The ISC Int'l Journal of Information Security

January 2019, Volume 11, Number 1 (pp. 47–55)

http://www.isecure-journal.org

Secure FPGA Design by Filling Unused Spaces

Mansoureh Labafniya [1,*], and Roghayeh Saeidi [2]

[1] *Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran*

[2] *Information and Communication Technology Research Institute, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 8 August 2018
Revised: 22 December 2018
Accepted: 26 January 2019
Published Online: 30 January 2019

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

ABSTRACT

There are different kinds of attacks on Field Programmable Gate Array (FPGA). As FPGAs are used in many different applications, its security becomes an important concern, especially in Internet of Things (IoT) applications. Hardware Trojan Horse (HTH) insertion is one of the major security threats that can be implemented in unused space of the FPGA. This unused space is unavoidable to meet the place and route requirements. In this paper, we introduce an efficient method to fill this space and thus leaving no free space for inserting HTHs. Using a shift register in combination with gate-chain is the best way of filling unused space, which incurs a no increase in power consumption of the main design. Experimental results of implementing a set of IWLS benchmarks on Xilinx Virtex devices show that the proposed prevention and detection scheme imposes a no power overhead and no degradation to performance and critical path delay of the main design.

1 Introduction

Field Programmable Gate Array (FPGA) is one of the main modules in the embedded systems including defense, automotive, health care, and the Internet of Things (IoT). FPGA is also increasingly used to improve the performance and energy efficiency of many computational applications. As a result, the security of FPGA has become an important parameter, which must be taken into account during the design process.

To secure the FPGA lifecycle, the vulnerability of design flows including base array and application design should be addressed. The base array design has the same vulnerability and security concerns as any other integrated circuits and semiconductor devices design. The FPGA is designed using commercial libraries and design tools. This design is manufactured at a foundry and then tested and sent back to the designer's company. Because of the high cost of manufacturing at foundries, the designer does not usually manufacture and package the FPGAs by themselves, and may cause security concerns. Malicious modifications of ICs, referred to hardware Trojans, are major security concerns in base array design flow, which usually aims to leak secret information or malfunction FPGA during field operation. During the manufacturing process of FPGA, final design is hidden because it will be specified with the end user in HDL code, place & route process, and final bitstream [1, 2].

The application design also has a design process including FPGA vendor tools, application developer or libraries from FPGA vendors. In comparison to Application Specific Integrated Circuits (ASIC) Solutions, the bitstream provides flexibility, which enables more security features and settings. This flexibility is because of partial reconfiguration and implementation

* Corresponding author.
Email addresses: mlabaf@eng.ac.ir (M. Labafniya), rosaeidi@yahoo.com (R. Saeidi)

ISSN: 2008-2045

changing with manipulating bitstream. More security features can be applied by the end user to hide the design details from the untrusted foundries and design house to improve the protection against malicious activity. Moreover, this flexibility allows saving cost and decreases the time to market of products [1].

After converting to bitstream, the application design is loaded on FPGA. It is programmed directly in antifuses or flash memory cells, but in SRAM based FPGAs, the bitstream is loaded on the external nonvolatile memory. Depending on the type of FPGA, an adversary may attack this bitstream. In comparison to nonvolatile FPGAs such as flash or antifuse FPGA, volatile FPGAs are more vulnerable to bitstream attack. Volatile FPGAs, particularly SRAM ones, has been criticized over this attack [3]. By reading the bitstream, the adversary can use reverse engineering to recover the circuit design. As a result, duplicating the functionality or inserting Hardware Trojans Horses (HTHs) will be possible. This is a unique aspect of FPGAs in comparison to ASICs, which makes them more vulnerable to malicious modification even after manufacturing the chip. Therefore, security protection mechanism is needed during run time or after shipment [4].

HTHs prevention and detection are two major issues in FPGA design because they can disturb or malfunction the system or leak secret information. Although many different prevention or detection methods have been proposed for FPGA design [5–10], there is no efficient scheme for both detection and prevention purposes. HTHs can be inserted either on the FPGA chip in the untrusted foundries by FPGA CAD tools, or in the bitstream of FPGA design. In this paper, we will focus on the potential HTHs in the bitstream.

This paper describes a new protection and detection method for HTHs insertion on FPGAs. In the proposed protection scheme, after designing the main circuit and its bitstream, we identify unused FFs and LUT and fill them in a way that not only it prevents the HTHs insertion but also improves their detection. We fill the unused space with HDL code. The proposed scheme has no static energy overhead with no delay overhead for main design. By using TCL instruction to create checkpoint, critical path of the main design will not change. Also, the dynamic power overhead of our method is just in test mode as the added gates to main design are not activated in normal mode.

The rest of the paper is organized as follows. Section 2 reviews previous work. In Section 3, our proposed method for prevention and detection of HTHs insertion is presented. Section 4 provides the simulation results and beneficial of our scheme; and finally, Section 5 concludes the paper.

2 Related Work

An adversary modifies the original design by inserting extra gates or changing the existing ones to accomplish the malicious intentions through the hardware Trojan. There are different approaches to prevent HTHs insertion in design such as delay based methods [5], rare event removal [9] or design for Trojan test [11].In delay based methods, the designer selects an internal path delay as a fingerprint. In the rare event removal methods, the designer tries to tune the transition frequency of internal nodes. By tuning this transition, there are not any potential places for HTHs insertion, and if any Trojan is inserted, it would be detected soon. In the design for the Trojan test methods, the attacks are prevented by hardening the design [12].

HTHs Detection methods are used to clarify the existence of HTHs. These detection methods are classified into three categories. The first category is, visual detection techniques, which use imaging by X-ray or microscopy scanning to identify HTH. These methods are time-consuming and expensive in addition lack of resolution [13, 14]. The second category is side channel analysis which involves measurement of information obtained from an IC's side channels. Power consumption, electromagnetic field, delay time and variation in current or voltage are obtained information from a design that is desirable for adversary [7, 10]. The third category of HTHs detection is logic testing. This method uses input vectors to test the functionality of logic blocks by automatic test pattern generation (ATPG) tools. As the circuits become complicated and have large number of internal nodes, exhaustive testing of all nodes are impossible and time-consuming [14, 15].

One of the special features of FPGAs is their bitstream. In addition to all prevention and detection methods for hardware security in both ASICs and FPGAs, extra methods must be used for security of bitstream in FPGAs. Although there are many different schemes for protecting bitstream against reverse engineering, like obfuscation methods in [1] or using ATMR structure [2, 16], the main challenge of FPGAs is their unused spaces. There are different places in design for HTHs insertion so that Unused space both in ASIC and FPGA is one of the potential places for HTHs insertion. For achieving an optimum place and route results in FPGA, as usual, just 60% of Look Up Table (LUT) and Flip Flops (FFs) should be used. The remaining unused spaces have a great potential for inserting Trojans; therefore, filling this unused area is essential for preventing HTHs insertion.

Paper [17] just fill a portion of FPGA with Ring Oscillators (RO) structure. RO is a common structure for detection of HTHs. RO is a circuit that oscillates

due to its inherent logic. The oscillation frequency depends on the exact components and the length of the RO. This frequency changes if any HTHs were enabled near RO. This sensitivity can be used just for HTHs detection without any prevention by not filling all the unused space [17]. Using RO structure has error deviation due to the existence of process variation which its behavior is like HTH. During this method, main design parameters may change.

In [18], the authors present a technique in which unused space in ASIC is filled using built-in self-authentication (BISA) technique and functional filler cells instead of non-functional ones. Using the BISA structure, all functional filler cells are tested to detect whether a Trojan is inserted and then produce a digital signature. Any modification on BISA will change this signature. However, a wisdom attacker can insert Trojan in unused space and change the BISA structure in a way that signature does not change. As a result of this attack, HTHs will not be detected. Using this method causes no degradation on main design parameters as BISA structure works independently to main design.

A more secure solution for ASICs was proposed in [19], in which instead of BISA, Shift Register is used. After filling unused space with Shift Registers, the other small unused space must be filled with gates and functional cells until all the unused spaces are filled. In this method, slack time is used to determine the priority of filling spaces. Like [19], this work has strong detection mechanism on HTHs insertion in unused space as there is a mechanism to check it in order that [20] is on ASICs and our paper is on FPGAs. As a result, the implementation way is different. The main design parameters do not change during this method as added structure works independently to the main design.

The author in [21] proposed a prevention scheme in FPGAs by filling unused space with fixed dummy information. This filling is added to the Xilinx Design Language (XDL) file to prevent HTHs insertion in unused cells. XDL is a human-readable representation of a netlist that contains placement and routing information. Since these added cells have fixed values without any clock pulse, it is possible for an attacker to detect them and change the functionality of these added cells to insert HTHs by reverse engineering the bitstream. There are reverse engineering tools which extract the bitstream from the external memory of the FPGA and utilize it to recover the XDL or netlist file [22, 23]. Considering routing of inputs, outputs and the fixed value in input pin of some cells from recovery file, we can discover these dummy cells, which are separated from other used module for main design. In this scheme, there is no feedback from design to monitor whether an attack occurs or not. Therefore, HTHs insertion in dummy space will not be detected. The author in [21] states that unused space works simultaneously with the main design without any static power overhead because FPGAs have fixed static power consumption. Filling unused space is a method to prevent HTHs insertion in these spaces. Previous works that their mechanisms are filling unused space are reviewed in the literature. Paper [18] and [19] discuss on ASIC circuit and the other ones, by [17] and [21] work on FPGA. Table 1 shows the comparison of these papers.

In the next Section, we will present a new scheme for both the prevention and detection of HTHs in unused spaces of FPGAs. The proposed structure uses a kind of modified RO or LFSR which helps to both HTHs prevention and detection. Our method forces no alteration on main design parameters like power consumption, place& route and path delays. This is due to the definition of test mode and the normal mode that is not used in [21]. Extra filled space works just in test mode. In this paper, we will focus on the potential HTHs in the bitstream, which manipulate it to insert extra malicious circuit in the design. The Trojan can be inserted by partial reconfiguration or by reverse engineering the bitstream. Table 1 shows the comparison of the proposed method by the other related papers.

3 The Proposed Scheme

As usual, most implementations on FPGAs are based on LUT and FFs although, in some version of FPGAs, there are other design components like DSP modules. In this paper, we suppose the attacker target is programmable parts of FPGA. The other parts like DSPs have another kind of threats and their HTHs are previously discussed in other articles [24]. If all LUTs and FFs are used in a design, there will be no free spaces for Trojan insertion. Figure 1 shows a Basic Logic Element (BLE) as the common structure of FPGA which consists of LUTs and FFs [25].

After the implementation of a design on a specific FPGA, the utilization rate of resources like LUTs and FFs are reported. In this paper, we want to propose a scheme to fill the unused space efficiently to prevent Trojan insertion and also facilitate HTHs detection.

The first step is to synthesize and implement the main unprotected design. It is possible to save current place and the route by using “write-checkpoint instruction in TCL section in Vivado. By using this checkpoint as a constraint, Vivado saves all critical paths and specification of the main design. By considering utilization rate after implementation, we will fill unused resources like LUTs and FFs by adding

Table 1. A Comparison between the implementation cost of 4×4 MDS matrices, for $m = 4, 8$ as the bit length

Methods	Hardware	Pros	Cons
[17]	FPGA	-Prevention by partial filling of unused space	-Sensitive to process variation -Degradation to main design parameters
[18]	ASIC	-Prevention by filling all unused space -No degradation to main design parameters	- No detection with weak feedback
[19]	ASIC	-Prevention by filling all unused space -Detection by strong feedback -No degradation to main design parameters	———————
[21]	FPGA	-Prevention by filling all unused space -No degradation to main design parameters	-No detection because of no feedback
This paper	FPGA	-Prevention by filling all unused space -Detection by strong feedback -No degradation to main design parameters -user friendly	———————

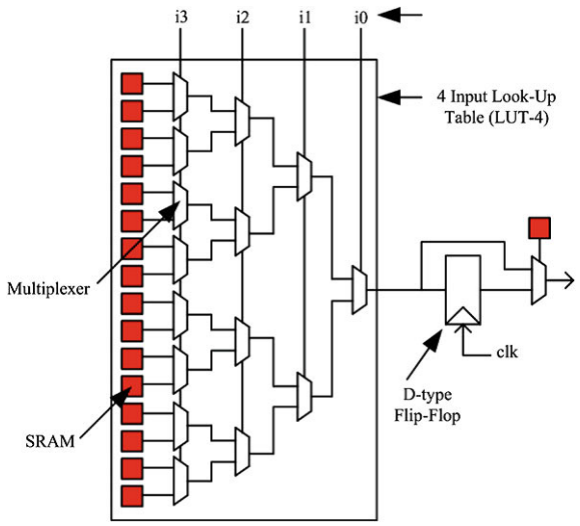

Figure 1. Basic Logic Element(BLE)[21].

a combination of the shift registers and gate-chains to the main design. This addition is by writing HDL code of proposed structures and adding it to the main design HDL code which is an easy and user friendly approach. The size of these two structures will be as large as the unused LUTs and FFs to fill all of them.

In comparison to LFSR, using shift register is more secure [15]. It is because the structure of LFSR is permanent and after inserting HTHs in unused space, the attacker can modify the LFSR structure to prevent HTHs detection. Structure of Shift register consists of FFs and structure of gate-chain consists of LUTs and FFs something like RO structure. We first fill all of the LUTs and the equivalent number of FFs using gate-chain. As the numbers of unused FFs usually are more than unused LUTs, if there are remaining FFs, they will be filled by the structure of Shift register. It is noteworthy to mention that as the functional part of the code is implemented on LUTs, filling them is sufficient for HTHs prevention, but to provide more obfuscation against reverse engineering, we also fill FFs. In this phase, the implementation setting is in such a way that it does not optimize the design and use saved checkpoint. HTHs insertion can be prevented with this scheme as there is no free space on FPGA. Any changes in added structure will be detected by at least one of the time, logic changes of chains output or power consumption variation.

Gate-chain can include different random type of gates with different feedback from chain to produce a special signature in output. Although for simplicity it can only contain NOT gate. Figure 2 shows our proposed structure in which unused space is filled by gate-chain and shift register.

In the next phase, after filling all the unused LUT and FFs by adding shift register and gate-chain to HDL code, EDIF file is generated from the final implementation. Figure 3 shows the flow diagram of the proposed scheme.

Because of using all extra LUTs and FFs in chains, the controllability and observability of them prevent the attacker to use them. In spite of this prevention method, simulation results in next section show that each alteration in added chain by the attacker will be detected.

In addition to prevention method, using the detection scheme will provide more security. In the proposed

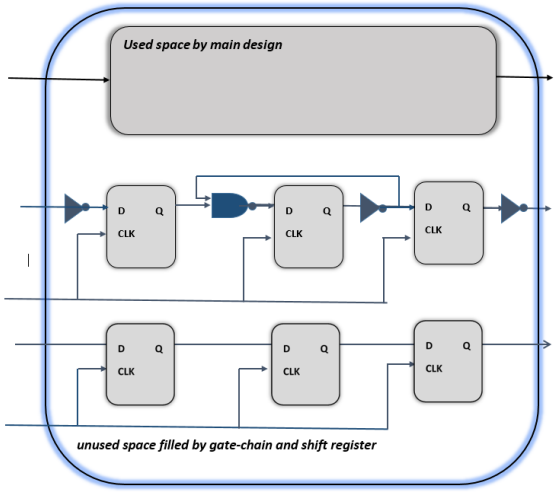

Figure 2. Proposed structure for HTHs prevention and detection in FPGAs

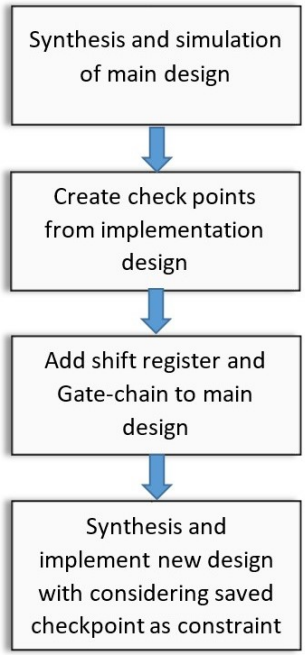

Figure 3. Flow diagram of the proposed scheme for HTHs prevention and detection in FPGAs.

structure after filling unused spaces with gate-chain and shift register, these places must be checked in test mode if any HTHs are inserted. Changing in output signature of the chain will reveal the attacks. HTHs can be inserted by partial reconfiguration of FPGA or by manipulating reversed engineered bitstream. As HTHs are added circuits in free space of chip, we consider the potential attack that inserts HTHs in unused space of chip which are now filled by the chain. The detection method in this phase is done by input and its responses from the added structure. If there is no HTH in design, output response will be the same as the desired output signature in value, delay, and power

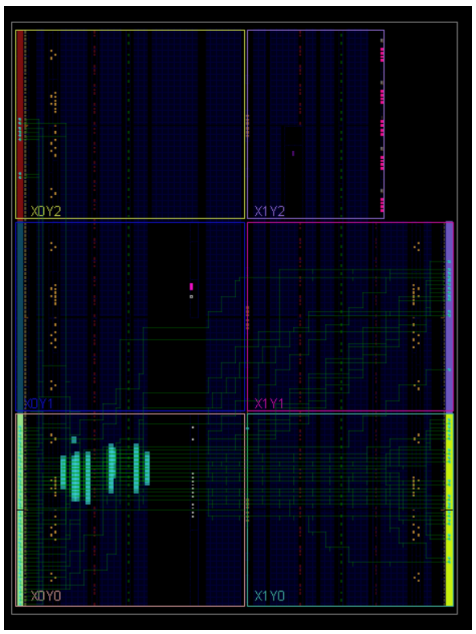

Figure 4. The rate of resource utilization in I2C design before filling unused space.

consumption. If HTHs are inserted in the main design by changing related LUTs and FFs functionality, it will be detected by sample inputs and their response in main design. Our proposed structure has two mode including the normal mode, in which just main design works, and the test mode, in which just added chains work to check if any HTHs are inserted or not. Defining two distinct functional modes for main design and added chains causes no power consumption overhead to main design.

In the next section, the simulation result of the proposed method is presented.

4 Simulation Results

To simulate the proposed scheme, we used the IWLS benchmark circuit [26]. The benchmark is simulated and implemented using Vivado design suite 2015.4. For each circuit, we tried to use the fittest device. Figure 4 shows I2C design before filling unused space, and Figure 5 shows the design after filling all the resources. It indicates that using the proposed method, we could fill all LUTs and FFs.

Since we use checkpoint to save a constraint from the main design, adding shift register and gate-chain will not modify the critical path and power consumption of the main design. It is because filler cells (gate-chain and shift register) and the main design never work simultaneously. It means that the Clock input of the main design is different from the Clock input of the added design in normal mode and test mode. Figure 6 shows a portion of the schematic, which is added to the main design by writing its HDL code. We

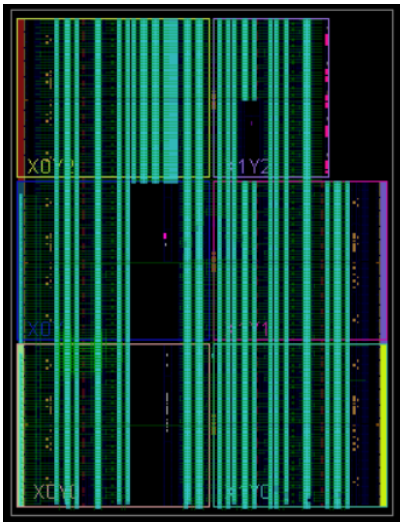

Figure 5. The rate of resource utilization in I2C design after filling unused space.

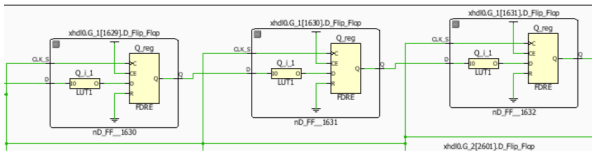

Figure 6. Schematic of gate-chain

considered two time-constraints, one for the clock of main design ($T = 10$ns) and the other one for the clock of gate-chain and shift register ($T = 2$ns). According to Table 2, implementing the proposed scheme on xa7a15tcpg236-2I and xa7a75tfgg484-2I FPGAs, do not affect the power consumption in normal mode in which just main design is working. It is because FPGAs have fixed static power consumption [21]. Dynamic power in test mode is increased more than 45 times compared to dynamic power in normal mode when we secure the FPGA although the static power is still fixed. Simulation results show that proposed scheme uses 100% of the resources.

4.1 Comparison

We can summarize the difference between our proposed method with paper [21] as follow:

- Paper [21] 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.
- Paper [21] uses a fixed value for dummy cells (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 [21] secures the design by manipulating XDL file but our method adds HDL code to main code which is more user friendly.

The similarity between paper [21] and our proposed method is:

- Both schemes have no effect on static power consumption as mentioned in [21]. Our simulation results in Table 2 confirms this fact. Our method has dynamic power consumption overhead during test mode, which is independent of the main design.
- Both papers can fill all LUTs and FFs successfully.
- Both methods protect the design against malicious object insertion in free resources of the device and are irrelevant to the modification made on the main design or other type of attacks.

4.2 Attack Analysis

In our paper we assume that attacker has physical accessibility to FPGA. So attacker can partial reconfigure or manipulate bitstream of FPGA. Although there are attackers that do not add extra circuit to attack the system [27], but in our paper, we only target the attackers who attack by adding extra gates in the main design. Since we have various types of attacker, there is no silver bullet and the detection/prevention method should be designed according to the attacker type. After filling the free spaces, it is impossible for the attacker to insert a HTH in dummy space. Even a successful reverse engineering may not help the attacker to launch an attack, because the free spaces are now filled with high testable chains. In spite of that if any HTHs are inserted, they will be detected by our proposed structure.

To launch a successful HTHs insertion, an attacker needs to insert a malicious circuit. This insertion can be in main design by changing the functionality of related LUTs and FFs or added extra circuit in unused space which is now filled by the chain. All these attacks are done by partial reconfiguration of FPGA or bitstream manipulation.

The proposed scheme protects the design against insertion of malicious objects in free resources and the HTH insertion in the main design is beyond the scope of this paper. Without considering the goal of the Trojan to change the functionality, denial of service or leakage of data from the main design, this type of attack will consume resources like LUTs and FFs. HTHs are inserted in low testable points of the circuit to make their detection more difficult. Changing in resource consumption and the routing between them will change: (1) the power consumption, (2) the logic of chain's output or (3) its response time. With each HTHs insertion in the added chain, at least one

Table 2. A Comparison between the implementation cost of 4×4 MDS matrices, for $m = 4, 8$ as the bit length

Benchmark	Unprotected design				Protected design			
	I/O utilization	Used LUT	Used FF	Power= Dyn + Static	Used LUT	Used FF	Power in normal mode= Dyn + Static	Power in test mode= Dyn + Static
I2C	87%	1%	1%	$0.075w = 0.005 + 0.07$	100%	100%	$0.075w = 0.005 + 0.07$	$0.56w = 0.497 + 0.07$
SPI	34%	5%	1%	$0.13w = 0.06 + 0.07$	100%	100%	$0.13w = 0.06 + 0.07$	$0.69 = 0.62 + 0.07$
Mem-cntl	94%	2%	1%	$0.12w = 0.03 + 0.09$	100%	100%	$0.12w = 0.03 + 0.09$	$0.84w = 0.75 + 0.09$
TV8	43%	12%	2%	$0.08w = 0.01 + 0.07$	100%	100%	$0.08w = 0.01 + 0.07$	$0.6w = 0.52 + 0.07$

of these parameters will be changed. It is hard and time-consuming for the attacker to find a circuit that has same delay, power consumption, and the logic response like the designed chain. Moreover, ensuring the malicious goal of the attacker to substitute the HTH with decreasing the possibility of detection is also very hard. If Trojan is small, variation in number of clock or variation in output logic helps detection process but for big HTHs in addition to these two parameters, power consumption variation helps to detect inserted Trojan. For designing more secure chain, it can consist of different kind of logic gates or combination circuit [19] although NOT gate is more simple.

HTHs insertion effects on chain are: 1) removal LUTs or FFs from chain to design HTHs circuit absolutely separated from chain, 2) change functionality of LUTs in order that changed LUTs still exist in chain. We tried to cover different kinds of HTHs that use omitting or changing the functionality of LUTs and FFs in chain. We choose these types of attacks because HTHs are usually inserted in free space or low testable points. This attack idea is the same as one which is used in [19]. For HTH insertion, we used the TRUST-HUB workbench by [28] that changes HDL code to insert HTHs. SPI code form IWLS is used for simulating different attack and observing the system response. We add one gate-chain with 4910 FFs and 4910 different type of gates that each of them is implemented with one LUT. Remaining FFs are used with one shift register.

- **Removal attack:** If an attacker omits some FFs or LUTs from shift register path or gate-chain, to design another circuit for malicious goal, it can be identified by the time delay between input and output response. Simulation results show that for SPI code in protected mode and without HTHs with 4910 FFs and LUTs in gate-chain, 4910 clock cycles are needed to get the chains output. If an attacker omits 10 FFs from the shift register and gate-chain for using them as HTHs separated from the chain, the time delay will decrease to 4900 clock cycles.

  According to the simulation result, omitting 10 FFs and LUTs from Gate-chain reduces dynamic power from 0.627 to 0.625, which is negligible. It shows that Analyzing power consumption variation can only be used for large size HTHs. It is noteworthy to mention that the accuracy of power analysis depends on the precision of the measurement tools.

  If an attacker omits just one of the AND-gate in chain, simulation results show that, output signature changes and HTH is detected. With same stream of inputs, before omitting, output stream is 0xE0768. After omitting one of the AND-gate in chain, the output changes to 0xF0EEC. If gate-chain is created just with NOT gate and attacker omits even number of them and their related FFs, this type of attack can be detected by delay time alteration or power consumption alteration although logic of output is not changed.
- **Redesign attack:** If an attacker changes the functionality of LUTs in the chain to build HTHs, which is not entirely isolated from the chain, it can be detected by the generated output signature. The designed attack change one of the AND gate in gate-chain to OR. Simulation results show that for same stream of inputs to gate-chain, output alters from 0X0768 to 0X0EEC.

Table 3 shows the simulation results for different kind of HTHs attacks.

5 Conclusion

In this paper, we presented a scheme to both prevent and detect HTHs attacks on FPGAs. In this method, we prevent HTHs attacks by filling unused space on FPGAs with shift registers and gate-chain to leave no free space for HTHs insertion. If an attack occurs

Table 3. A Comparison between the implementation cost of 4×4 MDS matrices, for $m = 4, 8$ as the bit length

Parameters to consider for HTHs detection	Output response without HTH	Output response with HTH	type of attack
power consumption	0.627w	0.625w	Omitting 10 LUTs and FFs from chain
output value	0xe0768	0x137c	Omitting one of the LUTs from chian
output delay	9.8 ms	7.8ms	Omitting 10 FFs from chain
output value	0xe0768	0x0eec	Exchanging one AND gate to OR gate in chain

and some extra design added in the free space which is now filled, to change or denial of services of main design, it would be detected using delay pattern and logical output analyzing of gate-chain and shift register. Experimental results showed that this protection/detection scheme imposes low power consumption with no degradation in delay and performance of the main design since added chain and main design work independently in normal mode and test mode.

References

[1] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Mutarch: Architectural diversity for fpga device and ip security," in Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, 2017, pp. 611-616: IEEE.

[2] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in fpga devices: threat analysis and effective counter measures," in Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI, 2014, pp. 287-292: ACM.

[3] S. M. Trimberger and J. J. Moore, "FPGA security: Motivations, features, and applications," Proceedings of the IEEE, vol. 102, no. 8, pp. 1248-1265, 2014.

[4] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware Trojan insertion by direct modification of FPGA configuration bitstream," IEEE Design & Test, vol. 30, no. 2, pp. 45-54, 2013.

[5] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan horse detection," in Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on, 2008, pp. 8-14: IEEE.

[6] S. Zamanzadeh and A. Jahanian, "ASIC design protection against reverse engineering during the fabrication process using automatic netlist obfuscation design flow," The ISC International Journal of Information Security, vol. 8, no. 2, pp. 93-104, 2016.

[7] M. Lecomte, J. Fournier, and P. Maurine, "An on-chip technique to detect hardware Trojans and assist counterfeit identification," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 12, pp. 3317-3330, 2017.

[8] M. Banga and M. S. Hsiao, "A novel sustained vector technique for the detection of hardware Trojans," in VLSI Design, 2009 22nd International Conference on, 2009, pp. 327-332: IEEE.

[9] M. S. Samimi, E. Aerabi, Z. Kazemi, M. Fazeli, and A. Patooghy, "Hardware enlightening: No where to hide your hardware trojans!," in On-Line Testing and Robust System Design (IOLTS), 2016 IEEE 22nd International Symposium on, 2016, pp. 251-256: IEEE.

[10] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2939-2948, 2017.

[11] S. Zamanzadeh and A. Jahanian, "Security Path: An Emerging Design Methodology to Protect the FPGA IPs Against Passive/Active Design Tampering," Journal of Electronic Testing, vol. 32, no. 3, pp. 329-343, 2016.

[12] M. Tehranipoor and C. Wang, Introduction to hardware security and trust. Springer Science & Business Media, 2011.

[13] F. Courbon, P. Loubet-Moundi, J. J. Fournier, and A. Tria, "A high efficiency hardware trojan detection technique based on fast SEM imaging," in Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, 2015, pp. 788-793: EDA Consortium.

[14] S. Mal-Sarkar, R. Karam, S. Narasimhan, A. Ghosh, A. Krishna, and S. Bhunia, "Design and validation for FPGa trust under hardware Trojan attacks," IEEE Transactions on Multi-Scale Computing Systems, vol. 2, no. 3, pp. 186-198, 2016.

[15] M.-L. Flottes, S. Dupuis, P.-S. Ba, and B. Rouzeyre, "On the limitations of logic testing for detecting hardware Trojans horses," in Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2015 10th International Conference on, 2015, pp. 1-5: IEEE.

[16] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures," Proceedings of the IEEE, vol. 102, no. 8, pp. 1229-1247, 2014.

[17] P. Kitsos and A. G. Voyiatzis, "FPGA Trojan detection using length-optimized ring oscillators," in Digital System Design (DSD), 2014 17th Euromicro Conference on, 2014, pp. 675-678: IEEE.
[18] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware Trojan insertion," in Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on, 2013, pp. 45-50: IEEE.
[19] P.-S. Ba, M. Palanichamy, S. Dupuis, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Hardware trojan prevention using layout-level design approach," in Circuit Theory and Design (ECCTD), 2015 European Conference on, 2015, pp. 1-4: IEEE.
[20] A. Amelian and S. E. Borujeni, "A Side-Channel Analysis for Hardware Trojan Detection Based on Path Delay Measurement," Journal of Circuits, Systems and Computers, vol. 27, no. 09, p. 1850138, 2018.
[21] B. Khaleghi, A. Ahari, H. Asadi, and S. Bayat-Sarmadi, "Fpga-based protection scheme against hardware trojan horse insertion using dummy logic," IEEE Embedded Systems Letters, vol. 7, no. 2, pp. 46-50, 2015.
[22] H. Yu, H. Lee, S. Lee, Y. Kim, and H.-M. Lee, "Recent Advances in FPGA Reverse Engineering," Electronics, vol. 7, no. 10, p. 246, 2018.
[23] J.-B. Note and . Rannaud, "From the bitstream to the netlist," in FPGA, 2008, vol. 8, pp. 264-264.
[24] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," IEEE design & test of computers, vol. 27, no. 1, 2010.
[25] U. Farooq, Z. Marrakchi, and H. Mehrez, Tree-based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization. Springer Science & Business Media, 2012.
[26] C. Albrecht, "IWLS 2005 benchmarks," in International Workshop for Logic Synthesis (IWLS): http://www. iwls. org, 2005.
[27] M. Ender, S. Ghandali, A. Moradi, and C. Paar, "The First Thorough Side-Channel Hardware Trojan," in International Conference on the Theory and Application of Cryptology and Information Security, 2017, pp. 755-780: Springer.
[28] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in Computer Design (ICCD), 2013 IEEE 31st International Conference on, 2013, pp. 471-474: IEEE.

Mansoureh Labafniya was born in Tehran, Iran and got 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 from Islamic Azad University, Science and Research branch, Tehran, Iran in 2010 and her second M.S. degree is in mechatronic from Sharif University, Tehran, Iran in 2012. Now she is Ph.D. student at Isfahan University. Her research interests include digital system design, hardware security and residue number system.

Roghayeh Saeidi received the B.S. and M.S. degrees in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2004 and 2007, respectively, and the Ph.D. degree from Sharif University of Technology, Tehran, Iran, in 2014. From 2009 to 2015, she was a member with the Advance Integrated Circuit Design Laboratory, Sharif University of Technology. She is currently an assistant professor at the Iran Telecommunication Research Center, Tehran, Iran. Her current research interests include low-power SRAM circuits, digital system design, hardware security, statistical analysis, analog and mixed-signal integrated circuits, and RFIC design.