Zhang JL, Wang WZ, Wang XW *et al.* Enhancing security of FPGA-based embedded systems with combinational logic binding. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(2): 329–339 Mar. 2017. DOI 10.1007/s11390-017-1700-8

Enhancing Security of FPGA-Based Embedded Systems with Combinational Logic Binding

Ji-Liang Zhang [1,2], *Member, CCF, ACM, IEEE*, Wei-Zheng Wang [3], Xing-Wei Wang [1], and Zhi-Hua Xia [4]

[1] *Software College, Northeastern University, Shenyang 110169, China*

[2] *Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education Nanjing 210096, China*

[3] *Department of Computer and Communication Engineering, Changsha University of Science and Technology Changsha 410014, China*

[4] *School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China*

E-mail: zhangjl@mail.neu.edu.cn; greaquer_w@yeah.net; wangxw@mail.neu.edu.cn; xia_zhihua@163.com

Received April 17, 2016; revised November 25, 2016.

Abstract With the increasing use of field-programmable gate arrays (FPGAs) in embedded systems and many embedded applications, the failure to protect FPGA-based embedded systems from cloning attacks has brought serious losses to system developers. This paper proposes a novel combinational logic binding technique to specially protect FPGA-based embedded systems from cloning attacks and provides a pay-per-device licensing model for the FPGA market. Security analysis shows that the proposed binding scheme is robust against various types of malicious attacks. Experimental evaluations demonstrate the low overhead of the proposed technique.

Keywords cloning attack, reverse engineering, FPGA (field-programmable gate array) security, hardware security

1 Introduction

1.1 Motivation

Hardware security has become more and more important in current information security architecture[1]. Field-programmable gate array (FPGA) as a kind of programmable hardware can be configured by end users to implement any digital circuits. Nowadays, FPGAs have become integral parts of embedded systems and many embedded applications such as computer vision, digital signal processing and many others due to the continuous improvement in quality and the decrease of production cost[2]. However, unlike ASIC (application specific integrated circuit), an FPGA-based system is essentially a binary bitstream file that can be easily copied by the third party or end users without reverse engineering, which brings serious losses to system developers and reduces the market share of their products. Hence, cloning has been considered to be the most common security vulnerability of volatile FPGAs[3]. Moreover, system developers cannot determine how many FPGA devices their products have been authorized run on, which forces them to adopt an up-front licensing model where a customer obtains unlimited use of an FPGA-based system on any FPGAs for a single relatively large payment. Current pricing model is fundamentally unsuited to the industry FPGA market since the basic motivation for using an FPGA rather than an ASIC is to trade off a higher per-unit cost to avoid a large up-front NRE (non-recurring engineering) payment[4]. Hence, in FPGA design security field, developing the anti-cloning techniques and providing a more suitable pay-per-device licensing model are urgent.

Regular Paper

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61602107, 61572123, 61303042, and the Fundamental Research Funds for the Central Universities of China under Grant No. N161704006.

1.2 Limitations of Prior Art

Recent studies on the FPGA design security aim to protect FPGA designs by watermarking/fingerprint and bitstream encryption.

In watermarking/fingerprint techniques[5], an encrypted watermark or fingerprint is embedded into FPGA designs to represent the ownership. When intellectual property (IP) disputes occur, its owner can ask a trusted third party (TTP) to recover the signature from the stolen IP core, which can potentially address the cloning issue. However, watermarking/fingerprint techniques are typical passive techniques. They cannot actively prevent the FPGA design from cloning. In addition, the watermark/fingerprint embedded in the FPGA design is more likely to be tampered with and covered than an ASIC, making the FPGA IP protection more difficult.

Bitstream encryption①②[6-7] is to encrypt the configuration bitstream and then load it into an FPGA. It is the most popular and efficient intellectual property protection technique against the direct cloning of FPGA bitstreams for high-end SRAM (static random-access memory) FPGA devices. For example, Xilinx Virtex II and Spartan/Virtex-6 FPGAs support bitstream encryption by storing the decryption key in a dedicated battery-backed SRAM or a one-time programmable eFuse register. However, FPGA design is stored as a bitstream in the external memory (EEPROM). Once an FPGA is powered up, the bitstream is loaded to configure the FPGA. During the loading, it is easy to clone the bitstream by wiretap since cloning requires no more than a logic analyzer and a competent technician[8]. Simple bitstream encryption is also a kind of passive bitstream protection technology. It cannot actively restrict FPGA-based systems running on specific FPGA devices. More importantly, it is a costly and heavy protection technique that is only deployed on the high-end FPGA devices and also not appropriate for resource-limited embedded systems due to the use of on-chip cryptographic decryption module and the permanent secure key storage. In addition, such permanent nonvolatile memory is often vulnerable to invasive attacks[9].

Bitstream encryption is a comprehensive solution to protect hardware intellectual property. However, when the FPGA configuration contains programmable components (such as a soft-core processor), the software intellectual property implemented on top of that soft-core processor requires separate protections. Hence, Gora *et al.*[10] proposed to bind software IPs by leveraging the qualities of a physical unclonable function (PUF) and a tight integration of hardware and software security features.

Recently, IC metering[11] has attracted lots of attentions in the ASIC security field. It can enable the design house to gain the post-fabrication control by the passive or active control of the number of produced ASIC chips. The work in [12] proposed a typical combinational logic metering technique, named EPIC, by randomly inserting XOR or XNOR, as shown in Fig.1. One of the inputs to an inserted gate is the functional input in the design and the other is one-bit key input. EPIC is the closest work to our approach. However, EPIC cannot be applied to protect FPGA-based embedded systems because, unlike ASIC, the FPGA design is essentially a bit-file which is vulnerable to reverse engineering[13], and the heavy key management and exchange make it difficult to be applied in resource-constrained areas. In addition, it also cannot provide pay-per-device licensing for FPGA designs.

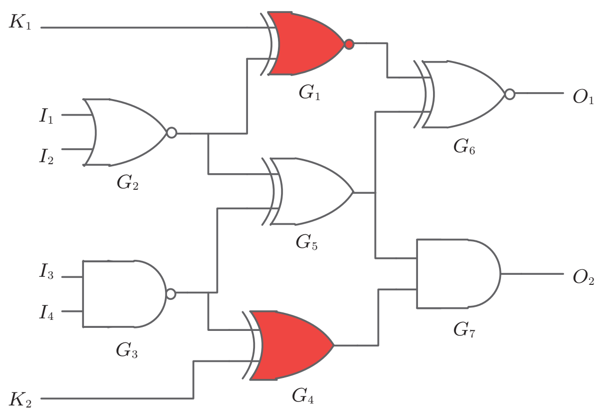


Fig.1. Circuit processed with combinational logic metering method[12].

1.3 Our Contributions

The key technique for the proposed combinational logic binding involves designing and implementing a simple combinational logic-based binding cell (BC) which is used to replace any 2×1 logic gates of the FPGA design (a 2×1 logic gate represents two-input

①Design security in Stratix III devices. https://www.altera.com/en_US/pdfs/literature/hb/stx3/stx3_siii51014.pdf, Feb. 2017.

②Using high security features in Virtex-II series FPGAs. http://www.xilinx.com/support/documentation/application_notes/xapp766.pdf, Feb. 2017.

and one-output logic gate). Without providing the correct key, the function of the design will not exhibit correctly. A PUF is used to XOR with the key of BCs to generate the chip-dependent license, which provides the pay-per-device licensing service that end users can use to buy their FPGA-based products at the low price based on the usage on authorized FPGAs instead of paying the expensive unlimited intellectual property (IP) license fees on all FPGAs. Note that the proposed combinational logic binding technique in this paper focuses on protecting FPGA-based (embedded) systems instead of hardware IP cores from cloning attacks.

The contributions of this paper are as follows.

1) The first combinational logic based binding technique for FPGA-based systems is proposed.

2) A binding protocol to provide a pay-per-device licensing model for FPGA-based system is designed.

3) Security analysis shows that the proposed binding scheme can resist potential attacks.

4) Experimental evaluations and the low overhead of the binding are demonstrated on standard benchmark circuits.

1.4 Outline of the Paper

The rest of this paper is organized as follows. The proposed new binding technique is elaborated in Section 2. Potential attacks are analyzed in Section 3. The experimental results and analysis are reported in Section 4. Finally, Section 5 concludes the paper.

2 Proposed Binding

With the increasing demands of security in various areas[14–29], cryptographic key storage has become one of the most challenging design concerns[30]. Silicon PUF is a promising solution for the challenge. PUF is a physical entity that exploits the intrinsic variation of IC manufacturing process to generate chip-unique information and is easy to evaluate but hard to predict. It has become a popular hardware security primitive for various security-related applications such as software binding[31] and key generation[32]. PUF has been successfully applied to commercial FPGAs③ and ASICs④ to uniquely identify devices and generate the device-specific key. An ideal PUF should satisfy three properties: persistent and unpredictable, unclonable, and tamper evident.

FPGA binding is to modify the original FPGA-based product to produce a bound netlist which is functionally equivalent to the former. The modified design reacts with the PUF and performs exactly the same with the function of original FPGA design as long as the correct device-dependent license is issued by the system developer. This means that only the FPGA devices authorized by the system developers can guarantee the correct functionalities. Hence, the binding can prevent FPGA-based products from cloning and provide the pay-per-device licensing. The details of the proposed binding technique are described as follows.

2.1 Design Flow of Binding

A typical FPGA design flow with the combinational logic based binding from register-transfer level (RTL) to layout is shown in Fig.2. A system developer develops an FPGA-based embedded system, which involves purchasing the third party intellectual property (IP) cores from outside design houses, integrating them, and generating the bitstream through logic synthesis, technology mapping, and placement and routing. The binding can be automatically integrated within the regular FPGA design flow without changing regular design flow to support the binding.

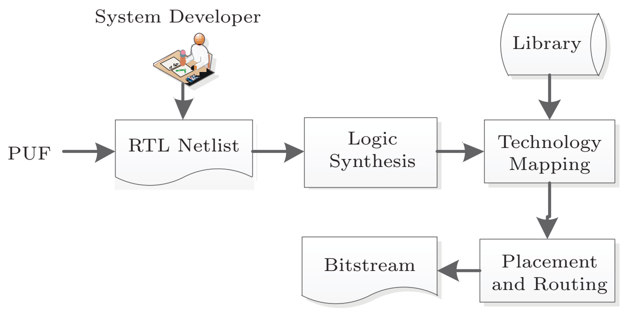


Fig.2. Typical FPGA design flow with the new binding from RTL to layout.

2.2 Binding Cell Structure

Assume a binding cell has n key inputs, we can have possible functions 2^n. For example, 1-input key binding cells can represent two different functions, as shown in Fig.3. The cell having more key bits would incur bigger overhead. In this paper, we just consider that each cell has 1 key input and produces only 1 output. The circuit schematic of the proposed 2×1 cell is shown in Fig.3,

③SmartFusion2 SoC FPGAs. http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2, Feb. 2017.

④Hardware Intrinsic Security. http://www.intrinsic-id.com/solutions/his-technology/, Feb. 2017.

where G1 and G2 denote two 2×1 logic gates and K_1 is a selection input of the Multiplexer. It can replace all 2×1 logic gates of its inputs A and B. For example, if an AND gate is replaced by a 2×1 binding cell, then we can set G1 as AND and G2 as any other 2×1 logic gates such as NAND, XOR and OR. In this case, K_1 would be set into logic-1.

2.3 Frame of Binding

The key bits of the binding cells are used to interact with the PUF response in order to generate a chip-dependent license to prevent piracy and provide the pay-per-device licensing service. An attacker with no information about the key bits of binding cells cannot compute the correct license to unlock the pirated design on unauthorized FPGAs. Hence, the designer is the only one who can issue the license to activate the chip. We must emphasize that the binding method does not depend on the PUF implemented in this paper. Any other PUFs such as SRAM PUFs[33], Arbiter PUFs[34] and RO PUFs[32] can also be applied to the binding scheme as long as they satisfy the properties described in [30].

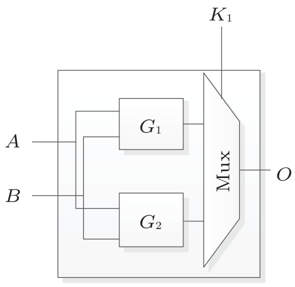


Fig.3. Structure of the proposed combinational logic based binding cell. Mux: multiplexer.

As shown in Fig.4, we use the PUF response to bind the combinational logic function of the design on specific FPGA devices. The error-corrected PUF response is used to XOR with the license to uniquely unlock the function of the FPGA design. The function of design would not perform correctly without the correct PUF response. Therefore, the circuit is kept locked until the correct license unlocks it. It should be noticed that the issued licenses can also be public and different PUF responses can be used to calculate different licenses. Additionally, the designer often computes the error correcting code (ECC) to adjust for any bit flips to the PUF response because the PUF output is hard to maintain absolutely stable due to the noise or other sources of physical uncertainty[30].

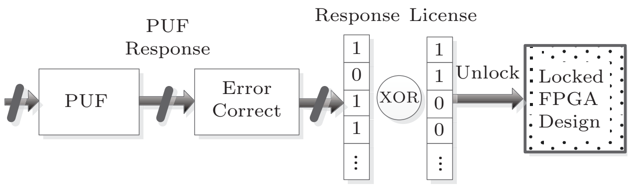


Fig.4. Unlocking the function of locked FPGA design with PUF response.

We give an example of the interaction among the key of binding cells, PUF response and license in Fig.5. Considering four binding cells in Fig.5, $K_1 \sim K_4$ are the four key bits of the binding cells which can be used to replace four 2×1 logic gates. Assume $K_1 \sim K_4 =$ "1010" and the PUF output value is "0110". To possibly activate the design, the 4-bit PUF output "0110" should be XOR'd with a 4-bit license that is able to generate the result of "1010" (in this case the license should be "1100"). The design can be correctly unlocked with the calculated license and the PUF response. The nonvolatile on-chip memories would be used to store the PUF challenges, the license, and the relevant ECC bits on each pertinent activated FPGA device. From this point on, every time when the FPGA powers on, it would automatically read the PUF challenge and ECCs and use them to unlock the FPGA design.

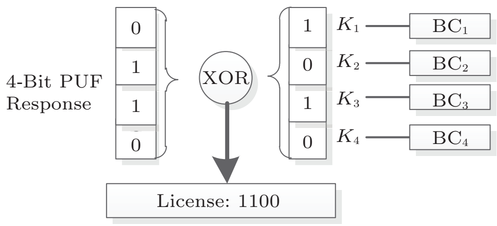


Fig.5. Generation of the license for binding.

2.4 Binding Protocol

In order to make the binding scheme work well, a binding protocol involving a system developer, an FPGA vendor, an end user and a trusted third party is needed. The protocol includes the following two steps.

1) *Enrollment.* The enrollment process is shown in Fig.6(a). First, an FPGA vendor records all FPGAs' ID, ID(FPGA). Second, the FPGA vendor sets the PUF's location constraint Loc(PUF) which is supported by Xilinx integration development kit and used to place a PUF design to a designated region. Third, the FPGA vendor downloads the PUF into all FPGAs according to Loc(PUF) to get the corresponding challenge-response pairs (CRPs). After that, the FPGA vendor sends ID(FPGA), Loc(PUF), CRPs to

the TTP. A system developer applies for the Loc(PUF) to start system development. The PUF will be integrated with the product, and finally the key of product and ID(product) are sent to the TTP.

2) *Product Licensing.* As shown in Fig.6(b), if end users would like to buy a license of the product developed by a system developer in order to obtain the permission to run on specific FPGA devices, they would send {ID(FPGA), ID(product)} to the system developer who then forwards {ID(FPGA), ID(product)} to a TTP. The TTP queries the database based on {ID(FPGA), ID(product)} to get the Key(product) and CRP (challenge-response pairs) and then calculates the licenses. The licenses will be sent to the system developer. The system developer sends the licenses to the end users. Hence, the end users obtain the permission of the purchased product running on the specific FPGA device.

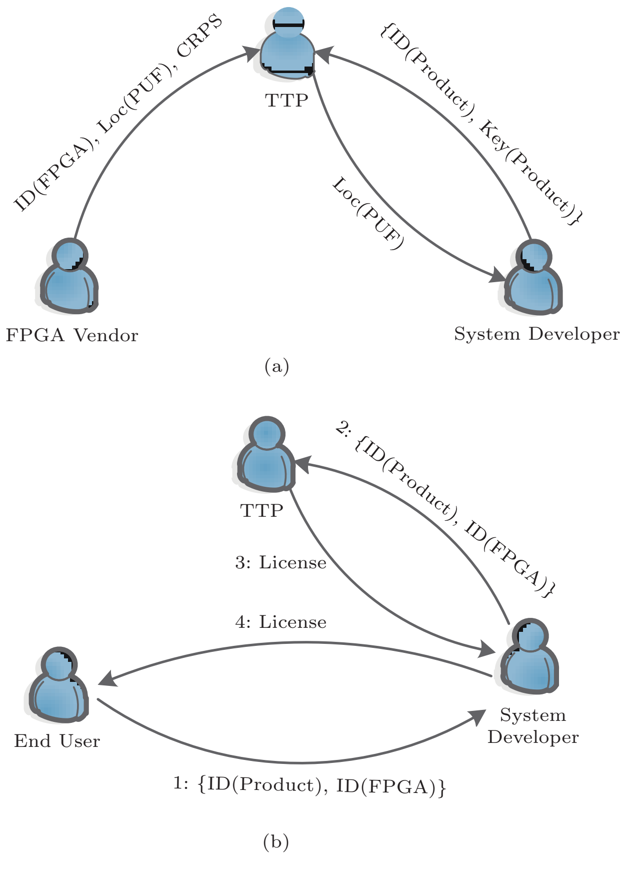


Fig.6. Binding protocol. (a) Enrollment. (b) Product licensing.

In the proposed binding protocol, the trusted party is responsible for computing the license or new license of updated FPGA systems and remotely updates the FPGA system to resist replay attacks. In our case, FPGA vendors can play the role of the third party.

2.5 Pay-per-Device Licensing

Binding can support a promising pay-per-device licensing model. This provides technical support for the product developers to sell their products in low licensing fees to buyers only for the FPGA devices they authorized. It also enables the product developers to freely distribute their products because they can ensure that the distributed products can only run on authorized FPGA devices. The binding brings a remarkable advantage that the product developers can take the full control over the use of their products and protect them from unlicensed use; the buyers who could not afford the expensive unlimited product license are now also able to obtain a number of single instances of the required products at a much lower cost.

3 Security Analysis

The proposed combinational logic based binding scheme in this paper would be compromised if an adversary can obtain the secret key or PUF response. In this section, we first give the threat model and then analyze various types of malicious attacks elaborately including reverse engineering bitstream to extract the gate-level netlist, brute-forcing the access key, side-channel attacks and replay attacks.

3.1 Threat Model

Attackers' goal is to reverse engineer and/or pirate FPGA designs. This paper assumes that attackers have the following abilities.

• Attackers can extract the gate-level netlist using the reverse engineering technique.

• Attackers can modify and simulate the bound gate-level netlist.

• Attackers can purchase legal licenses from the system developers.

• Attackers know all logic functions that a binding cell can implement.

3.2 Analysis

Our proposed combinational logic binding technique can resist the following potential attacks.

3.2.1 Reverse Engineering Attacks

In order to obtain the secret key, the adversary has to reverse engineer the FPGA bitstream. It is well known that FPGA vendors have taken a business position that they will not reveal the specification of their configuration streams, specifically to complicate the task of reverse engineering and thus protect the investment of their customers. Although reverse engineering FPGA bitstreams is difficult, for large organizations it is quite feasible. For example, this security-by-obscurity approach was broken at least 10 years ago. NeoCAD Inc. was able to reverse engineer bitstreams for the Xilinx XC4000 series devices through a directed investigation of the output produced by the Xilinx backend tools[35]. Therefore, in order to demonstrate the high security of our binding scheme, we prove that our proposed combinational logic based binding scheme is secure even if an adversary is able to reverse engineer any FPGA design from bitstreams to netlists.

We give a simple example of gate-level netlist of a design with our binding technique. As shown in Fig.7, a circuit is bound by replacing an AND gate with a binding cell. The primary inputs I_1~I_5 are the functional inputs; O_1~O_2 primary outputs are the functional outputs; G1 is a binding cell which denotes an AND gate and an XOR gate; K_1 is the key bit of G1. The function of this circuit would be obfuscated without knowing the key bits of binding cells even if an adversary is able to obtain the gate-level netlist by reverse engineering. Therefore, in our binding scheme, the adversary has to perform the following brute-forcing attacks since he/she does not know the logic function of binding cells.

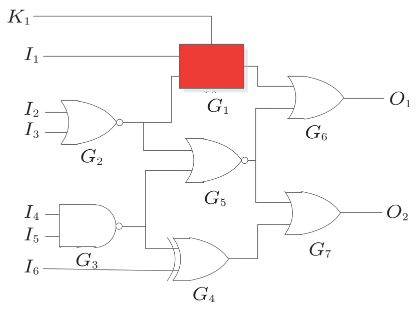


Fig.7. Gate-level netlist with a binding cell.

3.2.2 *Brute-Forcing the Access Key*

The steps of brute-forcing attacks are shown as following:

1) buying a license of a product from the system developer;

2) obtaining the netlist of the product by reverse engineering;

3) generating random input patterns for each possible key bit value of all binding cells;

4) simulating the input patterns and obtaining the output O_1;

5) applying these patterns on the product which is programmed into the authorized FPGA and obtains the output O_2;

6) repeating steps 3~5 until $O_1 = O_2$, the attacker would be successful to unbound the product.

A binding cell is designed to implement any 2×1 logic functions. The complexity of this brute-forcing attack can be 2^n, where n is the number of binding cells. Hence, such brute-forcing attack is difficult to break the binding scheme proposed in this paper even if the design has been reverse engineered.

3.2.3 *Side-Channel Attacks*

Side-channel attack is a powerful cryptanalysis technique that statistically analyzes the time, power consumption or electromagnetic emanation of the cryptographic devices to gain knowledge about integrated secrets. The adversary might perform side-channel attacks[36] to get the PUF response or secret key. In our binding scheme, the secret PUF response and key are only used to unlock the design at boot time and will be immediately cleared after use. This can avoid tapping the PUF response or key since the secret response is never present once the design is unlocked. Hence, our proposed combinational logic based binding technique would be less vulnerable to side-channel attacks than existing FPGA bitstream encryption techniques, where the permanent secure storage is used to store cryptographic keys.

3.2.4 *Replay Attacks*

FPGA replay attack, where an attacker downgrades an FPGA-based system to the previous version with known vulnerabilities, is particularly dangerous for FPGA-based system security because attackers can effectively preclude security-critical updates by replaying the previous FPGA configurations. Zhang *et al.*[37] proposed the concept of reconfigurable PUF-FSM binding that reconfigures both the PUF and the FSM structure to efficiently resist the attack. Similarly, in this paper, we can also reconfigure the PUF and the combinational logic binding cell to defeat the attack. We can change the location constraint information of the location-based reconfigurable PUF[37] to reconfigure it. The final goal of reconfiguring the binding structure is to change the key of the previous binding cells, which can be easily implemented by changing previous inserted or replaced binding cells in the FPGA design. For example, the system developers can randomly insert the binding cell and replace inverters with the binding cells again when they update their product from version 1 (V1) to version 2 (V2). Therefore, our proposed combinational logic based binding scheme can also resist the FPGA replay attacks.

4 Experimental Results

We have performed a set of experiments to evaluate the overhead of our proposed new binding technique. The experiments include the implementation of a delay-based PUF and the 2×1 binding cells on the 28 nm Xilinx Zynq-7000 FPGAs.

4.1 Overhead of PUF

A number of different PUF structures implemented on FPGAs[38-42] have been proposed in the past decade. For example, a 64-bit delay-based PUF[43] consumes 64 Slices on Xilinx Virtex-5 FPGAs. This PUF is designed in hardware description language (HDL) and hence can be automatically integrated with our binding scheme. We can also use an FPGA intrinsic PUF[42] for combinational logic binding, which incurs zero overhead. Hence, the implemented PUF incurs low and even zero overhead.

4.2 Overhead of Binding Cells

The experiments were performed on the circuits, which are described in Verilog format, from the ISCAS benchmark. The modification for binding is implemented using the C++ language. The original and modified designs are synthesized and implemented on the Xilinx Zynq-7000 FPGA using the Xilinx ISE 14.1 design suit.

We selected 2×1 logic gates for replacement using two methods, random method and timing-driven heuristic method. The random method is to randomly select the logic gates in the original FPGA design. The timing-driven heuristic method is to select the logic gates as many as possible in the non-critical paths of the original FPGA design. Table 1 gives the original statistics from ISCAS benchmark circuits. The first column denotes the benchmark circuits. Columns 2∼3 demonstrate the original synthesis statistics for resources and delay. We selected 5% logic gates in ISCAS benchmark circuits for logic binding. Hence, we give the number of binding cells (5%) used for replacement in column 4.

Table 2 shows the resource and the delay overhead for the original ISCAS benchmark circuits and the modified ISCAS benchmark circuits with the random method and the timing-driven heuristic method. Resources overhead is denoted by the increased "number of slice LUTs". We used the timing analyzer to analyze timing overhead which is measured by the minimum period degradation. Columns 2∼3 show the resource and the delay of binding cells using random selection respectively, and columns 4∼5 show the corresponding resource overhead (ΔR) and delay overhead (ΔD) respectively. Columns 6∼7 show the resource and the delay of binding cells using timing-driven heuristic method respectively, and columns 8∼9 give the corresponding resource and delay overhead respectively. The resource and the delay overhead are even negative in some instances. A negative percentage implies that our method has actually improved the performance. We can see from Table 2 that the average resource and the average delay overhead are 10.62% and 15.56% for the random method respectively, 15.21% and 0.16% for the heuristic method, respectively. Hence, the performance overhead is extremely low for our proposed new binding technique with the heuristic method.

Table 1. Original Statistics of ISCAS Benchmark

Circuit	Original Statistics		Number of Binding
	Number of LUTs	Delay (ns)	Cells (5%)
s208_1	11	1.270	2
s344	22	1.771	4
s349	22	1.771	5
s386	27	1.392	4
s400	31	1.623	5
s420_1	29	1.445	4
s444	31	1.618	5
s526n	28	1.821	5
s2641	43	1.723	6
s713	43	1.723	6
s820	60	2.278	8
s832	62	2.279	8
s838_1	66	1.956	8
s1196	110	1.788	15
s1238	112	1.603	15
s1423	209	2.957	17
s1488	116	1.827	15
s1494	116	1.852	15
s5378	345	1.959	35
s9234	239	2.803	28

Fig.8 and Fig.9 show the resource and the delay overhead of replacing different numbers of logic gates

Table 2. Resources and Timing Overheads of Binding Cells

Circuit	Random Method				Timing-Driven Heuristic Method			
	Number of LUTs	Delay (ns)	ΔR (%)	ΔD (%)	Number of LUTs	Delay (ns)	ΔR (%)	ΔD (%)
s208_1	12	1.695	9.09	33.46	13	1.281	18.18	0.87
s344	24	2.170	9.09	22.53	27	1.830	22.73	3.33
s349	25	2.177	13.64	22.92	27	1.775	22.73	0.23
s386	31	1.826	14.81	31.18	34	1.452	25.93	4.31
s400	36	2.129	16.13	31.18	38	1.618	22.58	−0.31
s420_1	35	1.804	20.69	24.84	37	1.435	27.59	−0.69
s444	30	2.002	−3.23	23.73	35	1.606	12.90	−0.74
s526n	37	1.502	32.14	−17.52	37	1.502	32.14	−17.52
s2641	46	2.059	6.98	19.50	45	1.723	4.65	0.00
s713	41	1.834	−4.65	6.44	45	1.803	4.65	4.64
s820	68	2.042	13.13	−10.36	68	2.042	13.33	−10.36
s832	74	2.157	19.35	−5.35	74	2.157	19.35	−5.35
s838_1	72	2.348	9.09	20.04	79	1.959	19.70	0.15
s1196	116	1.788	5.45	0.00	116	1.788	5.45	0.00
s1238	113	2.081	0.89	29.82	123	1.670	9.82	4.18
s1423	189	3.473	−9.57	17.45	218	3.086	4.31	4.36
s1488	134	2.073	15.52	13.46	133	1.917	14.66	4.93
s1494	150	2.132	29.31	15.12	127	1.925	9.48	3.94
s5378	357	2.461	3.48	25.63	356	2.049	3.19	4.59
s9234	265	3.001	10.88	7.06	265	2.875	10.88	2.57
Average			10.62	15.56			15.21	0.16

with binding cells using the timing-driven heuristic method. It can be seen that the resource and the timing overheads are nonlinear due to the optimization of the circuits during synthesis. Also it can be seen that the performance (delay) overhead is very low (less than 5%) and almost remains unchanged with the number of logic gates increasing.

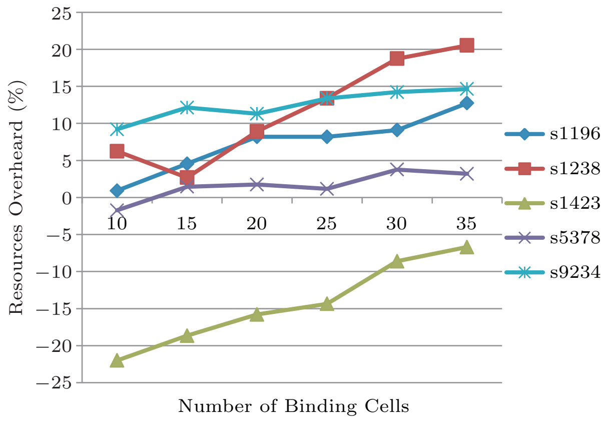


Fig.8. Resources overhead trend on benchmark circuits with different numbers of binding cells.

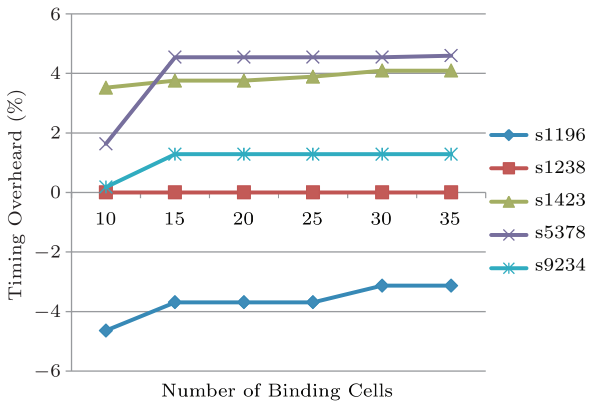


Fig.9. Timing overhead trend on benchmark circuits with different numbers of binding cells.

It should be noted that the performance overhead of our proposed binding cells should be zero if we avoid the delay overhead. The longest critical path delay (LCPD) of the benchmark circuits is analyzed to estimate the delay overhead. The binding is implemented using the timing-driven heuristic algorithm to avoid the longest critical path. Table 3 gives the original statistics reported on the ISCAS benchmark. We can see from Table 4 that the experimental results demonstrate the zero performance overhead of the proposed new binding method with the heuristic method. The average area (ΔA) and the power overhead (ΔP) are 8.48% and 13.41% for the heuristic method, respectively.

Table 3. Original Statistics Reported on the ISCAS Benchmark

Circuit	Number of Cells	Area	Delay (ns)	Power (μW)
s208_1	41	78.204	0.47	7.135 8
s298	72	132.734	0.47	13.040 5
s344	83	144.970	0.57	14.165 0
s349	83	145.502	0.56	14.070 4
s382	90	183.008	0.55	17.507 9
s386	75	96.824	0.44	7.023 0
s444	91	184.339	0.50	17.516 5
s526	105	199.234	0.54	17.865 7
s641	111	177.688	0.96	14.729 5
s713	111	177.688	0.96	17.757 3
s832	152	176.092	0.60	7.023 8
s1196	294	376.124	0.78	23.623 6
s1238	303	383.572	0.81	23.393 6
s1423	344	678.832	2.43	67.239 6
s5378	708	1 473.370	0.83	167.367 9
s9234	561	1 246.740	1.05	108.710 1
s13207	1 038	2 756.030	0.70	286.608 8
s15850	680	1 630.850	1.11	160.345 6

The proposed binding scheme can be automatically integrated within the regular FPGA design flow without changing the flow, and placement and routing can be performed automatically by EDA tools. Inevitably, additional routing overhead would be incurred due to the inserted binding cells. Our experimental results show that all FPGA designs with the proposed binding technique can be routed successfully and as discussed above: if system developers can hold the critical path information, there would be no delay overhead.

5 Conclusions

Current FPGA-oriented hardware security techniques such as watermarking/fingerprint and encryption cannot actively prevent FPGA design from directly cloning, encryption-based techniques are not appropriate for resource-limited embedded environments, and current pricing model is fundamentally unsuited to the industry FPGA market. In order to resolve these problems, in this paper, we proposed a new binding technique for FPGA-oriented hardware security to address these issues. The key idea is to replace any 2×1 logic gates with a binding cell at RTL or gate-level of FPGA design flow, and the PUF response is used to XOR with key bits of binding cells to generate a device-dependent license to prevent piracy and meanwhile provide a promising pay-per-chip licensing scheme which holds considerable advantages for system developers. Security analysis showed that the new technique can resist reverse engineering attacks, brute-forcing attacks, side-channel attacks and replay attacks to a certain extent. Experimental results on the implementation of the binding scheme showed that a 64-bit delay-based PUF utilizes only 64 Slices on Xilinx Virtex-5 FPGAs (if an FPGA intrinsic PUF is employed, zero overhead

Table 4. Overheads on the ISCAS Benchmark

Circuit	Number of Binding Cells (5%)	Area	Delay (ns)	Power (μW)	ΔD (%)	ΔA (%)	ΔP (%)
s208_1	2	84.056	0.47	7.474 0	0	7.48	4.74
s298	4	143.374	0.47	13.661 5	0	8.02	4.76
s344	4	156.674	0.57	16.318 3	0	8.07	15.20
s349	4	157.783	0.56	15.179 9	0	8.41	7.89
s382	5	196.574	0.55	18.344 0	0	7.41	4.78
s386	4	108.528	0.44	10.137 9	0	12.09	44.35
s444	5	198.436	0.50	20.147 0	0	7.65	15.02
s526	5	214.396	0.54	18.414 2	0	7.61	3.07
s641	6	195.244	0.96	15.943 5	0	9.88	8.24
s713	6	195.244	0.96	16.886 1	0	9.88	14.43
s832	8	199.500	0.60	10.562 7	0	13.29	50.38
s1196	15	420.546	0.78	27.347 8	0	11.81	15.76
s1238	15	427.994	0.81	26.753 9	0	11.58	14.36
s1423	17	732.032	2.43	71.632 9	0	7.84	6.53
s5378	35	1 558.228	0.83	170.055 3	0	5.76	1.61
s9234	28	1 301.272	1.05	125.039 5	0	4.37	15.02
s13207	52	2 908.710	0.70	306.439 8	0	5.54	6.92
s15850	34	1 728.734	1.11	173.772 5	0	6.00	8.37
Average					0	8.48	13.41

will be brought) and heuristically replacing 5% logic gates using the binding cells incurs 15.21% resources overhead and only 0.16% performance overhead on average for standard ISCAS benchmark circuits using the timing-driven method. If system developers can hold the critical path information, our proposed combinational logic based binding method would have no performance penalties.

Physically cloning attacks and machine-learning attacks are two kinds of potential attacks for the PUF. Physically cloning attacks have been reported successfully for SRAM PUFs, but other PUFs such as RO PUFs, Arbiter PUF, Glitch PUF and so on have not been reported to this date[37]. Machine-learning attacks (modeling attacks) have been used to model some strong PUFs with high prediction rate, but they need a huge amount of PUF CRPs during the learning phase[44]. Therefore, this attack will not be effective to weak PUFs such as the PUF[43] used in this paper. Moreover, in our binding scheme, the secret PUF response is ephemeral and will be immediately cleared after use. This can resist tapping PUF responses since the secret response is never present once the FPGA-based system is unlocked.

References

[1] Lv Y, Zhou Q, Cai Y *et al.* Trusted integrated circuits: The problem and challenges. *J. Comput. Sci. Technol.*, 2014, 29(5): 918-928.

[2] Fu H, Gan L, Clapp R *et al.* Scaling reverse time migration performance through reconfigurable dataflow engines. *IEEE Micro*, 2014, 34(1): 30-40.

[3] Zhang J, Qu Q. A survey on security and trust of FPGA-based systems. In *Proc. International Conference on Field-Programmable Technology* (*ICFPT*), Dec. 2014, pp.147-152.

[4] Kean T. Cryptographic rights management of FPGA intellectual property cores. In *Proc. ACM/SIGDA Symp. Field-Programmable Gate Arrays* (*FPGA*), Feb. 2002, pp.113-118.

[5] Qu G, Potkonjak M, Stojcev M. Intellectual Property Protection in VLSI Designs: Theory and Practice. Kluwer Academic Publishers, 2003.

[6] Hori Y, Satoh A, Sakane H *et al.* Bitstream encryption and authentication with AES-GCM in dynamically reconfigurable systems. In *Proc. International Conference on Field Programmable Logic and Applications*, Sept. 2008, pp.23-28.

[7] Trimberger S, Moore J, Lu W. Authenticated encryption for FPGA bitstreams. In *Proc. the 19th ACM/SIGDA Symp. Field-Programmable Gate Arrays* (*FPGA*), Feb.27-Mar.1, 2011, pp.83-86.

[8] Drimer S. Security for volatile FPGAs [Ph.D. Thesis], Computer Laboratory, University of Cambridge, Nov uCAM-CL-TR-763, 2009.

[9] Herder C, Yu M, Koushanfar F, Devadas S. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 2014, 102(8): 1126-1141.

[10] Gora M, Maiti A, Schaumont P. A flexible design flow for software IP binding in FPGA. *IEEE Trans. Ind. Informatics*, 2010, 6(4): 719-728.

[11] Koushanfar F. Integrated circuits metering for piracy protection and digital rights management. In *Proc. the 21st Great Lakes Symposium on VLSI*, May 2011, pp.449-454.

[12] Roy J, Koushanfar F, Markov I. EPIC: Ending piracy of integrated circuits. In *Proc. Design, Automation and Test in Europe*, March 2008, pp.1069-1074.

[13] Note J, Rannaud E. From the bitstream to the netlist. In *Proc. the 16th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2008, p.264.

[14] Xia Z, Wang X, Sun X, Wang Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 2016, 27(2): 340-352.

[15] Fu Z, Wu X, Guan C, Sun X, Ren K. Towards efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security*, 2016, 11(12): 2706-2716.

[16] Xia Z, Wang X, Zhang L, Qin Z, Sun X, Ren K. A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Transactions on Information Forensics and Security*, 2016, 11(11): 2594-2608.

[17] Fu Z, Ren K, Shu J, Sun X, Huang F. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*, 2016, 27(9): 2546-2559.

[18] Fu Z, Sun X, Liu Q, Zhou L, Shu J. Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Transactions on Communications*, 2015, E98-B(1): 190-200.

[19] Guo P, Wang J, Li B, Lee S. A variable threshold-value authentication architecture for wireless mesh networks. *Journal of Internet Technology*, 2014, 15(6): 929-936.

[20] Ma T, Zhou J, Tang M, Tian Y, Al-Dhelaan A, Al-Rodhaan M, Lee S. Social network and tag sources based augmenting collaborative recommender system. *IEICE Transactions on Information and Systems*, 2015, E98-D(4): 902-910.

[21] Ren Y, Shen J, Wang J, Han J, Lee S. Mutual verifiable provable data auditing in public cloud storage. *Journal of Internet Technology*, 2015, 16(2): 317-323.

[22] Li J, Li X, Yang B, Sun X. Segmentation-based image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security*, 2015, 10(3): 507-518.

[23] Xia Z, Wang X, Sun X, Liu Q, Xiong N. Steganalysis of LSB matching using differences between nonadjacent pixels. *Multimedia Tools and Applications*, 2016, 75(4): 1947-1962.

[24] Xia Z, Wang X, Sun X, Wang B. Steganalysis of least significant bit matching using multi-order differences. *Security and Communication Networks*, 2014, 7(8): 1283-1291.

[25] Yuan C, Sun X, Lv R. Fingerprint liveness detection based on multi-scale LPQ and PCA. *China Communications*, 2016, 13(7): 60-65.

[26] Zhou Z, Wang Y, Wu Q, Yang C, Sun X. Effective and efficient global context verification for image copy detection. *IEEE Transactions on Information Forensics and Security*, 2016, 12(1): 48-63.

[27] Zhang Y, Sun X, Wang B. Efficient algorithm for k-barrier coverage based on integer linear programming. *China Communications*, 2016, 13(7): 16-23.

[28] Xie S, Wang Y. Construction of tree network with limited delivery latency in homogeneous wireless sensor networks. *Wireless Personal Communications*, 2014, 78(1): 231-246.

[29] Shen J, Tan H, Wang J, Wang J, Lee S. A novel routing protocol providing good transmission reliability in underwater sensor networks. *Journal of Internet Technology*, 2015, 16(1): 171-178.

[30] Zhang J, Qu G, Lv Y, Zhou Q. A survey on silicon PUFs and recent advances in ring oscillator PUFs. *J. Comput. Sci. Technol.*, 2014, 29(4): 664-678.

[31] Atallah M, Bryant E, Korb J, Rice J. Binding software to specific native hardware in a VM environment. In *Proc. the 1st ACM Workshop on Virtual Machine Security*, Oct. 2008, pp.45-48.

[32] Suh G, Devadas S. Physical unclonable functions for device authentication and secret key generation. In *Proc. the 44th ACM/IEEE Design Automation Conference*, June 2007, pp.9-14.

[33] Holcomb D, Burleson W, Fu K. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Computers*, 2009, 58(9): 1198-1210.

[34] Lim D, Lee J, Gassport B *et al.* Extracting secret keys from integrated circuits. *IEEE Trans. VLSI Syst.*, 2005, 13(10): 1200-1205.

[35] Lach J, Mangione-Smith W, Potkonjak M. Fingerprinting techniques for field-programmable gate array intellectual property protection. *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2001, 20(10): 1253-1261.

[36] Merli D, Schuster D, Stumpf F, Sigl G. Side-channel analysis of PUFs and fuzzy extractors. In *Proc. the 4th International Conference on Trust and Trustworthy Computing*, June 2011, pp.33-47.

[37] Zhang J, Lin Y, Qu G. Reconfigurable binding against FPGA replay attacks. *ACM Trans. Des. Autom. Electron. Syst.*, 2015, 20(2): 33:1-33:20.

[38] Gao M, Lai K, Qu G. A highly flexible ring oscillator PUF. In *Proc. the 51th ACM/IEEE Design Automation Conference (DAC)*, June 2014, pp.89:1-89:6.

[39] Zhang J, Wu Q, Ding Y *et al.* Techniques for design and implementation of an FPGA-specific physical unclonable function. *Journal of Computer Science and Technology*, 2016, 31(1): 124-136.

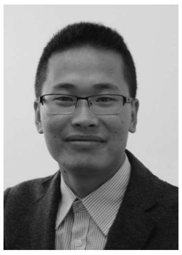
[40] Majzoobi M, Koushanfar F, Potkonjak M. Techniques for design and implementation of secure reconfigurable PUFs. *ACM Trans. Reconfigurable Technology and Systems*, 2009, 2(1): 5:1-5:33.

[41] Yin C, Qu G, Zhou Q. Design and implementation of a group-based RO PUF. In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, March 2013, pp.416-421.

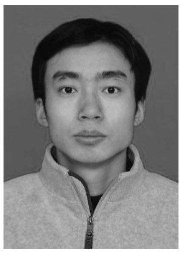
[42] Guajardo J, Kumar S, Schrijen G, Tuyls P. FPGA intrinsic PUFs and their use for IP protection. In *Proc. the 9th Int. Conf. Cryptographic Hardware and Embedded Systems*, Sept. 2007, pp.63-80.

[43] Anderson J. A PUF design for secure FPGA-based embedded systems. In *Proc. the 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2010, pp.1-6.

[44] Ruhrmair U, Solter J, Sehnke F *et al.* PUF modeling attacks on simulated and silicon data. *IEEE Trans. Information Forensics and Security*, 2013, 8(11): 1876-1891.



Ji-Liang Zhang received his Ph.D. degree in computer science and technology from Hunan University, Changsha, in 2015. In 2013∼2014, he worked as a research scholar at the Maryland Embedded Systems and Hardware Security Laboratory, University of Maryland, College Park. He is currently an associate professor in the Department of Information Security, Software College, Northeastern University, Shenyang. He authored over 30 papers in refereed international conferences and journals such as IEEE-TIFS, ACM-TODAES, IEEE-TVLSI, ACM/IEEE Design Automation Conference and so on. His research interests include hardware/hardware-assisted security, field programmable gate array, embedded system and emerging technologies.



Wei-Zheng Wang received his Ph.D. degree in technology of computer application from Hunan University, Changsha, in 2011. He is currently a lecturer at Department of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha. His research interests include low-power testing, design for testability and hardware security.

Xing-Wei Wang received his Ph.D. degree in computer science from the Northeastern University, Shenyang, in 1998. He is currently a professor at the Software College and the College of Information Science and Engineering, Northeastern University, Shenyang. His research interests include cloud computing and future Internet, etc.

Zhi-Hua Xia received his Ph.D. degree in computer science and technology from Hunan University, Changsha, in 2011. He is currently a lecturer in School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing. His research interests include privacy-preserving in cloud computing and digital forensic.