The ISC Int'l Journal of Information Security
ISeCure

January 2016, Volume 8, Number 1 (pp. 53–60)
http://www.isecure-journal.org

Self Authentication Path Insertion in FPGA-based Design Flow for Tamper-resistant Purpose

Sharareh Zamanzadeh [1,*], and Ali Jahanian [1]
[1] *Computer Science and Engineering Department, Shahid Beheshti University, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 15 August 2015
First Revised: 27 November 2015
Last Revised: 21 December 2015
Accepted: 31 December 2015
Published Online: 4 January 2016

Keywords:
FPGA, Hardware Security, IP Protection, Security Path.

ABSTRACT

FPGA platforms have been widely used in many modern digital applications due to their low prototyping cost, short time-to-market, and flexibility. Field-programmability of FPGA bitstream has made it a flexible and easy-to-use platform. However, access to the bitstream degraded the security of FPGA IPs because there is no efficient method to authenticate the originality of a bitstream by the FPGA programmer. The issue of secure transmission of configuration information to the FPGAs is of paramount importance to both users and IP providers. In this paper, we presented a "Self Authentication" methodology in which the originality of sub-components in the bitstream is authenticated in parallel with the intrinsic operation of the design. In the case of discovering violation, the normal data flow is obfuscated and the circuit would be locked. Experimental results show that this methodology considerably improves the IP security against malicious updates with reasonable overheads.

1 Introduction

Field-Programmable Gate Arrays (FPGAs) offer flexibility in high-performance computing. FPGAs are grown in capacity and complexity while they are provided with lower prototyping cost and less time-to-market. These merits make them suitable for all application ranges from consumer products to military systems. As FPGAs have become larger and more capable, the value of the IP of application designs are grown. However, there is not a good balance between security measures and the value of the information being protected. This fact motivates IP providers to invest in built-in security functions [1].

FPGAs are programmed using a binary file called "configuration bitstream". The generated bitstream is a software form of the hardware design loaded in the FPGA. In fact, it is qualitatively much like microprocessor software. Despite advantages like design simplicity, configurability and short time-to-market, bitstream is susceptible to all the same security concerns that surround software, including design tampering or reverse engineering, IP piracy or Trojan insertion [2].

Configuration bitstream contains the logic and connectivity of the design and it should be transferred from the PC where the CAD tool is running to the FPGA. The syntax and semantic of these files are commercially confidential and there is little information about their details in the public domain. For several years, this kind of obfuscation, known as obscurity, was the main mechanism to keep the programmed design secure. However, recently some papers such as [3] proposed an easy process to decompile the bitstream of modern FPGA families to a textual netlist description in a short time. Reverse engineering of the bitstream which is performed in [3] leads to netlist

[*] Corresponding author.
Email addresses: sh_zamanzadeh@sbu.ac.ir (S. Zamanzadeh), jahanian@sbu.ac.ir (A. Jahanian)
ISSN: 2008-2045 © 2016 ISC. All rights reserved.

extraction and makes the whole design information available to attacker and also it is the introduction to IP theft or malicious design tampering.

There are also some reports of directly inserted Trojans in the configuration bitstream [2], [4]. In [2], a ring oscillator circuit is inserted in to the bitstream in order to decreas the expected lifetime by increasing the chip operation temperature. On the other hand, Trojan insertion in [4] is more intelligent and it is performed intentionally to open a back door. This leaked information would be used to break the encryption algorithms like AES and 3DES that are embedded in the configuration bitstream.

Since bitstream should be transferred as IP through global network or in the commercial market, the embedded netlist is threatened with tampering or IP theft. Although, encryption techniques are used as secure bitstream transferring, there are some gaps in security domain which is provided by encryption techniques and bitstream encryption also has its own security challenges as follows:

- There are serious security and technological problems in key storage and key transmission if the bitstream is encrypted because the attacker has full access to bitstream and body of the FPGA. In more details, bitstream decryption key should be programmed by the IP provider in a non-volatile memory inside the FPGA but modern FPGAs are SRAM-based and have no E2PROM.
- Encrypting and decrypting the bitstream has overheads at both IP provider side and user side that cannot be easily tolerated, especially for runtime reconfiguration applications. FPGAs with built in encryption are expensive and power-hungry for many embedded applications [3, 5].
- It has been reported that the built in encryption has been flawed by side-channel analysis attacks [6–8]. It is worth noting that there are many families with numerous deployed FPGAs which support bitstream encryption [2], but they cannot be used in many of the applications. An important point is that encryption is just a well-known method to improve bitstream confidentiality [6, 7], but it does not provide authentication and integrity for the design [9] during the operation time. Any bitstream manipulation is not necessarily a meaningful Trojan and tampered bitstream may decrypt to nonsense configuration [9].
- Bitstream encryption methods provides a secure cube for shielded message transmission.In other word, they are used when a protected bitstream is going to be transferred and programmed in the FPGA, however protecting IPs from the evil designers who misuse the bought IPs or immoral attackers in the foundry and IP market is a different security aspect.

In this paper, we have set up a built-in authentication path in parallel with the data and control path of the design to efficiently authenticate the originality of the IP, in the form of bitstream for both IP providers and end users. In this self-authentication methodology, the functionality of sub-components is monitored in parallel with the intrinsic operation of the design. In the case of discovering violation, the normal data flow is blocked and the circuit would be locked. This path is composed of two main elements (authentication and blocking modules). The main contributions of this research are as follows:

- Proposing the semantic of self-authentication path for FPGA-based designs, to prevent from runtime functional-tampering.
- Automating the self-authentication path insertion process to make FPGA CAD tools include authentication plug-in option.

The rest of the paper is organized as follows. Threat model is discussed in Section 2. Section 3 3 describes the proposed methodology, including authentication path architecture and the security flow that these elements have made. Experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2 Threat Model

In modern EDA business, FPGAs are widely used in low circulation products and security of the FPGA at runtime is very important for designer, service provider and end customer. Moreover, faults of the EDA tool and FPGA platform can be found during the test process but runtime attacks can not be controlled by the designer. Besides there is no rigid boundary between the vulnerabilities of FPGA EDA tools and ASIC. Also, the security issues associated with the design and manufacture of the base array are similar to those of other semiconductor devices. This paper is focused on the security concerns that comes from the need to protect the application design in different stages of bitstream lifecycle especially in post-configuration (runtime stage) [1] and [10].

Quick innovation, well design and professional subcircuits, spreading out the cost of design are the advantages that cause IPs cores to become common practice in semiconductor Industry in both aspects of FPGA logic design and ASIC chip designs. IPs are presented mostly in the form of bitstream and are transferred among designers in plain text. The main targeted issue in this paper is related to when the FPGA based application is compiled into a bitstream and should be transmitted into the FPGA base array or be delivered to the market as an IP-core. The bitstream may be partially updated, or environmental faults might be injected to the FPGA to tamper the programmed bitstream. Also, it may be at the risk of malicious manipulation of designers who can alter the core by the information obtained from reveres engineering.

In this paper, integrity and authentication of the design on FPGA are considered rather than its confidentiality. It is worth noting that the provided mechanism is not the superseded for the classic encryption /message authentication mechanisms, but is used along with existing security techniques to compensate for security weak points and provides balance the cost benefit concerns of practical applications. Moreover, Trojans that would change the functionality of the design are studied in this defense mechanism. But the alterations with the aim of physical damage (e.g. pin manipulation) is not targeted. The opponents are neither the IP designer nor IP provider, but competitors or ill-wishers who are skillfull in circuit design or the IP consumers. Also, there would be no computational, financial and temporal limitation for them. The attackers may acquire unauthorized knowledge (key, design ides), open back-door during bitstream malicious manipulations. Also runtime malfunctions such as environmental faults or tampers through runtime reconfiguration.

In the proposed methodology, security path would be inserted by the CAD tool automatically after place and route phases and before bitstream generation. No reported information would be saved from the security path generating process by the CAD tool. Also, the converting process for updating bitstream with security path would be carried out internally. No key is applied as input to the self-authentication protection mechanism, hence no key management mechanism is necessary.

3 Proposed Methodology

As mentioned before, we proposed a new methodology in which a new path (security path) is added to regular ASIC design flow to improve the security of IP/IC. Developing the security path in the design makes it self-authenticated against the bitstream manipulations which will effect the functionality of the circuit. The proposed security path is an extra path, parallel with data/control paths to authenticate the configured circuit at runtime. The functionality of the whole/some parts of the circuit will be obfuscated if any violation is detected through the authentication modules. This mechanism is implemented in VPR CAD tool as a security plug-in phase. This phase is added to VPR design flow after placement and routing steps and before the final bitstream is being generated. Figure 1 illustrates the authentication path concept in abstract form.

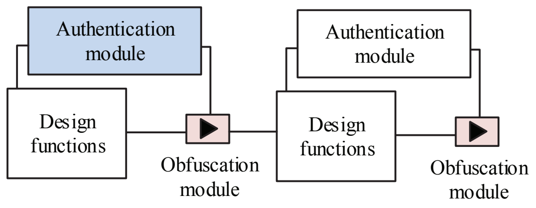

Figure 1. Conceptual view of Security path.

As shown in Figure 1, authentication path consists of two main parts (dark modules); Authentication and obfuscation modules. Authentication modules verify functionality of the design's components. In this methodology, functionality of the circuit will be corrupted by the security path when authentication module(s) discover any deviation. As it is illustrated in the Figure 1, the authentication path is like a chain with several sub-paths each of which verifies a small part of the circuit. Each sub-path gets input from the data path and applies on data path as well. Obfuscation module is a junction for both data and authentication path. In the case of violation detection, the obfuscation modules will propagate faulty data (opposite logic) through data flow. It is worth noting that whenever authentication signal is activated it would remain active until the FPGA is reconfigured again.

It is worth noting that this approach does not have any inconsistency with any other common security solutions described at introduction section. This approach can be applied along with them as a supplementary solution and/or individually as a superseded solution. Moreover, in the proposed technique, the regularity of the FPGA architecture is kept. In the next subsections the main tasks of authentication path are explained in details.

3.1 Proposed Authentication Path Architecture

The main functionality of authentication module is checking the correctness of LUT functionality at run time. In the proposed architecture, each LUT is equipped with an individual authentication module. LUT and its related authentication module built a compound component by the name of security cell. Authentication modules are programmable elements which are programmed at configuration time; however, their bitstream is independent from LUTs. It also means that authentication module is aware of LUT's function, its configuration bits are distinct from the related LUT. In this proposed mechanism we are focused on authentication detection rather than correction, since in order to conceal the caused alteration, the attacker should modify bitstream in such a way that bitstreams of authentication and related LUT have consistent functionality. However, this alteration is considerably hard. Therefore, it does not matter which one (authentication module or the LUT) is modified, alteration in any of them leads to authentication detection.

Authentication module is a normal LUT that is equipped with hardwired output that authenticates the originality of the related LUT. It can be the same size as corresponding LUT or a subset of it. In this case, only a subset of LUT's minterms can be verified. It is worth noting that authentication LUTs can be smaller than regular LUTs if they are prefabricated with lower number inputs in FPGA array. It is possible to have authentication modules of various sizes whose distribution are adjusted based on statistical analysis of security and implementation overhead tradeoff. In this paper, we have implemented the architecture with ($n = m$) since in this situation the merits and demerits of proposed authentication mechanism (the security path) would be assessed more clearly. However, statistical analysis to include smaller-sized authentication modules is going to be discussed in the further research. Including smaller-sized authentication module would bring a desired degradation in authentication level but it would balance the costs with the value of applications in which the FPGA would be used.

Input orders to security cell are specified with configuration bits of connection blocks. These inputs are fully ($n = m$) or partially ($m < n$) shared among LUT and the authentication module; however, the order of inputs for LUT and authentication module is different. This feature enables us to have distinct bitstream in LUT and its related authentication modules. Figure 2, shows detail of authentication path architecture. Unlike LUT, authentication module is not connected to the channels but to the obfuscation cells.

Switch boxes are promoted to play the role of obfuscation cells as well. They are triggered by the authentication modules when a violation is found out. Whenever switch boxes are triggered, they invert their input logic values and put a faulty data on the channel. In other word, switch boxes transfer either true data when the authentication signal is off, or inverted data when the authentication signal is activated.

Switch boxes are suitable to work as obfuscation modules. Since they realize the connections of wires at intersection of a vertical and horizontal channel and they are the main routing resources in the FPGAs. An important point is that any disordering in net connections in switch boxes leads to the netlist change and the created disarrangement would be propagated throughout the chip. In the MUX-based FPGAs the connections in the switch boxes are implemented through some multiplexers and some SRAM cells provide value for the select-pins of these multiplexers. We have established our implementation on the mux-based architecture which is a fundamental architecture for the modern FPGAs [11]. In order to transform either "true data" or "inverted data", 2-input multiplexers are put en route of outputs of switch boxes. We call them obf_Mux. All selection pins of these multiplexers in a switch box come from one SRAM cell that is configured by bitstream. Authentication signal changes the value stored in this cell when the violation is occurred and the altered value is remained unchanged until next reconfiguration process of the FPGA array.

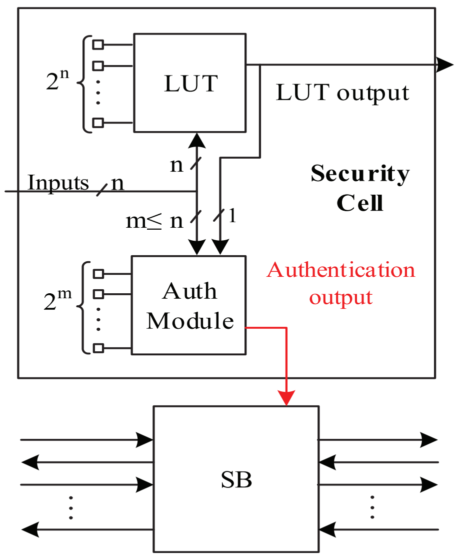

Figure 2. Authentication path architecture.

3.2 Proposed Design Flow

Figure 3 shows the overall CAD flow of implementing our approach. Solid boxes represent the unmodified steps in an standard FPGA CAD flow. The dotted boxes are additional integrated steps. The dark gray boxes show the name of CAD phases based on the VTR Tool. The additional phases are labeled as Integrated.

The CAD flow (as shown in Figure 3) starts with synthesizing of RTL design. ODIN phase converts a Verilog Hardware Description Language (HDL) design into a flattened netlist consisting of logic gates and black boxes that represent heterogeneous blocks. Next, the ABC synthesis package is used to perform technology independent logic optimization of each circuit and then each circuit is technology-mapped into LUTs and flip flops. VPR then packs this netlist into more coarse-grained logic blocks, places the circuit, and routes it. Authentication modules are labeled as

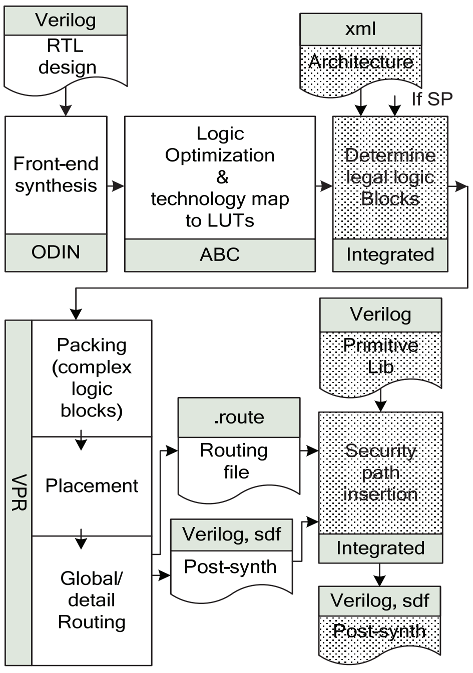

Figure 3. Security path insertion flow.

Security Path Insertion Algorithm	
Step1:	Read routing output of VPR, primitive library file in verilog and post-synthesis netlist output of VPR
Step2:	Make FPGA Graph (netlist, channels, SBs)
Step3:	Generate Auth Bitstream
Step4:	Make auth-network
Step4-1:	Aggregate-Authentication-CLB()
Step4-2:	Aggregate-Authentication-SB()
Step4-3:	Setting-Obfuscation-Module()
Step5:	Update Post-synthesis

Figure 4. Security Path insertion algorithm

illegal nodes for placement, thus VPR placement does not place the main design functions on authentication modules. When the VPR process has been finished some output files would be provided, two of them are used as input to the security path insertion phase. One file describes the circuit's routing while another one is the post-synthesis netlist, which consists instantiated modules of the primitives in the circuit such as LUTs, Interconnects, IOs and etc.[12]. This post-synthesis netlist just includes the main functions of the design and it will be revised with the security path.

Security path requires some primitive modules that do not exist in the primitive library. The primitive library is updated with the Verilog description of these modules like: XOR, OR and NAND gates, special multiplexer module to equip switch boxes with obfuscation capability. Security path insertion phase design authentication functions for each authentication module which is left vacant in placement phase. Next, the authentication network should be inserted and also this network should be connected to the switch boxes. Although the architecture of authentication network is pre-designed in FPGA architecture, the final Verilog netlist which is used to perform functional simulation should be updated with the special primitives and the required netlist alteration to implement their connections. Figure 4 shows the security path algorithm.

In the following paragraphs, details of this algorithm are described.

Step 1: The required input files are read and parsed. Some new data structures are added to the traditional Tool. These Data structures are instantiated and get value during parsing step.

Step 2: In this step a resource graph is generated consisting of four matrixes, one represents the CLBs (including security cells), one for switch boxes (obfuscation modules), and two of them show the channels X and Y. LUTs and interconnects that are defined in the input netlist (resources that are occupied for main functions of the design) are mapped to this graph. In this step, a top view is drawn to show mapping of the design on the FPGA resources.

Step 3: For each LUT in the netlist file, an authentication module is generated and it is placed to the respective module.

Step 4: In this step, the primitive modules that make authentication network are instantiated. They are also connected to the netlist. Generating authentication network has tree general subroutines that are introduced as the following:

- **Aggregate-Authentication-CLB():** In this step, a single authentication signal is made out of all authentication signals that are generated inside a CLB. This activity is done by inserting some OR gates that receive authentication signals as inputs and finally generate a single authentication signal for that CLB called CLB_authentication.
- **Aggregate-Authentication-SB():** This function finds the CLBs that are neighbor with a given switch box then their CLB_authentication signals are given to the inputs of some other OR gates to form a single authentication called SB_authentication signal.
- **Setting-Obfuscation-Module():** This function connects SB_Authentications to the switch boxes to provide the selection inputs of obf_Mux multiplexers.

Step 5: In this step the netlist file generated by the VPR is updated with the security path instances and the final post-synthesis netlist file is generated in

Verilog format.

4 Experimental Results

The self-authenticate path mechanism is added to the VTR framework 7. In this version, VPR tool generates the Verilog and SDF files for the post-synthesized circuit. The Verilog file can be used to perform functional simulation and the SDF file enables timing simulation of the post-synthesized circuit. The Verilog file contains instantiated modules of the primitives in the circuit. Actually the Verilog file represents the mapped design on given FPGA architecture. This framework runs on an Intel 2.7 GHZ Quad Core CPU with 4GB RAM platform. The self-authentication path is implemented on the single module designs provided by the VTR frame work in .blif format files. Table 1, shows the specifications of attempted benchmarks.

Table 1. Statistical characteristics of attempted benchmarks.

Benchmark	CLB#	LUT#	Nets# routed on channels
c6288	11 x 11	586	1244
apex4	16 x 16	1262	2507
alu4	18 x 18	1522	3291
apex2	20 x 20	1878	3766
pdc	31 x 31	4575	10550

4.1 Security Improvement

Table 2, illustrates the security status that authentication path provides. It indicates 100% violation coverage of the provided self-authentication path mechanism, Our experimental scenario consists of 4 steps. We start our test with very little amount of manipulation size in the bitstream (1.5%) to test the capability of our mechanism strictly. We have applied this amount of manipulation in the bitstreams with two steps: distributed and locally. In the first step we have changed one configuration bit in each LUTs (distributed manipulation) and then in second step this amount of bit changes is applied to some LUTs; somehow roughly half of their configuration bits are altered (localized manipulation). In the given architecture, the size of all LUTs are equal and each LUT has 64 configurations bit. Therefore, to have distributed alteration with 1.5% fault rate, each LUT should have only one corrupted bit ($1/64 = 1.5\%$). Also it is worth noting that in above experiments, bits and LUTs in above steps are randomly selected for fault insertion. In the third step we have decreased the tamper size to 0.75% of bitstream. In the fourth step only one LUT is altered, the LUT is selected randomly and roughly 50% of its configuration bits are altered.

In all tests the aggregated authentication signals of CLBs and Switch boxes are monitored in the functional simulation, to determine event times in which the tamper is activated.

In Table 2, two criteria are measured to show the obtained security level by authentication path insertion.

- Authentication coverage: column “Auth the number of times that tamper is activated and the output vector is deviated from the original one. As it is shown in all input test patterns and all benchmarks with various fault amount insertions, no correct output is generated whenever a tamper is activated in the circuit.
- Hamming distance: column “HD shows the deviation amount of output from the original one. The user would notify the violation when there is an extremely deviation among output vectors and correct ones. It is possible to drive an extra output pin for authentication output, to show if any violation has been detected or not, however it imposes noticeable cost on circuit implementation. Therefore, in the proposed mechanism, rather than making an error output signal, we propagated the discovered fault throughout the circuit to make them discoverable rapidly and certainly. In this case, greater distance between obfuscated and correct output indicates better obfuscation. However, it is safer if the correlation between the corrupted and original outputs is minimized. In this way attackers would face with the maximum ambiguity to obtain knowledge about the obfuscation behavior. Thus the best amount for distance corruption would be HD=50% [13]. As shown in the table, HD is very near to 50% when the fault amount of the bitstream is more than 1%.

4.2 Overheads

As explained before, switch boxes inverts the correct logic and transmit it to the channel whenever the authentication modules report deviation trough SB_authentication signal to the switch boxes. If each channel of an FPGA has n tracks, it is acceptable to imagine that in each channel $n/2$ of tracks are in same direction also we assumed each track is unidirectional. On the other hand, each switch box needs one inverter (inv) and a multiplexer (m2x1) at its each output net to perform obfuscation feature. Also, the selection pin of these multiplexers are all connected to each other and come from a latch, where the SB_authentication signal is saved. The area overhead percentage of obfuscation feature in switch box would be: the area of obfuscation feature for each output net divided by the area of each connection in switch box. Equation 1 shows the area overhead calculation in one switch box.

Table 2. Experimental results in terms of security improvement

Benchmark	1.5% alteration				0.75% alteration		1 LUT only		
	Distributed		Localized		HD	Auth	HD	Auth	Fault
	HD	Auth	HD	Auth					Size
c6288	49	100	41	100	50	100	30	100	8.5e-4
apex4	31	100	31	100	31	100	0.5	100	4e-4
alu4	57	100	55	100	55	100	31	100	3e-4
apex2	75	100	75	100	75	100	42	100	2e-4
pdc	35	100	35	100	36	100	2	100	1e-4
Ave	49.5	100	48	100	49.5	100	21	100	<1%

$$SBAO = \frac{2n \times (A_{INV} + A_{MUX2x1}) + \frac{A_{LATCH}}{2n}}{2n \times (A_{MUX4x1} + 2 \times A_{LATCH})} \qquad (1)$$

where

$$A_{INV} = u(baseunit),\ A_{MUX2x1} = 4u,$$
$$A_{LATCH} = 5u \ \ and \ \ A_{MUX2x1} = 10u \qquad (2)$$

therefore

$$SBAO = \frac{1}{4} + \frac{1}{(80 * n^2)} \approx 1/4 = 25\% \qquad (3)$$

In the worst case where $(n = m)$, authentication module's area is equal to an LUT size. However, because the adder, FF circuitry and output buffers is unchanged and comprises roughly half the area, the overall area of a standard LE increases by roughly 50% [14]. However, the (60% ,20%) of total area in each FPGA is dedicated respectively to the Switch Boxes and logic elements [15], therefore the total area overhead is as shown in Equation 4.

$$AO = ((60\% \times total\_area)25\%)$$
$$+ ((20\% \times total\_area) \times 50\%) \qquad (4)$$

As mentioned before, authentication modules are working in parallel with the data path flow. The delay of authentication modules is not effective in the delay of the data flow. Also the obfuscation latch delay does not affect in data path delay. Since there is no difference in how many Nano second later the functional obfuscation would take place. Thus, the only affective element is the m2x1 which is en the route of switch boxes outputs. This mux is on the way of data path whatever original data or inverted one is going to the output. Thus, the total delay overhead would be equal to the delay caused by locking modules on the critical nets in each benchmark. The delay overhead is illustrated in Table 3.

In this table first column shows the benchmarks, the second column SB# represents the number of switch boxes that the critical path in each benchmark has passed. In each switch box the critical path has passed from one locking module that its delay is 2.5E-11 ns (k6_N10_40nm.xml). "DO shows the overhead percentage that is implied to the critical path because of lock modules.

Table 3. Delay overhead

Benchmark	SB	DO
c6288	89	21.80%
apex4	89	22.45%
alu4	101	26.37%
apex2	127	23.83%
pdc	165	24.75%
Average		23.84%

The power consumption of locking logic can be estimated by SPICE simulation or mathematical power analysis. Our analyses show that this part of power consumption is very smaller than power overhead of added LUT (the size of obfuscation module is equal to the size of a mux2x1). Therefore, we extracted the total power consumption of the benchmarks before and after authentication path insertion which is related to the added LUTs. Table 4 compares the power consumption of benchmarks before and after authentication modules are inserted.

Table 4. Power overhead.

Benchmark	Before Auth. Insertion (mW)	After Auth. Insertion (mW)	Power Overhead%
c6288	2.956	3.276	11%
apex4	7.382	8.219	11%
alu4	9.333	12.05	29%
apex2	10.95	11.33	4%
pdc	20.81	28.0	35%
Average			18%

5 Conclusion

In this paper we proposed a self-authentication mechanism to authenticate functional units of the designs at run time. This mechanism is implemented as an authentication path in parallel with the data path which is automatically added to the design by CAD tool, right before bitstream is generated. The functionality of the design is obfuscated if any violation is detected in the circuit. This mechanism is self-oriented and does not need key management protocol and storage element. The security experimental results show that there is 100% coverage for tampers which are affective in the circuit functionality.

As a future contribution, it is possible to find a reasonable tradeoff between area-power and delay overheads and authentication coverage. Also, we are working on an authentication-aware placement and routing algorithms to confine authentication costs to security required portion of the IPs.

References

[1] S. M. Trimberger, Jason J. Moore, "FPGA security: Motivations, features, and applications", Invited paper in *Proceedings of the IEEE*, Vol. 102, Issue 8, pp. 1248 - 1265, Aug. 2014.

[2] R. S. Chakraborty, I. Saha and A. Palchaudhuri, "Hardware Trojan insertion by direct modification of FPGA configuration bitstream", In *IEEE Design and Test*, Vol. 30 Issue 2, pp. 45-54, 2013.

[3] J. Note and E. Rannaud, "From the bitstream to the netlist", In *proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays (FPGA'08)*, pp. 264, 2008.

[4] P. Swierczynski, M. Fyrbiak and C. Paar et al, "FPGA Trojans through Detecting and Weakening of Cryptographic Primitives", In *TCAD*. pp. 1-13, 2015.

[5] T. H. Vu and N, V. Cuong, "A framework for secure remote updating of bitstream of runtime reconfigurable embedded platforms", In *Communications and Electronics (ICCE)*, pp. 471 - 476, 2012.

[6] A. Moradi, A. Barenghi and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks", In *ACM Conference on Computer and Communications Security*, pp. 111-123, 2011.

[7] Y. Hori, T. katashita and A. Sasaki et. al, "Electromagnetic side-channel attack against 28-nm FPGA device", In *Pre-proceedings of WISA*, 2012.

[8] E. Mudler, P. Buysschaert and P. Delmotte, "Electromagnetic analysis attack on an FPGA implementation of an Elliptic curve cryptosystem", In *Computer as a Tool (EUROCON)*, pp. 1879 - 1882, 2005.

[9] S. Trimberger, "Trusted design in FPGAs", In *Proceedings of International of Design Automation Conference* , pp. 5-8. 2007.

[10] S. Trimberger and J. Moore, "FPGA security: From features to capabilities to trusted systems", In Proc. 51st Annual Design Automation Conf., 2014.

[11] U. Farooq, H. Parvez, H. Mehrez, and Z. Marrakchi, "Exploration of heterogeneous FPGA architectures", In *International Journal of Reconfigurable Computing* , 2011 .

[12] J. Rose, J. Luu, and J.Goeders, "VTR 7.0: next generation architecture and CAD system for FPGAs", In ACM Transaction on Recongurable Technology and Systems (TRETS), Vol. 7, No. 2, 2014.

[13] J. Rajendran, H. Zhang, R. Karri et al, "Fault analysis-based logic encryption", In IEEE *Transactions on computers*, Vol. 64, No. 2, pp. 410-424, 2015.

[14] M. Hutton, D. Lewis, B. Pedersen, J. Schleicher, R. Yuan, G. Baeckler, A. Lee, R. Saini, and H. Kim, "Fracturable FPGA logic elements", Technical Report, Available on `http://www. altera. Com/literature/cp/cp-01006. pdf, 2006.`

[15] U. Farooq, Z. Marrakchi, H. Mehrez, "Tree-based Heterogeneous FPGA Architectures", Springer 2012.

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.