The ISC Int'l Journal of Information Security

ISeCure

January 2021, Volume 13, Number 1 (pp. 47–57)

http://www.isecure-journal.org

Hardware Trojan Prevention and Detection by Filling Unused Space Using Shift registers, Gate-chain and Extra Routing

Mansoureh Labafniya [1], Shahram Etemadi Borujeni [1,*], and Roghayeh Saeidi [2]

[1] *Faculty of computer engineering, University of Isfahan, Isfahan, Iran*

[2] *Faculty of ICT security Department, ICT research Institute, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: January 10, 2020
Revised: May 30, 2020
Accepted: October 10, 2020
Published Online: October 19, 2020

Keywords:
Controllable Point, Field Programmable Gate Array, Hardware Trojan Horses, Observable Points, Security

Type: Research Article

doi: 10.22042/isecure.2020.215265.510

Abstract

Nowadays the security of the design is so important because of the different available attacks to the system. the main aim of this paper is to improve the security of the circuit design implemented on FPGA device. Two approaches are proposed for this purpose. The first is to fill out empty space using flip-flops and LUTs so that there is no available space for inserting a hardware Trojan. We name this filling structure as Gate-chain. The second approach increases the security of the implemented design by identifying the low observable/controllable points of the main design and wiring them to the unused ports or the pre-designed Gate-chains. The proposed solutions not only prevent Trojan insertion but also increase the Trojan detection capabilities. Simulation results on Xilinx devices implementing different benchmarks show that the proposed method incurs dynamic power overhead just in test mode with less than one percent of delay overhead for critical path in normal mode.

1 Introduction

Current Field programmable gate array (FPGA) devices are well suited for a wide variety of applications, ranging from data processing and cryptography to communication and industrial automation applications. Using FPGA platform for the implementation of the Internet of Thing (IoT) applications is a hot topic, which has attracted much attention in recent years [1]. The security of IoT devices is one of the main issues that need to be concerned with spreading IoT applications. Therefore, recent literature focuses on securing FPGA-based platform.

To analyze the hardware security of FPGA device, we need to consider its whole lifecycle from design concept to validation testing. This lifecycle contains two main flows: manufacturing and application design flow. The manufacturing flow of the FPGA devices is the same as the IC manufacturing of any other semiconductor devices. The globalization of this flow for lower costs and gaining more access to the latest technologies has significantly increased the probability of hardware Trojan (HT) attack. Hardware Trojan is a malicious modification of ICs for leaking secret information or malfunctioning of the final application. These type of attacks are not limited to manufacturing design flow but also can threaten the application design flow. In application design flow, a designer uses FPGA vendor implementation tools, libraries, and the hardware configuration file to generate the bitstream file. This file can be accessed and modified for partial reconfiguration in order to save cost and decrease the time to market of products [2]. In addition, this reconfiguration has two security aspects, one for attackers and one for a wise designer. An

* Corresponding author.
Email addresses: mlabaf@eng.ui.ac.ir, etemadi@eng.ui.ac.ir, r.saeidi@itrc.ac.ir.com
ISSN: 2008-2045

attacker can manipulate the bitstream in a stealthy manner for malicious modification. It is not possible for the foundry or design house to predict this type of attack and prevent it [3]. On the other hand, a wise designer can use this flexibility as an opportunity to provide more security features against hardware Trojan attack. Similar to the first aspect of the security, since these features are added in the application design phase, the end user can bypass the potential threat. These security features can also be used for hardware Trojan detection and prevention, which is the main aim of this paper.

The generated bitstream in application design flow is either programmed directly in antifuses and flash memory cells or loaded in external non-volatile memory (in SRAM based FPGAs). The later one needs a communication channel for programming the FPGA, which can be used by an attacker to leak bitstream information [4]. The bitstream contains all information of a design, for example, the list of used resources and how they are connected and configured. In this way, the attacker can launch a more ingenious attack, which is not comparable with those that can be made in Application Specific Integrated Circuits (ASICs). Therefore, the bitstream opens a new battlefront in the circuit design area, and its influence range is increasing by spreading IoT technologies. This security issue has promoted the researchers to work on HTHs (Hardware Trojan Horses) prevention and detection methods [5–9]. In this paper, we strengthen the secure design in [10] which was secured against hardware Trojan horses inserted in unused space by filling them with change-aware gates. Paper [10] just considered the Trojans which are inserted in unused space but in this paper we consider all types of HTHs which are inserted in unused space or change the content of used resources to be implemented. We analyze the used resources to explore the critical points and implement certain circuits in free spaces to improve the controllability and observability of these points. Same as [10], we have two functional modes for design, including normal mode, in which just the main design is active and test mode, in which the filled unused space is active. Since the added gates to the main design are only activated during the test mode, they incur low energy overhead. Moreover, due to using design checkpoint and Tool Commend Language (TCL) instruction, the critical path of the main design will change with overhead less than %1. The contribution of this paper can be summarized as follows: 1. Prevention of hardware Trojan insertion by manipulating the LUTs of the main circuit and unused resources. 2. Detection of hardware Trojan insertion in vulnerable (critical) points of the main design and unused resources. The rest of the paper is organized as follows. In Section 2, we review the related work in recent years. Section 3 describes the proposed method for securing the FPGA design against HTHs insertion. In Section 4, we present the simulation results and benefits of our scheme; and finally, Section 5 concludes the paper.

2 Related Work

Hardware Trojans are malicious circuits that are inserted into the system to change the performance or functionality of the design. Sometimes this malicious circuit is designed for information leakage without modifying the system specification. Various methods have been proposed recently to detect or prevent the insertion of a hardware Trojan. There are papers that propose methods to prevent attacking the system. Some are based on some kind of obfuscation. In this technique, the design is obscure, and therefore the attacker cannot analyze the circuit to manipulate the design. Moreover, eavesdropping may not be effective because the validity of the outputs depends on the correct key insertion, which is hidden from the attacker [2]. Using evolutionary algorithm(EA) can be helpful to prevent and detect HTHs in circuit [11]. Balancing the activity of the different regions of the design and increasing the testability of internal nodes are effective methods for securing design against power analyzing attacks and the attacks that use the rare event for triggering the hardware Trojan. These methods are named logic encryption [12]. The suggested methods in [13] provide secure design methods to increase the cost of attack by inserting security-path in FPGAs. When the attack is discovered by security-paths and inserted XOR gate, the normal data flow is obfuscated and the circuit is blocked. Trojan detection methods can be categorized into three groups: visual detection methods, side-channel analysis, and logical-based testing. Observation-based methods examine various snapshots of the design, for example, X-ray imaging and microscope scanning, to detect suspicious region for Trojan. These expensive methods are time-consuming and not sufficiently precise [14, 15] to detect expertly inserted Trojans. The second type of detection methods analyzes the output parameters of the circuit and their changes to detect the unusual behavior of the circuit. These parameters include power consumption, supply current, voltage variation, path delay, and electromagnetic radiation [9, 15]. In paper [16][16], a new DFT methodology is proposed in which special test-vector generated to rise the transition of low controllable nets to the Maximum Transition Probability (MTP) and increase the full activation time and side channel sensitivity for both logic testing and side channel detection approaches. In the third detection method, the

existence of a Trojan is examined by giving different input patterns, checking the output, and comparing the generated outputs with the expected outputs [17]. Paper [18], proposed reconfigurable assertion checkers (RACs) for hardware Trojan detection on a system-on-chip (SoC). In this paper, flexible RAC architecture is developed to implement different types of assertions. The methods mentioned above can apply to both FPGA and ASIC designs. Since the final design on FPGA is completed by the bitstream, several methods have been proposed to secure design in this last step, such as ambiguity or use of the Adapted Triple Modular Redundancy (ATMR) structure [15]. Despite the popularity of these methods for securing the FPGA designs, the main challenge of implemented circuits on FPGA is the unused free space. In order to have an optimal routing and placement during FPGA programming process, the designers usually use only %60 of the available resources in the FPGA platform. This unused space introduces security vulnerability in the system. The proposed method in [19] mitigates the vulnerability of the unused spaces by filling them with Ring Oscillator(RO). The frequency of these RO modules depend on their location, length, and circuit characteristics; therefore, adding or modifying anything near their location may lead to a frequency deviation. The designer can use this sensitivity to detect a Trojan. This structure detect insertion of HTHs and does not fill all empty spaces, and finally cannot prevent the insertion of hardware Trojan. In [20], all unused space is filled with the Built-in self-authentication (BISA) structure and appropriate modules to fit in empty space. Using this structure, the signature is generated in the output of the filled sections, in which, inserting any Trojan will change this signature. Because of the certain structure of the BISA, this signature is constant for each test pattern; therefore, an attacker can deliberately insert a Trojan in unused space and regenerate the expected signature. This BISA structure cannot protect the LUTs of the main circuit against Trojan insertion and only fills out the empty space. The proposed method in [21] attempts to fill the unused space of an ASIC design by using shift registers and appropriate modules to fit in the empty space. The output of the added circuits depends on the input of the shifter registers. This method provides a better security feature for preventing Trojan insertion because it uses the shift register structure rather than the BISA structure, which can be threatened by a structure-aware attacker. Nevertheless, this article is about preventing and detecting Trojan insertion in the empty space and does not provide any capability for securing the main design against Trojan insertion The proposed method in [22] fills the unused space with constant data information to prevent Trojan insertion. To this end, these codes are added to the Xilinx Design Language(XDL) output file, which is generated during synthesis and implementation steps to completely consume all resources of the FPFA. However, the attacker can distinguish the main design from the manually filled space by using reverse engineering tools and analyzing fixed-value cells. These tools extract the bitstream from the external memory of the FPGA and utilize it to recover the XDL or netlist file [23, 24]. In this way, the dummy cells can be replaced by the desired hardware Trojan without changing the main circuit. Moreover, author in [22][23] does not propose any solution to prevent tampering of fixed-value information circuit or detect hardware Trojan insertion in the main design space. In addition to these issues, working with the XDL file is not user-friendly as the Verilog coding.

Filling unused space is a common method to prevent and sometimes detect the hardware Trojan insertion in the implemented design. In the previous methods, after filling unused space, the design is still vulnerable against HTHs insertion in the main design or unused space without any feedback to detect them. In this paper we improve our secure design in [10] by suggesting the design against all HTHs inserted in unused space and used space in main design.

We update the security of the design in FPGA by combining two methods, which include both removing unsecure points in main design and filling unused space, but [10] and [22] just secure the design by filling unused space.

In the next Section, we will present a new scheme for prevention and detection of HTHs in both original design and unused spaces of FPGAs without any alteration in main design parameters like delay and power consumption. The proposed structure uses a kind of modified Ring Oscillator(RO), which increases the prevention and detection capability of the design. Table (1) shows the comparison of the proposed method by the other related explained papers.

3 The Proposed Scheme for Hardware Trojan Prevention and Detection

Look Up Tables (LUTs) and Flip-Flops (FFs) are the most common types of resources on FPGAs. Although today's FPGAs have other more specialized programmable function units, in this article we only focus on securing the LUT and FF resources, which are the fundamental reconfigurable blocks of the design. If the designer can use all LUTs and FFs resources on FPGA to implement the circuit, no free space is left for an attacker to insert the desired Trojan. In this case, the attacker may change the func-

Table 1. Comparison of the different methods for prevention/detection of HTH insertion by filling unused space

method	hardware	pros	cons
[19]	FPGA	-Prevention by partial filling of unused space -Degradation of main design parameters	-Sensitive to process variation
[20]	ASIC	-Prevention by filling all unused space	-No detection with weak feedback
		-No degradation to main design parameters	-No detection of HTH insertion by main design modification
[21]	ASIC	-Prevention by filling all unused space	-No detection of HTH insertion by main design modification
		-Detection by strong feedback	
		-No degradation to main design parameters	
[22]	FPGA	-Prevention by filling all unused space	-No detection because of no feedback
		-No degradation to main design parameters	-No detection of HTH insertion by main design modification
[10]	FPGA	-Prevention by filling all unused space	-No detection of HTHs insertion by main design modification
		-Detection by strong feedback	
		-No degradation to main design parameters	
		-user friendly	
proposed method	FPGA	-user friendly	
		-No degradation to main design	- nothing-
		-Detection by strong feedback	
		-Prevention by routing unsecured points of main design to chain or unused port	
		-Prevention by filling all unused space	

tionality of the used LUTs for a denial-of-service attack. To prevent this type of attack, the designer should analyse the internal nodes of the design to identify and secure the potential sites of attack. In this Section, we explain the proposed method to detect and prevent the insertion of these two types of HTHs: manipulating the used resources of the main design or exploiting the unused LUTs and FFs. This proposed method contains two phases to secure the FPGA. In the first phase, we fill out all unused LUTs and FFs, similar to our contribution in [10] to prevent HTHs in empty space. In the second phase, we focus on securing the main design and mitigate its vulnerabilities by increasing the testability of the critical points, which is our continuation in this paper. In the proposed scheme, we also utilize the unused spaces to improve the testability of the unsecured points in the main design and hence decrease the probability of HTHs insertion. The number of possible wires and routing between logic resources is much more than the number of logic resources. Despite that, if there are some logic resources without any possible routing, it is also a limitation for the attacker to rout and implement the HTHs properly in design. This limitation is worse for the attacker because he/she needs to insert the Trojan stealthily. Fig. (1) shows the flow diagram of the proposed scheme in six steps, which is described as follows.

3.1 Step One

The first step is to synthesize the main design, which is still unprotected. Next step is finding the unsecured points of the implemented design. One kind of the unsecured potential sites for hardware attacks and insertion of Functional Trojans is low observable/controllable points in the main design. The points near the primary outputs have high observability, and the points near the primary inputs have high controllability. Our criterion for selecting ponts for securing is based on paper [6] in which the points in the middle of the paths are far from the input/output ports, so they have low testability (i.e., low observability/controllability) especially on the long path. Moreover, the other parameters like slack time, rare signal or trigger probability of the

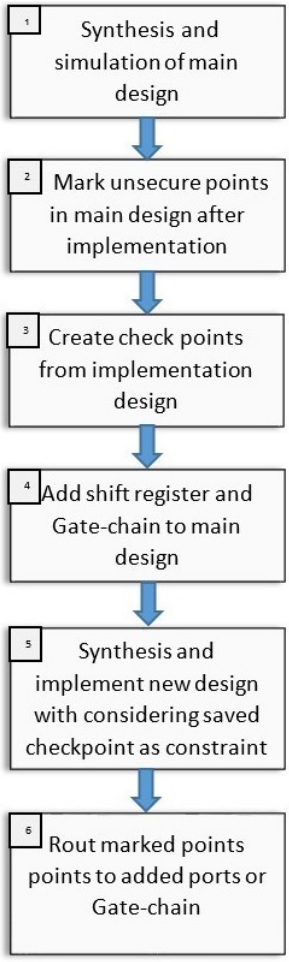

Figure 1. Flow diagram of the proposed scheme for HTHs prevention and detection in FPGAs

point are usually used for determining these critical points [5, 25–27]. It is noteworthy that increasing testability of these points improves the security of the whole system. In this step designer can select the unsecured points according to his/her goal.

3.2 Step Two

In the second step, we consider different paths in the main design, which are reported by Vivado after implementation process. We used Vivado to mark the middle of the longest paths in the schematic view of the design to secure them in the next steps of the proposed method. The number of marked points depends on the level of security that the designer determines.

3.3 Step Three

In the third step, we save current "place and route" by using "write-checkpoint" instruction in TCL section in Vivado. Using this design checkpoint as a constraint, Vivado saves all critical paths and specification of the main design.

3.4 Step Four

After implementing the original circuit, a number of LUTs and FFs are left unused on FPGA. In fourth step, we fill out these resources with a chain of logic gates and shift registers same as [10]. First, we utilize all LUTs and a number of FFs to make the chains. In the proposed method, LUTs have higher priority than the FFs, because they are the functional units of the FPGA and the adversary needs to exploit them to run an attack. Since the number of FFs are usually more than the number of LUTs, some FFs may remain unused. Therefore, we utilize them in shift register structures to make sure there is no free space for Trojan insertion. Fig. (2) shows how to use the mentioned resources in each structure, in which the Gate-chain structures contain FFs and LUTs and the shift registers only consist of FFs. As mentioned in [10], Gate-chain can consist of different and random type of gates with different feedbacks from other parts of the chain. This capability improves the strength of output signature in comparison with the NOT-chain that only uses the NOT gates.

3.5 Step Five

In the fifth step, after inserting Gate-chains and shift registers in the HDL codes of the main design, the EDIF file is generated from the final implementation. In this step, the implementation setting must be configured using the saved checkpoint in the third step. This scheme can prevent HTHs insertion (e.g., adding extra gates to design) as no free space is left on FPGA.

3.6 Step Six

In the last step, we route the marked unsecured points to the unused ports. Since the number of unused I/O ports is limited, we can route the remaining unsecured points to the inputs of the gates in Gate-chain. Both routing methods increase the controllability/observability and thus monitoring capability of the points. Fig. (2) shows the schematic view of this protection scheme. Steps 2 and 6 are the added steps by our proposed method in this paper in comparison to [10]. It is noteworthy to mention that defining new ports in the sixth step can be done using TCL command "create-(port/pin/net)". Routing unsecured points to unused ports or to the Gate-chain is available by TCL command "connect-(port/net)". These instructions directly change the final netlist to secure the design. The total number of routing from unsecured points to unused ports or Gate-chain is flexible.

The proposed design has two operating modes: normal mode and test mode. In normal mode, only the

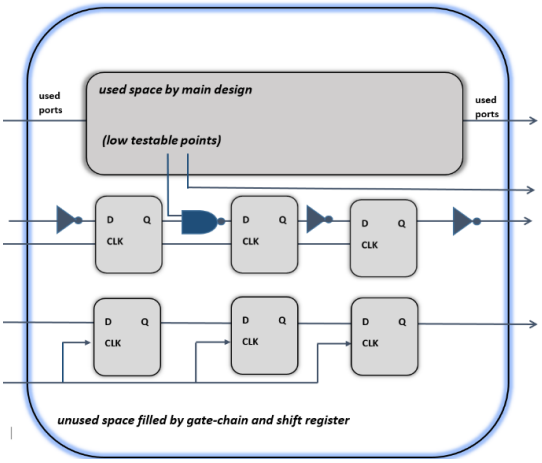

Figure 2. Schematic view of the proposed method

original design is executing, and its clock entry is activated. In this way, the design parameters will remain unchanged and the circuit is functioning in normal operation except the delay overhead of the critical path which is increased less than %1. In test mode, only the shift register and the Gate-chain are enabled. In addition, some inputs of the main design must be active for enabling and testing the routed points to Gate-chain. In this mode, we can check if any HTH is inserted in unused space, which is now filled by Gate-chain. This mode has also the capability of the HTH detection in main design by monitoring the low testable points.

4 Simulation of the Proposed Method: Attack and Results

In this section, we explain the simulation results of our proposed method in the described phases. We synthesize and implement the proposed design using Vivado design suit 2018.2. AES, DES and SHA cryptography codes from Cookbook benchmark [28] and s38417 code from ISCAS98 benchmark are used as samples of high-performance computing algorithms. I2C and SPI are used from the IWLS benchmark circuit as samples of simple code [29] to show the efficiency of the proposed method for both small and large code sizes. After implementing s38417 code in Fig. (3), a checkpoint is saved from the design. Remaining unused LUTs and FFs specify the size of the Gate-chain and the shift register. The HDL code of these two structures is added to the main code to fill all resources. Fig. (4) shows the implementation result of s38417 code on FPGA after the first phase security procedure (i.e., filling the unused space). As can be seen in this figure, all free resources of the FPGA is completely filled using the proposed method.

Since we use checkpoint to save the constraint of the main design, adding shift register and Gate-chain

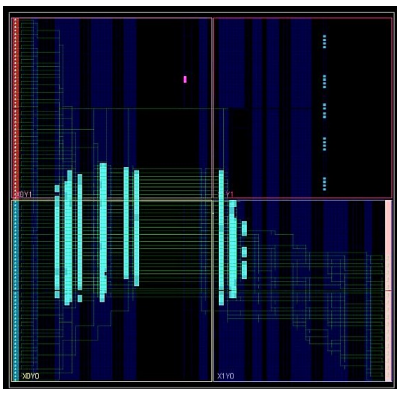

Figure 3. Resource utilization in s38417 design before filling the unused space

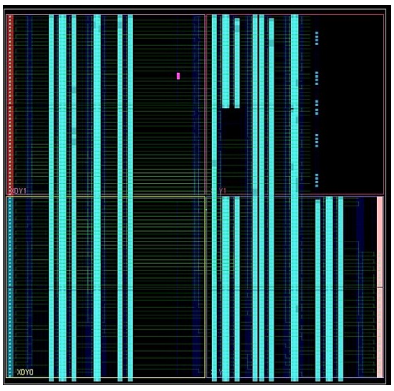

Figure 4. Resource utilization in s38417 design after filling the unused space

will not modify the parameters of the main design. As mentioned before, the proposed method has two operation modes : normal mode and test mode which should be considered for circuit implementation. For this purpose, the Clock input of the main design and the Clock input of the added design should never pulse simultaneously. In normal mode, only the main design is clocked and in test mode, just the Gate-chain

and the shift register are clocked. We consider two different time constraints for implementing the main design and added structure, which are equal to (T=10ns) and (T=2ns), respectively. Table (2) shows the implementation results of the first phase of the implemented secure procedure on xa7a15tcpg236-2I or xa7a12tcsg325-2I device. According to these results, we can utilize all resources to protect the system against HTHs insertion which is also verified in [10].

In FPGAs, static power is always fixed in FPGA, but dynamic power depends on the number of resource consumption, clock period, and switching activity [22, 30]. Table (2) shows the usability of our method for different design sizes. The obtained results in Table (2) confirms this fact, in which static power is fixed and dynamic power is varied for each design.

Table 2. simulation results

	unprotected design				protected design					
benchmark	used	used	used	energy	used	used	energy in normal mode	energy in test mode	dyn Power (unprotected)/(protected)	
	I/O	LUT	FF	dyn+static	FF	LUT	dyn+static	dyn+static	overhead	dyn energy
SHA	69%	19%	10%	0.06+0.008	100%	100%	0.06+0.008	0.06+0.08	0.79	10
AES	67%	42%	19%	0.06+0.02	100%	100%	0.06+0.02	0.06+0.09	0.07	5
DES	77%	17%	10%	0.06+0.14	100%	100%	0.06+0.14	0.06+0.18	0.04	1.5
s38417	90%	26%	10%	0.06+0.07	100%	100%	0.06+0.07	0.06+0.08	0.01	1.5
SPI	43%	5%	1%	0.07+0.06	100%	100%	0.06+0.06	0.06+0.62	0.6	10

The power consumption of the protected design in normal mode is same as the power consumption of the unprotected design because in both of them the same design (i.e., main design) is running. The measured power consumption of the protected design in test mode indicates that increasing the number of resource consumption leads to the increment of the dynamic power consumption. In this mode, the rate of increasing power consumption of the small circuit is higher than the large circuit as it has more unused LUTs and FFs, which are supposed to be filled to secure the design. The overhead of dynamic power is from 0.02w to 0.7w which is more than %50 in some cases. We compare the delay of critical path before and after step6 in order that less than %1 overhead was detected in it.

Fig. (5) illustrates the schematic view of the proposed implemention method, which consists of LUTs and FFs in [10]. Each gate in the chain is implemented by one LUT. After filling all resources of the FPGA and considering the saved design checkpoints in implementation, in the second phase we can route insecure points of the main design to the unused ports/Gate-chain. After completing the routing process, all changes will be saved as a new EDIF file. Fig. (6) shows the routing of a selected insecure point in SPI schematic to one of the unused port which is our suggested method in this paper.

The longest paths are the ones with the longest delay, which are reported in implementation stage. We imagined all the modules in all paths have the same delay for LUTs and logic levels. According to the desired level of security, we can start with the paths with higher delay values to secure them and then move to the faster paths. In each selected path, we can rout a signal to a pin or instead we can select several signals as a byte or word and rout them to a port .

Using TCL command, one of the unused port is selected, which is defined as "x" as shown in Fig. (6). The Net in the middle of the selected path is routed to this defined port. A Net is a logical connection between two or more symbol instance pins in FPGAs.

4.1 Comparison

The difference between our proposed method with paper [22] and [10] are summarized as follows:

- Paper [22] presented a HTH prevention method by filling all unused space with fixed and random values, but our proposed method provides both prevention and detection mechanisms.
- [22] and [10] methods protect the design against malicious object insertion in unused resources of the device and cannot be applied to secure system against other types of attacks, especially the malicious modification made in the main design. But in this paper we secure both unused space and used space against any HTHs insertion.
- Paper [22] uses a fixed value for dummy cells (i.e., FFs and LUTs), which are always active. However, in our proposed method, we inactivate the gate-chains when the main design is in operation mode, and activate it in secure mode.
- Paper [22] secures the design by manipulating XDL file but our method adds HDL code to the main code and uses TCL command which are more user friendly.

The similarity between our suggested method with paper [10] and paper [22] is as follows.

- All of the schemes have no effect on the static power consumption. Our simulation results in Table (2) confirms this fact. However, our method has dynamic power consumption overhead during test mode, which is independent of the main design same as paper [10].
- All mentioned papers can fill all LUTs and FFs successfully.
- All mentioned papers prevent HTHs insertion in unused space.

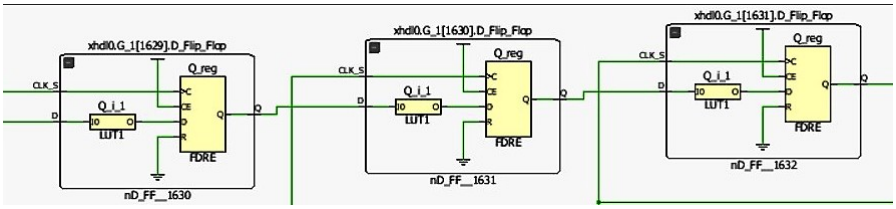

Figure 5. Added Gate-chain to design

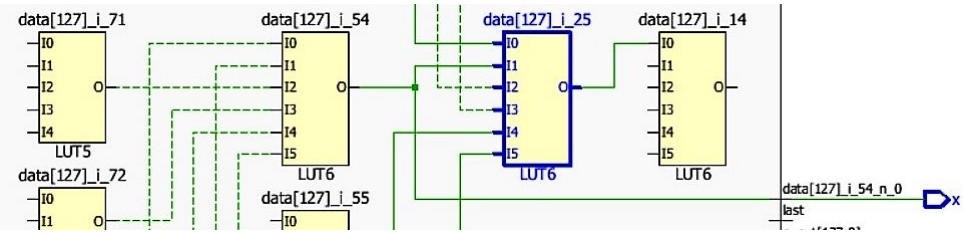

Figure 6. The insecure point in the main design, which is routed to an unused port

4.2 Attack Analysis

In this paper, we assume that attacker has physical access to FPGA. Therefore, attacker can partially reconfigure or manipulate bitstream of FPGA. The proposed method covers all types of HTHs and secures the design against the adversaries who attack by adding extra gates or changing the type of existing gates. For the unused space, which are filled as mentioned in [10], our method detects HTHs with their effect on power consumption, output signature, or response time. Variation of clock cycles and output signature of Gate-chain can reveal the existence of the small Trojans; while a large size Trojan can be detected using all three mentioned parameters. The security of the Gate-chain can be improved using different kinds of logic gates or adding combinational logic circuit, nevertheless using just NOT gate is more straightforward. For securing the used space, sensitive points of the main design are routed to the ports or Gate-chain to detect all HTHs that change the functionality of these points. SPI code from IWLS benchmark is used for simulating different attacks and observing the system response. We added a gate-chain using 4910 FFs and 4910 logic gates with two different implementations. In the first implementation, Gate-chain is made of only NOT gates; and in the second implementation, different types of gates are used. As mentioned before, remaining FFs are consumed with a shift register. To launch a successful attack, an attacker needs to insert a malicious design stealthily. In addition to the fact that it is hard and time-consuming for the attacker to reverse engineer the bitstream [24, 25], finding free space with improved testability is hard. In spite of that, if an adversary succeeds to attack, three potential HTHs can be inserted on the main design and Gate-chain: (1) removal attack, (2) redesign attack in main design, and (3) redesign attack in Gate-chain. 1) Removal attack: Changing the time delay between the input and output response of the Gate-chain can indicate the adversary may omit some FFs or LUTs of the design. For SPI code in secure mode, this time delay is 2.13 ms. If an attacker omits 10k FFs from the shift register, the time delay will decrease to 1.1 ms. The simulation results show that if the attacker omits 10 number of FFs and LUTs from the Gate-chain, the dynamic power will reduce from 0.627 w to 0.625 w. This shows that variation in power consumption is just useful for large size HTHs and its accuracy depends on the precision of the measurement tools. The output signature is another parameter for removal attack detection. Simulation results show that by omitting just one of the AND-gate from the chain, with a same input data, output signature changes from 0XE0768 to 0XF0EEC. If Gate-chain is created just with NOT gate and attacker omits even number of gates and the related FFs, the output signature will not change. In this case, just delay time or power consumption alteration can be used to detect HTHs, which imply that using a complicated Gate-chain is more secure than the simple NOT gate chain.

2) Redesign attack in the chain: If an attacker changes the functionality of the LUTs of the chain, the generated output signature reveals the attack. To analyse the effect of redesign attack on Gate chain, we implemented two designs. In the first design, we use NOT-chain as a filler cell for more simplicity. In these chains, as the number of NOT gates are even, inputs and their corresponding output responses have the same value. For example, for input 0xAAA, the output response is the same in secure mode; however, after inserting HTHs, in which one of the NOT gates is exchanged with an AND gate, the output response changed to 0xFFF. For more security, instead

of NOT gates, it is better to insert another type of gates to design a special long sequential circuit. If any HTHs is inserted in this Gate-chain, it can be detected by its effect on the signature and delay time of the chain. In the second design, we use a different type of gate in Gate-chain including AND, OR and NOT gates. Simulation result shows that if one of the AND gates in Gate-chain converts to OR-gate, for the same stream of inputs, the output alters from 0X0768 to 0X0EEC. According to the simulation results, we can conclude that any omitting followed by the inserting of Gate in Gate-chain, even if it produces the same functionality, can be revealed by changing in time or power consumption. 3) Redesign attack on the main design: An attacker can change the functionality of the low observable/controllable points by changing the content of the related LUTs in the main design. The purpose of the attacker is denial of service or change the functionality of the system. As all low-testable points are routed to the unused output ports or to the input of the Gates of the designed chain, this kind of attacks can be detected by monitoring the design in test mode. Simulation results show that manipulating the content of LUT6, which named (data [127]_i_54) in Fig. (6), changes the functionality of the selected LUT output, which is observable in the output port "x". We set the content of LUT6 (data[127]_i_54) to zero for all different inputs. Monitoring port "x" which is zero for all different input samples of the main design reveals the attack. As mentioned before, if there is no free output port, unsecured point in the main design is routed to a gate in the chain. By injecting logical proper input to the chain and imposing proper test vector to the inputs of the main design, the related response of all zero will be declared in the output of the chain. Table (3) compares the input and its response with and without HTHs. In this section, we reviewed the capability of the proposed method against hardware Trojan attacks. This attack analysis showed the suggested method in this paper and in [10] can improve the security of the FPGA against HTHs that change the functionality of the existing LUTs and HTHs that add extra gates to the main design. We also examined the efficiency of the proposed method in terms of dealing with small and large sized HTHs. Simulation results show our proposed method can detect all of them with analyzing different output parameters [5].

Table 3. Output Response and output delay of Circuit in Secure Mode compared to The Presence of The HTHs

system response without HTH	system response with HTH	type of attack
2.13ms	1.1ms	Omitting 1000 FFs from Shift register
0.627w	0.625w	Omitting 10 FFs and gate from chain
0xE0768	0xF0EEC	Omitting one gate from chain
0XAAA	0XFFF	Changing the type of a gate from NOT to AND
0X0768	0X0EEC	Changing the type of a gate from AND to OR
Output x varied with different input	Output x fixed to zero with different input	Changing functionality of one LUT in main design

5 Conclusion

In this paper, we proposed and discussed an efficient two-phase method to prevent and detect hardware Trojan insertion. This method detects the HTHs insertion regardless of their size, which are inserted by functional changing of existing LUTs or adding extra circuit. In the first phase, empty spaces on FPGA are filled with a chain of logic gates and FFs so that there is no free space for Trojan insertion. If an attack occurred, it would be detected using delay and power pattern and logical output analysis of Gate-chain and shift register. In the second phase, the proposed scheme improves the testability of the low observable/controllable points of design by routing them to the unused ports or one of the gates in the Gate-chain. Our user-friendly approach to secure the FPGA has the capability of both detecting and preventing from HTHs insertion in main design and unused space, which cover different categories of HTHs. Dynamic power overhead of the protected design in test mode depends on the size of the original circuit and unused space that must be filled. The proposed method incurs no power overhead for the main circuits with less than %1 delay overhead which is negligible , and therefore improves the security without any performance deterioration.

References

[1] Ajay Rupani and Gajendra Sujediya. A review of fpga implementation of internet of things. *International Journal of Innovative Research in Computer and Communication Engineering*, 4(9), 2016.

[2] Greg Stitt, Robert Karam, Kai Yang, and Swarup Bhunia. A uniquified virtualization approach to hardware security. *IEEE Embedded Systems Letters*, 9(3):53–56, 2017.

[3] Rajat Subhra Chakraborty, Indrasish Saha, Ayan Palchaudhuri, and Gowtham Kumar Naik. Hardware trojan insertion by direct modification of fpga configuration bitstream. *IEEE Design &*

Test, 30(2):45–54, 2013.

[4] Stephen M Trimberger and Jason J Moore. Fpga security: Motivations, features, and applications. *Proceedings of the IEEE*, 102(8):1248–1265, 2014.

[5] Jie Li and John Lach. At-speed delay characterization for ic authentication and trojan horse detection. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 8–14. IEEE, 2008.

[6] Sharareh Zamanzadeh and Ali Jahanian. Asic design protection against reverse engineering during the fabrication process using automatic netlist obfuscation design flow. *ISeCure*, 8(2), 2016.

[7] Maxime Lecomte, Jacques Fournier, and Philippe Maurine. An on-chip technique to detect hardware trojans and assist counterfeit identification. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3317–3330, 2016.

[8] Mainak Banga and Michael S Hsiao. A novel sustained vector technique for the detection of hardware trojans. In *2009 22nd international conference on VLSI design*, pages 327–332. IEEE, 2009.

[9] Jiaji He, Yiqiang Zhao, Xiaolong Guo, and Yier Jin. Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10):2939–2948, 2017.

[10] Mansoureh Labbafniya and Roghaye Saeidi. Secure fpga design by filling unused spaces. *ISeCure-The ISC International Journal of Information Security*, 11(1):47–56, 2019.

[11] Mansoureh Labafniya, Stjepan Picek, Shahram Etemadi Borujeni, and Nele Mentens. On the feasibility of using evolvable hardware for hardware trojan detection and prevention. *Applied Soft Computing*, page 106247, 2020.

[12] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 1, pages 246–251. IEEE, 2004.

[13] Sharareh Zamanzadeh and Ali Jahanian. Security path: An emerging design methodology to protect the fpga ips against passive/active design tampering. *Journal of Electronic Testing*, 32(3):329–343, 2016.

[14] Franck Courbon, Philippe Loubet-Moundi, Jacques JA Fournier, and Assia Tria. A high efficiency hardware trojan detection technique based on fast sem imaging. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 788–793. IEEE, 2015.

[15] Sanchita Mal-Sarkar, Robert Karam, Seetharam Narasimhan, Anandaroop Ghosh, Aswin Krishna, and Swarup Bhunia. Design and validation for fpga trust under hardware trojan attacks. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):186–198, 2016.

[16] Ahmad Shabani and Bijan Alizadeh. Pmtp: A max-sat based approach to detect hardware trojan using propagation of maximum transition probability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[17] Marie-Lise Flottes, Sophie Dupuis, Papa-Sidy Ba, and Bruno Rouzeyre. On the limitations of logic testing for detecting hardware trojans horses. In *2015 10th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–5. IEEE, 2015.

[18] Uthman Alsaiari and Fayez Gebali. Hardware trojan detection using reconfigurable assertion checkers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(7):1575–1586, 2019.

[19] Paris Kitsos and Artemios G Voyiatzis. Fpga trojan detection using length-optimized ring oscillators. In *2014 17th Euromicro Conference on Digital System Design*, pages 675–678. IEEE, 2014.

[20] Kan Xiao and Mohammed Tehranipoor. Bisa: Built-in self-authentication for preventing hardware trojan insertion. In *2013 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 45–50. IEEE, 2013.

[21] Papa-Sidy Ba, Manikandan Palanichamy, Sophie Dupuis, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyre. Hardware trojan prevention using layout-level design approach. In *2015 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. IEEE, 2015.

[22] Behnam Khaleghi, Ali Ahari, Hossein Asadi, and Siavash Bayat-Sarmadi. Fpga-based protection scheme against hardware trojan horse insertion using dummy logic. *IEEE Embedded Systems Letters*, 7(2):46–50, 2015.

[23] Hoyoung Yu, Hansol Lee, Sangil Lee, Youngmin Kim, and Hyung-Min Lee. Recent advances in fpga reverse engineering. *Electronics*, 7(10):246, 2018.

[24] Jean-Baptiste Note and Éric Rannaud. From the bitstream to the netlist. In *FPGA*, volume 8, pages 264–264, 2008.

[25] Mohammad Saleh Samimi, Ehsan Aerabi, Zahra Kazemi, Mahdi Fazeli, and Ahmad Patooghy. Hardware enlightening: No where to hide your

hardware trojans! In *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 251–256. IEEE, 2016.

[26] Sophie Dupuis, Papa-Sidi Ba, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, pages 49–54. IEEE, 2014.

[27] Andrea Marcelli, Marco Restifo, Ernesto Sanchez, and Giovanni Squillero. An evolutionary approach to hardware encryption and trojan-horse mitigation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1593–1598. IEEE, 2017.

[28] Srinath Perera. *Hadoop MapReduce Cookbook*. Packt Publishing Ltd, 2013.

[29] Christoph Albrecht. Iwls 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*, 2005.

[30] Li Shang, Alireza S Kaviani, and Kusuma Bathala. Dynamic power consumption in virtexâĎć-ii fpga family. In *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pages 157–164, 2002.

Mansoureh Labafniya received her B.S. in hardware computer engineering from Islamic Azad University, South Tehran branch, Tehran, Iran in 2008. Her first M.S. degree is in computer architecture in 2010 and her second M.S. degree is in mechatronic Engineering from Sharif University of Technology, Tehran, Iran in 2012. She was a visiting researcher at KU Leuven for six months in 2018 and 2019. Now she is Ph.D. student at Isfahan University. Her research interests include hardware security, digital system design and residue number system.

Shahram Etemadi Borujeni is born in Borujen, Iran in 1964. He got his B.Sc. in Electrical Engineering from Iranian University of Science and Technology in 1987, and his M.Tech. degree in Radar and Communication Engineering from Indian Institute of Technology, Delhi in 1992. He got his Ph.D. degree in Computer Architecture Engineering at Shahid Beheshti University, Iran in 2010. He is now with Computer Engineering Faculty at University of Isfahan, Iran. His research interest includes Image Encryption, Design for Test and Hardware Security.

Roghayeh Saeidi received the M.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2007, and the Ph.D. degree from Sharif University of Technology, Tehran, Iran, in 2014. From 2009 to 2015, she was a member of the Advanced Integrated Circuit Design Laboratory, Sharif University of Technology. Since 2015, she has been an Assistant Professor of Iran Telecommunication Research Center (ITRC). Since 2019, he has been a postdoctoral research assistant with UCLouvain. Her current research interests include ultra-low-power integrated circuits design for biomedical applications, DC-DC converters, low-power SRAM circuits, analog, mixed-signal integrated circuits, and hardware security. She enjoys working on engineering products to improve the quality of life without discrimination.