-obfuscation. DAG graph is an acyclic graph, $N = (V, E)$, where N is a logic network, V is a set of nodes, and E is the set of edges. The nodes in V represent a standard cell (logic gates), and the interconnections represent the pre-routed nets.

Also, below complementary conditions are applied during net selection process:

- The WS-cells should not be inserted on the critical path. It is obvious that inserting WS-cells on the critical path would have a great side effect in the performance of the circuit.
- Select a net only one time. This condition is important since standing two or more back to back WS-cells, will increase the delay with no improve in security level.
- Separate nets should be selected. Two nets are called separate if, they do not belong to the same path from primary inputs to primary outputs. By selecting separate nets no two WS-cells would be located one after the other along the path, therefore the delay caused by WS-cells does not grow, and timing constraints of the host circuit will not be violated.

Meeting above complementary conditions indicates the avoidance of inefficient WS-cell insertion that may cause timing side-effects or loosing obfuscation gain.

4.1.2 Netlist encryption algorithm

WS-cell insertion process is taking place at the early stages of physical design. In this step the circuit has been synthesized and the final netlist has been extracted. However, placement step is not carried out yet. Therefore, this stage is the best time for modifying netlist by WS-cells since the characteristic of nets is specified completely and after WS-cell insertion regular placement tools can determine the best locations of the WS-cells as well as other standard cells.

The proposed algorithm is a bi-partition based algorithm that cuts the circuit into two separate sub-graphs trough, which all paths from primary inputs to primary outputs are crossed. WS-cells are inserted on the cut edges. Nets in the middle of the DAG graph are candidate as low controllable/observable nets, so the best place for this cut is nets around middle of the DAG graph. Figure 4 is a simple example that illustrates the process of WS-cell insertion in a small circuit. Figure 4-a shows the original circuit, in Figure 4-b the DAG graph of the circuit has been extracted and the edges which are located in the middle of graph are highlighted (dark arrows) as cut-list. The WS-cells (with the size of 2x2) are connected to the cut-nets. The dashed net is a dummy wire.

Finding the middle of the DAG graph is a 2-directional BFS algorithm that traverses the graph from primary inputs/outputs toward center of the graph simultaneously. It means that we traverse toward the center of the graph, hub by hub in both directions in turn. The cut line would be formed at the confluence of these two traverses.

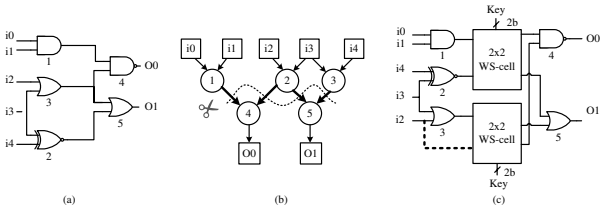

Figure 4. a) Original circuit b) Cut-nets in the DAG graph c) Encrypted netlist in the circuit

It is possible to change the height of the cut in very few steps toward the primary outputs/inputs. In this regard we would give high priority to *most toggle rate* nets and those with more terminals. The nets whose toggle rates are in the range of 10% of most transaction rates are labeled as *most toggle rate*. Figure 5 shows the proposed algorithm for net selection and WS-cell standard cells insertion. In the following paragraphs,

Physical design flow consists intelligent WS-cell insertion algorithm	
Step 1:	Extract graph from synthesized netlist.
Step 2:	Intelligent WS-cell insertion algorithm
Step 2-1:	Calculate the height of each net in the DAG graph.
Step 2-2:	Obtain nets in the middle of graph
Step 2-3:	Refinement cut-list
Step 2-3-1:	Move at most two Hub toward primary inputs/primary outputs
Step 2-3-2:	For each movement: if new movement is more efficient
Step 2-3-3:	Replace (new-list → cut-list);
Step 2-3-4:	Report the best list as cut-list;
Step 2-4:	Connect cut-nets to WS-cells: CALL "Key generation algorithm"
Step 3:	Canonical Placement.
Step 4:	Canonical Routing.

Figure 5. Physical design flow consisting of the WS-cell insertion algorithm

details of this algorithm is described.

Step 1: At the first step, the synthesized netlist which is generated by the standard synthesis tools (design compiler Synopsys tool) is parsed and DAG graph is created.

Step 2: In this stage of the flow, the suitable nets are selected and list as cut-list.

Step 2-1: The distance of each net from primary inputs and primary outputs are calculated.

Step 2-2: Those nets that have equal distance both from primary outputs and primary inputs, would be selected as cut-list.

Step 2-3: A refinement process is done. During this process, movements with maximum two hubs toward primary inputs/outputs are done; and check which one of these movements is more efficient than present cut-list. Finally, the refinement process reports the best list.

Step 2-3-1 to 2-3-4: For each movement, the total terminal numbers of nets are calculated ($\sum terminal\#$) and the net numbers which are labeled as *most toggle rate* are counted. The movement that has the most amounts of these variables would be the nominated list.

Step 2-4: Call *key generation* algorithm to connect nets of cut-list to WS-cells.

Steps 3 and 4: After WS-cells insertion, rest of the canonical physical design flow (e.g. placement and routing) is performed. Placement and routing algorithms consider WS-cells as standard cells.

The explained algorithm satisfies the complementary conditions, which are mentioned above. First if a net in cut-list is on the critical path, this net will not be connected to the WS-cells. Secondly, before reporting cut-list the WS-cell insertion algorithm will remove any redundant net in the cut-list and there would not be any chance to have more than one WS-cell on one net. Finally, since DAG graph is an acyclic graph and all nets in cut-list are in the same height in the DAG graph, there would not be any two nets that are belong to a same path. As mentioned before, satisfying above complementary conditions shows that with a specified number of WS-cells, the best obfuscation gain with minimum performance overhead has been obtained.

4.2 Key Generation algorithm

Each WS-cell has 8-bits key and the total key of the circuit is generated by concatenation of these 8 bits. Therefore, the length of total key is equal to $8 \times number$ of WS-cells. Every WS-cell has a randomly generated key and they are independent of each other. Key can be applied to the chip in personalization phase after fabrication. Figure 6 demonstrates key generation algorithm in details.

Key Generation Algorithm	
Step1:	Fragment cut-list to 4 member subsets. (According to the capacity of WS cell).
Step2:	Choose one subset and connect nets to the WS-cells.
Step2-1:	Loop if all 4 nets in the subset are not processed.
Step3:	Concatenate the key of present WS-cell with previous ones to make the total key.
Step4:	If all sets are not processed go to **Step 2**.

Figure 6. Description of key generation process

Step1: First we fragment the cut-list to the subsets of nets with 4 members (4=n the size of WS-cell). The nets with the same source terminal should be grouped in one sub-set.

Step2: $\forall$ sub-set $\subset$ cut-list, connect this sub-set to a WS-cell. The nets of the sub-set are randomly connected to the input/output pins of the WS-cell. The connection of each net (xi) to the WS-cell specifies two bits of the key.

Step 2-1: The above process is repeated until all xi $\in$ sub-set are connected to the WS-cell.

Step 3: When the key of the WS-cell is generated, it is concatenated with other keys to make the whole key of the IP.

Step 4: Check if there is any net belong to cut-list that are not connected to a WS-cell.

As mentioned before, the circuit will be send incomplete to foundry and fab (without sending the key) in our proposed technique. Therefore, any key confidentiality management does not require. Key confidentiality is an important challenge in current IP/IC protection solutions. Therefore, a secure framework must be established to protect the key inside an IP/IC.

5 Analysis of Netlist Encryption Mechanism

Multiple attacks have been presented against logic obfuscation techniques with objective of key extraction. In all these attacks the attacker is looking for a way to get use of leaked information to minimize the search space in brute force attack. Although all these attacks are implemented to extract key in logic obfuscation technique proposed in [11], their strategies are applicable against any other obfuscation method as well. In this section, we analyze the strengths of netlist en-

cryption mechanism against these information leakage attacks.

5.1 Logic Cone Attack

The idea is that brute force can be applied first to the logic cone containing the fewest number of key inputs, (the least secure logic cone) then the secret key is recovered by employing brute force attack (all combinations of key inputs can be tried to identify for which combinations of values the simulated netlist output for that logic cone matches the functional IC output)[18]. The process is then repeated for the remaining logic cones in the circuit, sorted in an increasing order in the logic cone size. The existing overlaps among logic cones would provide more information and also help attacker to break the puzzle in smaller parts. However, in our approach the circuit is cut into two separate parts. It is worth noting that all paths from inputs to outputs are cut. It means that it is not possible to specify the size of logic cones and separate them to simulate individually (Figure 7). Also, the WS cells are inserted where the nets have minimum controllability and observability. Therefore, the logic value of wires in inputs and outputs of WS cells will be uncertain. The above explanations are illustrated in the Figure 7. In this figure the clouds represent sub-circuits and WS-cells show the wire scramblers inserted on cut wires.

In [18] the author has suggested to insert MUX gates in such a way to scramble the logic cones to improve the strength of logic obfuscation against this attack. These MUXes as explained in [11] are 2-input MUXes that are simulating XOR gates and two nets with almost opposite logic values should be connected to them. As shown in [11] it is not easy to guarantee there are enough nets with this condition in any circuit. However, this condition is not necessary for our proposed idea in this paper.

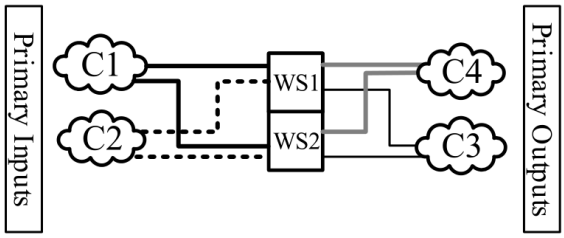

Figure 7. WS cell insertion in circuit topology

5.2 Boolean Satisfiability (SAT) Attack

In this attack the Boolean function of outputs are generated based on the key inputs that would be applied to them [19]. Also, it is assumed that the correct test vectors are available or attacker access to an activated IC to apply input patterns and observe outputs. In this attack, there are some assumption that are not valid in our proposed threat model.

(1) The key inputs are involved in the logic value of the output signals and key gates play the role of inventor or buffer. However, the WS cells associate all functions together. The Boolean function of each output net of WS cell is a combination of all input functions this fact leads to equations with many unknowns.
(2) In our threat model we have suggested to re-insert the WS cells in each tape out, it would prevent attackers to reach activated IC.

5.3 WS Removal Attack

In this kind of attack, adversary tries to remove WS-cells by applying random keys. Actually, this attack is not practicable in the proposed mechanism (netlist encryption). It means that WS cells cannot remove from the circuit except applying the correct key. The WS cells act somewhat similar to switch boxes in FPGAs. They are programmed with the key bits. As routing is incomplete without configuration of the switch boxes, in netlist encryption also the routing of the IC would be completed when the correct bits are applied to WS cells.

5.4 Retrieving Good Input-output Pairs

An attacker can buy a working IC from the market. Thereby, he will have access to good input-output pairs of the IC. This attack is more intelligent than brute force attack and would have smaller search space. The attacker also accesses to the layout information of a circuit, she can analyze the weak and strength points of the circuit (like where the keys are masked, in which sub-circuit a constant value is often generated and etc.). These analyses help attacker to select some special test vectors which lead to meaningful behavior in the circuit. Also she can suggest good points in the circuit that controlling their logic values (writing or reading) would help to discover useful information for key extraction. These special logic values are called good inputs-outputs pairs. Good inputs /outputs may include both primary and pseudo inputs/outputs (flip flops are also considered as pseudo inputs /outputs)[11]. However, when the IC's manufacturing process is finished the designer disables the scan test access port of the IC [20] so these internal flip flops are not available from outside. The activated IC act as a black box and just provides the primary input-output pairs. In this case retrieving good input-output pairs would downgrade to brute force attack.

5.5 Brute Force Attack

In this attack, an attacker tries all possible key combinations until he finds the correct key. In the experimental result section, a large number of input-output pairs are applied to the circuit. The pairs are selected and applied randomly. As mentioned before in manufacturing stage the attacker does not reach all input-output pairs of the circuit. Also, the post-wafer testing process would be done with incorrect key and their related test vectors. Therefore, not enough true input-output pairs are available to attack the circuit. However, this attack has been analyzed in the experimental results just to estimate the strength of the proposed mechanism (inserting WS cells on low controllable and low observable nets).

6 Experimental Results

We implemented the proposed wire scrambling algorithm in the EduCAD framework [21], a Linux-based educational physical design platform, on an Intel 2.7 GHZ Quad Core CPU with 4GB RAM. Eight circuits from IWLS [22] benchmark suite are selected to evaluate the proposed method. Table 2 represents the specifications of attempted benchmarks. In this table columns *#Cells*, *#Nets* and *#Pins* show the number of instances, nets and primary pins of the benchmarks, respectively.

As described before, a wire selection and WS-cell insertion algorithm is presented in this paper in which the suitable wires for obfuscation and the location of WS-cells are determined according to special metrics enumerated in Section 4. These metrics help us to obtain maximum obfuscation with minimum overhead. We have implemented a random WS-cell insertion algorithm to compare its results with those of intelligent algorithm, to estimate the effectiveness of the proposed approach in terms of benefit (output Hamming Distance) and cost (delay overhead). These experiments are presented in the following section.

Table 2. Statistical characteristics of attempted benchmarks

Benchmark	#cell	#Net	#pin
spi	1539	1637	92
des_area	1994	2234	304
mem_ctrl	389	426	32
ac97_ctrl	5503	5739	132
usb_funct	7042	7850	249
pci_bridge32	3112	3726	827
aes_core	13426	13775	388
wb_conmax	19098	20516	2546

6.1 Security Improvement

Security improvement in our proposed technique is equivalent to the statement that how much the reverse engineering process is hindered. The most popular method for this measurement is to calculate the size of search space. In other word it should be considered that there is a brute-force attacker who randomly applies keys and knows the expected outputs for his/her applied input test vectors. The difficulty level of his/her attack can be reported as the probability in which he/she would be able to find the correct key. This parameter is equal to *ideal probability of reverse engineering, IPRE*, which is calculated in Equation 2. In Table 3, this parameter is calculated for each benchmark circuit to show the immunity improvement of the attempted benchmarks against reverse engineering when their wires are scrambled with the proposed technique. In this table, column #WS-cell shows the number of WS-cells that are inserted and columns #Net and #DN represent the total number of nets and number of scrambled nets, respectively. Column SN% represents the percentage of scrambled nets and the last column is IPRE that represents ideal probability of reverse engineering.

Table 3 shows that almost less than 10% nets of a design is required to be scrambled to make a great search space against adversaries. In other word, brute-force attack for finding secrete key is not practical and the probability of finding secret key is almost zero. However, there may be some situations that despite applying wrong key, correct values are generated at output vectors. These cases help reverse engineers to learn about circuit's functionality and exclude some input patterns in the brute-force attack to compact the search space. We count this kind of records and report them as *learned information*. Another factor that is suitable for evaluating the security level is the number of deviated bits from original output bits when wrong-key is applied (output hamming distance).

We designed an experiment to show how much information may be extracted from output pins to bypass the brute force attack. In this experiment, we simulated a circuit with large number of random input vectors and calculated the learned information and Hamming distance. We repeated this experiment with different wrong-keys vary in wrong bits from 60%-90%. It is worth noting that there would be no considerable chance for reverse engineers when learned information is zero or/and output Hamming distance is close to 50%. Table 4 shows the experiments results for SPI benchmark. Each row of this table is dedicated to a range of incorrect key bits. Learned information and HD are calculated for these ranges.

As can be seen in Table 4, hamming distance of

Table 3. Experimental results in terms of immunity improvement

Benchmark	#WS-cell	#Net	#DN	SN(%)	IPRE
spi	39	1637	156	10%	1.20e-94
des_area	31	2234	124	6%	2.21e-75
mem_ctrl	5	426	20	5%	9.09e-13
ac97_ctrl	173	5739	692	12%	2.37e-417
usb_funct	217	7850	868	11%	2.58e-523
pci_bridge32	126	3726	504	14%	3.65e-304
aes_core	279	13775	1116	8%	1.26e-672
wb_conmax	380	20516	1520	7%	7.39e-916

Table 4. Experimental results in terms of resilient to attacks

Incorrect Bits of Key	Learned Information	Output HD (incorrect bit #)
60%-70%	0%	32%
70%-80%	0%	39.1%
80%-90%	0%	40.8%

outputs is more than 40% when most bits of the applied key are incorrect. Moreover, the learned information is absolutely zero. In this situation, percentage of leaked information to output is zero, practically.

An important point is that the output Hamming distance can be improved (closer to 50%) with more of inserted WS-cells. Therefore, designers can improve the immunity of the design based on the required level of security in their application. Moreover, the Hamming distance will be closer to 50% for larger circuits because they have longer paths with more number of WS-cells. Consequently, the proposed technique makes a scalable and configurable solution for designers to make a reasonable trade-off between trust and cost.

In the best case, the output Hamming distance in in random Ws-cell insertion algorithm is 30% while the Hamming distance in intelligent algorithm is about 40%. This fact shows that the security level obtained from random algorithm is less than the security level through intelligent algorithm with a same amount of WS-cells inserted in the circuit.

6.2 Overheads

As mentioned before, protecting the design against reverse engineering is possible in cost of area, power and total wire length of the design. Table 5 shows the experimental results in terms of the overheads. In this table, columns Area, THPWL and Delay represent the overheads before and after the wire scrambling methodology, respectively. Finally, columns WO, DO and AO represent wire length, delay and area overheads in percentage.

As can be seen in this table, overheads are not significant and the proposed method is completely reasonable for regular circuits. We have mentioned before that we avoid inserting WS-cells on critical nets to prevent performance drop. Therefore, the overhead of delay is negligible. It is expected that adding the WS-cells increases the area, total wire length and delay of the benchmarks. However, the placement phase is based on a stochastic algorithm and the results may be improved after placement of the circuit comprising of the WS-cells. In some rows, delay overheads are negative that shows the placement results in that run are better and overhead amounts are less than improvement margin of placement phase. It also confirms that the WS-cells are not on critical path.

Table 6 shows the comparison between random and intelligent algorithm in delay overhead. In these two implementation for each test bench the number of WS-cells are equal; however, the location of the WS-cells in the netlist is different. Since WS-cells are not inserted on the critical path, delay overhead is expected to be the same in these two implementation. Nonetheless, WS-cells inserted in random implementation may be located back-to-back in a same path and increase the path delay. The delay of entire chip would be influenced if the altered path is a near critical path. As shown in Table 6 with the same number of WS-cells, delay overhead in random implementation would be increased more than twice, compared with intelligent implementation. In this table, column *Befor SP* represents the delay of the circuit before any WS-cell is inserted and columns *DIn* and *DR* show delay of the design when WS-cells are inserted through intelligent and random algorithm, respectively. Finally, columns *In-R* and *R-BSP* indicate delay differences of random implementation with intelligent implementation and original circuit, respectively.

Table 5. Experimental results in terms of overheads

	TWL (μm)		**Delay**(ps)		**Area**μm^2		**WO (%)**	**DO (%)**	**AO (%)**
	Before	After	Before	After	Before	After			
spi	1.7e8	1.8e8	3.11e3	3.03e3	4.75e04	5.75e04	6.00	-3.00	21
des_area	1.96e8	2.20e8	3.74e3	3.65e3	4.75e04	5.75e04	12.24	-2.00	21
mem_ctrl	4.13e7	4.33e7	8.85e2	8.84e2	1.68e04	1.83e04	5.00	0.00	9
ac97_ctrl	9.1e8	9.96e8	2.50e3	2.82e3	2.50e05	3.05e05	9.00	13	22
usb_funct	1.18e9	1.38e9	5.30e3	5.57e3	2.35e05	3.00e05	17.00	5.00	28
pci_bridge32	5.18e8	5.84e8	5.64e3	5.71e3	1.93e05	2.35e05	13.00	1.00	22
aes_core	1.96e9	2.70e9	2.59e3	2.94e3	2.90e05	3.78e05	37	14.00	30
wb_conmax	2.68e9	3.13e9	4.92e3	5.02e3	3.95e05	5.18e05	17.00	-2.00	31
Average							**14.5%**	**3.25%**	**23%**

Table 6. Compare delay overhead in random and intelligent algorithm implementation

	Before SP	**After SP**		**Delta**	
		DIn	DR	In-R%	R-BSP%
spi	3.11e03	3.03e03	3.03e03	0	-3.00
des_area	3.74e03	3.65e03	3.85e03	5.45	2.90
mem_ctrl	8.85e02	8.84e02	9.62e02	8.86	8.74
ac97_ctrl	2.50e03	2.82e03	2.82e03	0	13
usb_funct	5.30e03	5.57e03	6.24e03	12.04	17.74
pci_bridge32	5.64e03	5.71e03	5.79e03	1.42	2.68
aes_core	2.59e03	2.94e03	2.94e03	0	14.00
wb_conmax	4.92e03	5.02e03	5.02e3	0	-2.00
Average				**3.5%**	**7.13%**

7 Conclusion and Future Work

We proposed a mechanism for hindering the reverse engineering in automatic ASIC design flow. We designed special standard cells (Wire Scrambling cells) to scramble the design netlist and then, proposed a physical design methodology in which wiring topology of the circuit is scrambled automatically using the wire scrambling cells. The main feature of this mechanism is that it can be performed without detailed information about the functionality and structure of the netlist and can be automated easily. The proposed technique allows the designers to make a reasonable trade-off between trust and cost.

References

[1] V. Leest and P. Tuyls, "Anti-counterfeiting with hardware intrinsic security, "In *Proceedings of the Conference on Design, Automation and Test in Europe*, pp.1137- 1142, 2013.

[2] R.S. Chakraborty, S. Paul, and S. Bhunia, "On-Demand transparency for improving hardware Trojan detect-ability, "In *Proceedings of international Workshop Hardware-Oriented Security and Trust (HOST 08)*, pp.48-50, 2008.

[3] S. Bhunia, M.S. Hsiao, M. Banga and S. Narashimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures, " *In Proceedings of IEEE, Vol.102, No.8*, pp.1129-1247, 2014.

[4] R.S. Chakraborty, S. Bhunia, "Security Against Hardware Trojan Attacks Using Key-Based Design Obfuscation," In *J. Electron Test*, Vol.27, No.6, pp.767-785, 2011.

[5] R.S. Chakraborty and S. Bhunia, "Hardware Protection and Authentication through Netlist Level Obfuscation," In *International Conference on Computer Aided Design, ICCAD 2008*, IEEE/ACM, pp.674-677, 2008.

[6] R.S. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits*

and Systems, TCAD 28(10), pp. 1493-1502, 2009.

[7] Y. Alkabani, F. koushanfar, M.Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," In *International conference on computer Aided Design, ICCAD 2007*, IEEE, pp.674-677, 2007.

[8] J. Rajendran, A. Ali, O. Sinanoglu and R. Karri, "Belling the CAD: Toward Security-Centric Electronic System Design," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 11*, pp. 1756-1769, Nov. 2015.

[9] J.A Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," In *proceedings of the Design, Automation and Test in Europe. DATE'08*, pp.1069-1074, 2008.

[10] M. Yasin; J. Rajendran; O. Sinanoglu; R. Karri, "On Improving the Security of Logic Locking," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.PP, no.99*, pp.1-1, 2015.

[11] J. Rajendran, Y. Pino, O. Sinanoglu and R. Karri, "Fault analysis-based logic encryption," In *IEEE Transactions on computers*, Vol.64, No.2, pp.410-424, 2015.

[12] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre, "A Novel Hardware Logic Encryption Technique for Thwarting Illegal Overproduction and Hardware Trojans," in Proc. *IEEE International On-Line Testing Symposium*, 2014, pp. 49-54.

[13] A. Baumgarten, A.Tyagi and J.Zambreno, "Preventing IC piracy using reconfigurable logic barriers, "In *IEEE Design and Test of Computers*, Vol.27, No.1, pp.66-75, 2010.

[14] S. Zamanzadeh and A. Jahanian. "Automatic netlist scrambling methodology in ASIC design flow to hinder the reverse engineering," In *VLSI-SoC, 2013 IFIP/IEEE 21st International Conference on*, pp. 52-53, 2013.

[15] S. Khaleghi, Kai Da Zhao and Wenjing Rao, "IC Piracy prevention via Design Withholding and Entanglement," *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, Chiba, 2015, pp. 821-826.

[16] Xilinx Power Estimator User Guide, UG440 (v2.0) May 4, 2009. Available on: http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug440.pdf

[17] Z. Navabi, Digital System Test and Testable Design: Using HDL Models and Architectures, Springer, 2010.

[18] Y.W. Lee and N. Touba, "Improving Logic Obfuscation via Logic Cone professor of electrical and computer engineering at Analysis," in Proc. *Latin-American Test Symposium*, 2015, pp. 1-6.

[19] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the Security of Logic Encryption Algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2015, pp. 137-143.

[20] ARM.(2010).i.MX35 applications processors for industrial and consumer products [Online]. Available: http://www.freescale.com/webapp/sps/site/\prod_summary.jsp?code=i.MX357&fpsp=1&tab= Documentation_Tab

[21] S. Amanollahi and A. Jahanian, "EduCAD: an efficient, flexible and easily revisable physical design tool for educational purposes," In *proceedings of the Design, Automation and Test in Europe*, DATE 2011.

[22] IWLS 2005 benchmarks, [online] Available on: http://www.iwls.org/iwls2005, 2005.

Sharareh Zamanzadeh received her B.S. degree in computer engineering from Islamic Azad University, south Tehran Branch, Tehran, Iran in 2004, and the M.Sc. degree in computer System Architecture from Amirkabir University of Technology, Tehran, Iran in 2008. She is currently a Ph.D student in computer system architecture at Shahid Beheshti University. Her research interests are hardware security and VLSI computer aided design.

Ali Jahanian received his B.S. degree in computer engineering from University of Tehran, Tehran, Iran in 1996 and the M.S. and Ph.D. degrees in computer engineering from Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His research interests are VLSI design automation, hardware security and secure chip design.