A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing

Jiliang Zhang, Yaping Lin, Yongqiang Lyu, and Gang Qu, *Senior Member, IEEE*

***Abstract*— With its reprogrammability, low design cost, and increasing capacity, field-programmable gate array (FPGA) has become a popular design platform and a target for intellectual property (IP) infringement. Currently available IP protection solutions are usually limited to protect single FPGA configurations and require permanent secret key storage in the FPGA. In addition, they cannot provide a commercially popular pay-per-device licensing solution. In this paper, we propose a novel IP protection mechanism to restrict IP's execution only on specific FPGA devices in order to efficiently protect IPs from being cloned, copied, or used with unauthorized integration. This mechanism can also enforce the pay-per-device licensing, which enables the system developers to purchase IPs from the core vendors at the low price based on usage instead of paying the expensive unlimited IP license fees. In our proposed binding-based mechanism, FPGA vendors embed into each enrolled FPGA device with a physical unclonable function (PUF) customized for FPGAs; IP vendors embed augmented finite-state machines (FSM) into the original IPs such that the FSM can be activated by the PUF responses from the FPGA device. We propose protocols to lock and unlock FPGA IPs, demonstrate how PUF can be embedded onto FPGA devices, and analyze the security vulnerabilities of our PUF-FSM binding method. We implement a 128-bit delay-based PUF on 28-nm FPGAs with only 258 RAM-lookup tables and 256 flipflops. The PUF responses are unique and reliable against environment changes. We also synthesize a variety of FSM benchmark circuits. On large benchmarks, the average timing overhead is 0.64% and power overhead in 0.01%.**

***Index Terms*— Binding, field-programmable gate array (FPGA), finite state machine (FSM), hardware metering, intellectual property (IP) protection, physical unclonable functions (PUFs).**

I. Introduction

A. Motivations

FIELD-PROGRAMMABLE gate arrays (FPGAs) are the semiconductor devices that can be reprogrammed by the end-users to implement any digital system. Comparing to the implementation with Application-specific Integrated Circuits (ASICs), FPGA design has the advantages of shorter time-to-market, lower non-recurring engineering costs and higher flexibility. These have made FPGA a popular design platform for many applications such as automotive electronics, consumer electronics and aerospace equipments. In this FPGA-based design platform, third-party intellectual properties (IPs) are widely used due to both the technical merits (e.g., the IPs proven functionality, compatibility, and performance) and non-technical concerns (e.g., time-to-market, cost, and patent enforcement). However, there are severe piracy attacks to the FPGA IPs and the current licensing schemes are also not flexible enough to precisely control the authorized usage.

Firstly, from the perspective of the attack, piracy attacks, such as cloning, copy, misuse and unauthorized integration, are considered to be the most common security vulnerability of volatile FPGAs [1]. Un-configured FPGA devices are off-the-shelf products, and the configuration bitstreams can be obtained by eavesdropping or directly from the volatile SRAM FPGAs [1], which not only reduces the profits and market share, but also causes the damage to the brand reputation and even leads to severe early product failures and safety hazards [1], [2]. Furthermore, this is not limited to high-value single FPGA designs; the third-party FPGA intellectual property (IP) cores are also vulnerable to those attacks.

Secondly, from the perspective of licensing, it is often vital to ensure that the configuration bit-streams can only be used on the licensed FPGA devices. In such a case, IP core vendors would prefer to sell their IP products through pay-per-device licensing rather than through up-front license fees that allows users to configure any FPGA device. In order to adapt the IP core business model for the low/medium-volume FPGA applications [3], effective pay-per-device licensing techniques are in urgent need.

Mainstream FPGA vendors have been paying more and more efforts in protecting their IPs from piracy attacks and improving licensing schemes to activate and protect the IP-based commercial flow. However, the state-of-art techniques still have some drawbacks. In this paper, we consider hardware IPs (HWIPs) as the soft-core (synthesized from HDL) hardware modules stored in the FPGA configuration bitstreams [11]. Our goal is to develop techniques to solve the piracy and licensing challenges. We propose to solve these problems by a binding mechanism that seeks to restrict the execution of the protected IPs to the authorized FPGA devices only.

Manuscript received January 3, 2014; revised May 30, 2014 and November 29, 2014; accepted January 28, 2015. Date of publication February 5, 2015; date of current version April 13, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61173038 and Grant 61228204 and in part by a scholarship from China Scholarship Council under Grant 201306130042. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Farinaz Koushanfar.

J. Zhang and Y. Lin are with the College of Information Science and Engineering, Hunan University, Changsha 410082, China (e-mail: hnu.zjl@gmail.com; yplin@hnu.edu.cn).

Y. Lyu is with the Research Institute of Information Technology, Tsinghua University, Beijing 100084, China (e-mail: luyq@tsinghua.edu.cn).

G. Qu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: gangqu@umd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2400413

B. Limitations of Prior Art

FPGA HWIP protection techniques have been well-studied in academic [2]–[5] and widely used in industry [6]–[9]. However, all the existing HWIP protection techniques are based on encryption and have the following main drawbacks:

1) The commercially available encryption-based techniques can only protect the single large FPGA configurations.
2) The commercially encryption-based techniques cannot provide a solution to the commercially popular pay-per-device licensing requirement for both single large configurations and individual IP cores.
3) The current encryption-based FPGA IP protection methods introduce security vulnerabilities (e.g., physical attacks and side channel attacks) for permanent key storage and management.

C. Our Contributions

In this paper, we propose a binding scheme that binds the HWIPs to specific FPGA devices via the interaction between physical unclonable functions (PUF) built on the FPGA devices and the FSMs in the HWIPs in order to address the limitations of existing FPGA HWIP protection techniques. We first report this concept in [37]. In this article, 1) we provide a concrete construction and implementation of a delay-based PUF on 28nm FPGAs as a reference design of our binding scheme; 2) we implement and verify the proposed binding scheme by synthesizing MCNC'91 circuits and large FSM benchmarks from GenFSM on FPGAs; 3) we elaborate the details of the proposed binding scheme with illustrative example and in-depth discussion on design flow, system integration, and security vulnerabilities. To the best of our knowledge, this is the first non-encryption based FPGA HWIP binding method. Comparing to the traditional encryption-based HWIP protection methods, our approach has the following advantages:

1) It can be used to protect both single FPGA configurations and third-party FPGA IP cores. Currently available encryption-based commercial methods can only protect the former, but not the latter.
2) It supports the pay-per-device licensing mechanism. The FPGA configuration bitstream can only be used to configure specific FPGA devices, giving IP vendors control of their IPs and allowing product developers to pay licensing fee only for the FPGA devices they are using.
3) It does not need permanent storage for secret keys in the FPGA. In our binding scheme, the secret PUF response can be ephemeral and immediately cleared after use. Therefore, it eliminates the security vulnerabilities of the permanent key management and exchange.
4) It has low hardware overhead. We implement the proposed method on Virtex-5 FPGA devices and find that the 128-bit delay-based PUF needs about 256 slices [41] and the modified FSM only introduces 0.64% timing overhead and 0.01% power overhead on average for ten large FSM designs. As a comparison, previous FPGA IP protection schemes consume 6776 LUTs for a SHA-1 core and an ECDH core [4], [5].

D. Outline of the Paper

The rest of this paper is organized as follows. Related work is surveyed in Section II. The necessary background information on PUF, FSM, and parties involved in HWIP binding is presented in Section III. The proposed binding method and its working mechanism are elaborated in Section IV. An IP locking mechanism and a reference implementation of PUF for the proposed binding method are then given in Section V and Section VI, respectively. Potential security threats and countermeasures are analyzed in Section VII. The detailed experimental results and analysis are reported in Section VIII. Finally, we conclude in Section IX.

II. Related Work

A. FPGA HWIP Protection Techniques

Many intellectual property protection techniques for FPGAs have been proposed in academic and industry.

In commercial tools, bit-stream encryption [6]–[9] is the most popular intellectual property protection method against direct cloning of single large FPGA configurations for high-end FPGA devices. Some recent FPGAs employ the advanced encryption standard (AES) core or triple data encryption standard (3DES) core to support the encryption of the FPGA configuration bitstreams; some FPGAs employ keyed-hash message authentication code (HMAC) core to enable bit-stream authentication [8]. They all need the on-chip cryptographic decryption module and the permanent secure key storage. Unfortunately, these solutions come with some practical limitations: they are not appropriate for resource-limited environments, and more importantly, it is well-known that such permanent key storage scheme allows attackers to attack at any time.

In the academic domain, Gneysu *et al.* [4] proposed a protection scheme for the FPGA bitstreams, which uses the secondary secure key register and the authenticated bitstream encryption and requires minor modification to the current FPGA technology. They employed a public-key-based protocol between the IP providers and the FPGA-based system developers, and a trusted third party (TTP) is used to handle key exchange and installation in the symmetric-key-decryption engines. This solution is only suitable for the protection of single large FPGA configurations, and the protection of individual HWIP cores remains as a challenging problem. Drimer *et al.* [5] presented an encryption-based method to protect multiple IPs, and Kepa *et al.* [13] proposed a secure reconfigurable controller based method to support license enforcement within the partial reconfiguration flow. More recently, Maes *et al.* [2] introduced a valuable "pay-per-use" licensing scheme to protect multiple FPGA IPs through the self-reconfiguring capabilities of modern FPGAs and a TTP for metering the service.

As we can see from the above, all commercial and academic FPGA configuration bitstream protection methods are encryption-based; they have three shortcomings: 1) the commercial methods are limited to the protection of single large FPGA configurations; 2) they cannot support the pay-per-device licensing; 3) the previous encryption-based

HWIP protection methods require permanent key storage and on-chip cryptographic decryption modules to decrypt the bitstream, which introduces some security vulnerabilities and high overhead. Our approach overcomes these limitations.

B. Metering ASIC Intellectual Properties

A number of watermarking methods for ASIC/FPGA intellectual property protection have been proposed [32]–[35]. However, watermarking techniques are passive and only used to identify the intellectual property. In 2001, Koushanfar and Qu [38] proposed the first hardware metering method that can enable the design house to gain the post-fabrication control by passive or active control of the number of produced ICs. Alkabani *et al.* [24] proposed an anti-overbuilding mechanism which exploits the functional description of the design and the unique and unclonable IC identifiers. The locks can be embedded via modifying the hardware computational model such as an FSM. They also presented another FSM manipulation method [25] which introduces only a few new states. These solutions are only suitable for protecting single ASIC chips. Later on, they further extended their scheme to actively control multiple IP cores [26] for ASIC chips. Recently, Koushanfar [27] improved again the locking structure in [24] by a multi-point function. Meanwhile, Roy *et al.* [20] presented another kind of cryptography-based metering methods, but their solution has a very high overhead. These metering mechanisms are designed for anti-overbuilding ASIC devices, they are not appropriate for pay-per-device licensing of FPGA designs.

In this paper, our proposed FPGA HWIP binding technique not only addresses the main drawbacks of the traditional FPGA HWIP protection methods, it can also support a *pay-per-device* licensing scheme. This provides technical support for the product developers (system developers) to pay IP licensing fees only for the FPGA devices they are using. It also enables the IP vendors to freely distribute their IPs because they can ensure that the distributed IPs run only on specific FPGAs rather than all the FPGAs. This binding scheme brings a remarkable advantage for the IP-based business model: the IP owners can take the full control over the use of their IP cores and protect them from unlicensed use; the FPGA-based product developers who could not afford the expensive unlimited IP license are now also able to obtain a number of single instances of the required IP cores at a much lower cost.

III. PRELIMINARIES

In this section, we will introduce the general terms and concepts used throughout the paper. More specific definitions would be described as necessary.

A. Physical Unclonable Function (PUF)

PUF provides a unique chip-dependent mapping from a set of digital inputs (challenges) to a set of digital outputs (responses) based on the unclonable properties of the underlying physical device. Although it is difficult to come up with a uniform definition for all types of PUFs, they should all satisfy the following properties [39]:

- *Persistent and unpredictable*. The response (R_i) to a challenge (C_i) is random and unpredictable, but should remain the same for the same challenge over multiple observations.
- *Unclonable*. It is impossible to obtain R_i from C_i without the physical presence of the PUF. In other words, given a PUF, it is infeasible for an adversary to build another PUF that provides the same responses to every possible challenge. This is assumed to be true due to the uncontrollable technology variations.
- *Tamper evident*. Invasive attacks to PUFs will destroy the PUFs and thus can be detected easily.

Because of those properties, PUF has become an efficient mechanism to address security and trust problems in many applications, such as binding software IPs to specific FPGAs [11], hardware/software authentication [16], FPGA IP protection [18], [43], anti-overbuilding [24]–[27] and resisting FPGA replay attacks [36].

B. Finite State Machine (FSM)

FSM is a popular model for sequential systems. In this paper, we employ FSMs to bind HWIPs to the FPGAs with PUFs to restrict the HWIP's usage so that it can only work on the enrolled FPGA devices. Similar to the FSM-based works such as [15] and [24]–[27], the method proposed in this paper is not applicable to some high-speed designs that do not have FSMs. These high-speed designs are normally small dedicated modules such as digital filters, channel equalizers, address decoders and arithmetic logic units. Fortunately, for the HWIPs in industrial designs that we target to protect, the sequential components or functions, and therefore FSMs, are ubiquitous [15].

C. Parties Involved in HWIP Binding

In order to facilitate our study, we consider the following parties involved in the binding mechanism and their respective roles:

- FPGA vendor (FV): FV designs and manufactures un-configured FPGA devices and can securely deploy PUF in the fabric of these devices.
- System developer (SD): SD integrates the third-party IPs along with their own designs to create a commercial product on an FPGA chip. The product will be synthesized into a configuration bitstream file for the FPGA chip to download using the computer aided design (CAD) tools provided by the FV.
- IP core vendor (CV): CV creates innovative logic circuits (HWIP cores) and sells them to SDs for profits. CV needs an effective technique to keep the full control over the use of the HWIP cores.
- End user (EU): EU purchases the FPGA products developed by the SD. The SD expects that EUs cannot ‘clone’ the products by copying the FPGA configuration bitstream file and run on unauthorized FPGA devices.

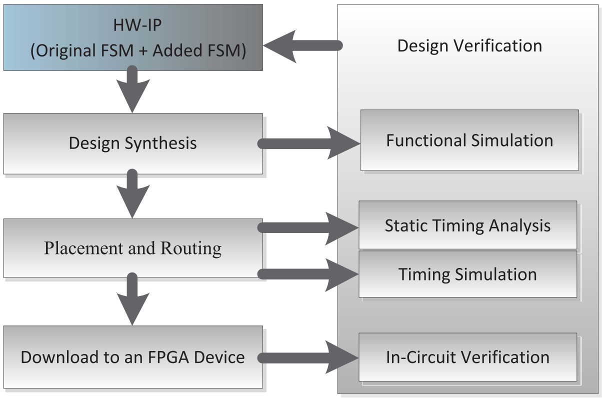


Fig. 1. Design flow of modifying hardware IP.

Our goal is to design a new binding mechanism so the SDs and CVs can protect their FPGA designs or IPs from piracy without introducing much inconvenience and large performance degradation to the EUs and FVs.

IV. The Proposed Binding Scheme

Traditionally, the HWIPs are written without any concern of binding to any specific FPGA. The configuration bitstream can be used to configure any FPGA device of the same type. Given a HWIP, our goal is to modify the original FSM of the HWIP to produce an augmented FSM which is functionally equivalent to the former. The modified FSM reacts with the intrinsic PUF located in the specific FPGA hardware, and it can perform exactly the same as that of the HWIP as long as the challenges issued by the HWIP obtain correct responses through the PUF. This means that only the FPGA chips authorized by the CVs can guarantee the correct functionalities. Meanwhile, as long as there are PUFs embedded in the FPGA chips from the FV, and the CV modifies their IP designs to support PUF, no more changes are needed at any party when a new HWIP is developed and needs to be deployed to a new FPGA device. The details of the binding scheme are depicted in figures 1, 2, and 3 and described as follows.

A. Design Flow

The design flow of modifying the HWIP together with a standard FPGA design methodology is shown in Fig. 1. First, the CV uses the high-level design description to setup the behavioral model of the FSM. Next the original FSM is modified so that the added FSM structure (such as additional states and transitions) and the original FSM form a new augmented FSM. The standard phases of the FPGA design methodology (e.g., design synthesis, placement and routing) can then be carried out. Finally, the HWIP configuration bitstream file can be downloaded into an FPGA device to run. Although there is an inevitable verification/testing overhead due to the added features in the augmented FSM, the entire traditional design methodology is maintained so the introduced design overhead can be controlled.

TABLE I

Symbols and Acronyms Used in the Protocol

Symbols and Acronyms	Description
F_i	an FPGA device
F^i_{PUF}	a F_i with a PUF_i deployed inside
$HWIP_j$	a hardware IP core
$b(HWIP_j)$	the (partial) bit-stream of the $HWIP_j$
P_j	a product developed by the SD
$b(P_j)$	the bit-stream of P_j
$ID(HWIP_j)$	the uniquely public identifier for $HWIP_j$
$ID(P_j)$	the uniquely public identifier for P_j
$ID(F^i_{PUF})$	the uniquely public identifier for F^i_{PUF}

B. Description of the Protocol

For reader's convenience, we list the symbols and acronyms used in the protocol, as shown in Table I. The proposed PUF-FSM binding protocol is described as follows.

1) FPGA Device Enrollment: The device enrollment protocol is shown in Fig. 2(a). To enable the proposed scheme, the FPGA vendor (FV) initially tests the PUF for every piece of FPGA chip to obtain their random challenge-response pairs (CRPs) before selling them. The PUF challenges are stored in the non-volatile on-chip memory, which is automatically configured on F^i_{PUF} immediately when the device is powered on. Note that the PUF challenges can be public and do not need to be encrypted or hidden because of the uniqueness and unpredictability of the PUF responses. In addition, FV can also generate the $ID(F^i_{PUF})$ which is a public unique serial number burned in at manufacturing time (e.g., Xilinx Device DNA [19]). If the core vendor (CV) or system developer (SD) wants to buy the FPGA embedded with the PUF, F^i_{PUF}, to start the HWIP/system development, the FV will respond with the $ID(F^i_{PUF})$ from database and then sell the FPGA device F^i_{PUF} to the CV/SD.

2) Hardware IP Core Enrollment and Distribution: As Fig. 2(b) shows, before the system developer (SD) develops its product, the core vendor (CV) creates the IP with $ID(HWIP_j)$. The CV then synthesizes the $HWIP_j$ with the PUF-binding FSM into the bit-stream to generate the new version $b\{HWIP_j\}_{locked}$. This process can be expressed as $b\{HWIP_j\}_{locked}$ = Lock $b\{HWIP_j\}$. The CV stores $ID(HWIP_j)$ and $b\{HWIP_j\}_{locked}$ in its database, and releases $ID(HWIP_j)$ for sale. When a SD needs the $HWIP_j$ to develop FPGA-based products, it asks for buying it via sending the $ID(HWIP_j)$ to the CV. The CV then looks up the database for $ID(HWIP_j)$ and sends the corresponding $b\{HWIP_j\}_{locked}$, the locked HWIP bit-stream, to the SD.

3) Hardware IP Core Licensing: As Fig. 2(c) shows, when the system developer (SD) requires to unlock the purchased $b\{HWIP_j\}_{locked}$ in their FPGA-based products, it sends $ID(F^i_{PUF})$ and $ID(HWIP_j)$ to the core vendor (CV). The CV will send $ID(F^i_{PUF})$ to the FPGA vendor (FV) to obtain the corresponding CRPs and then calculate licenses based on the CRPs and the modified FSM. The computed licenses can be public. Finally, the licenses are sent to the SD to unlock b$\{HWIP_j\}$. This process can be expressed as $b\{HWIP_j\}_{unlocked} = b\{HWIP_j\}_{locked}(Licenses)$. Note that the CRPs should be securely transferred from the FV to the CV or SD.

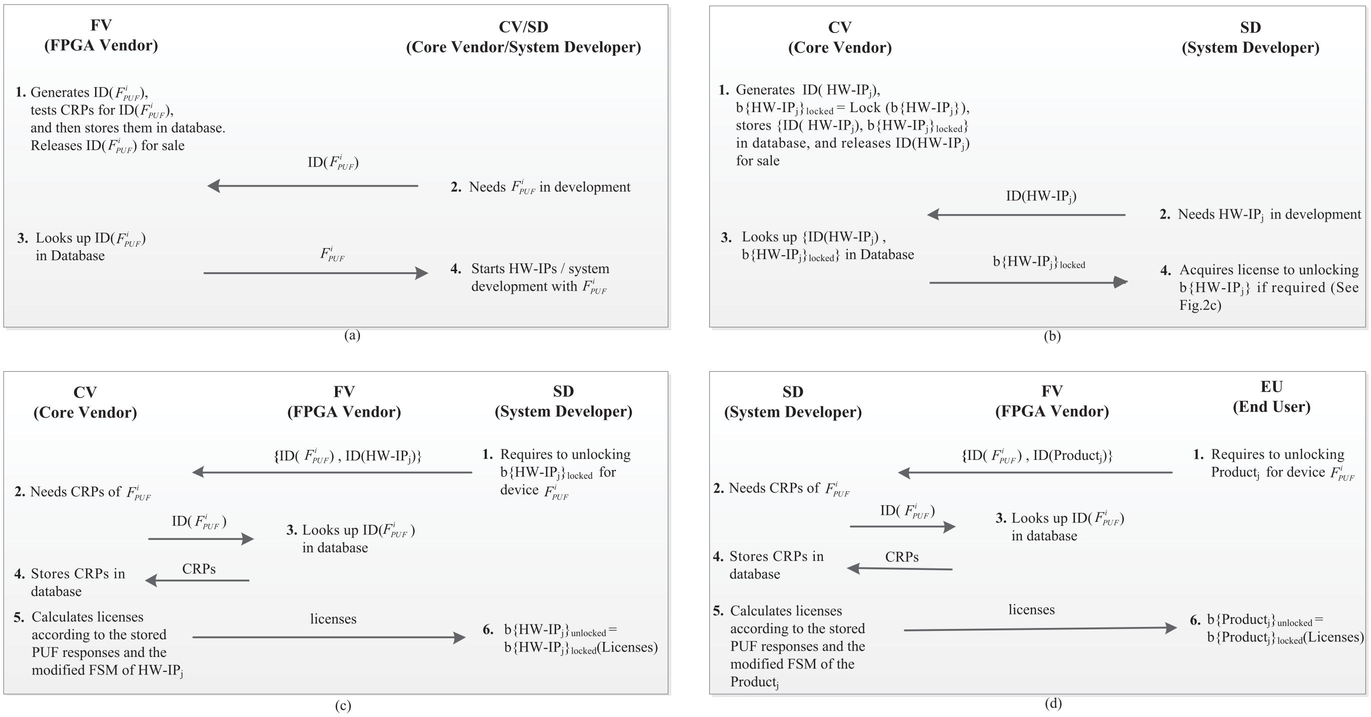

FV
(FPGA Vendor)
CV/SD
(Core Vendor/System Developer)
1. Generates ID(F^i_{PUF}), tests CRPs for ID(F^i_{PUF}), and then stores them in database. Releases ID(F^i_{PUF}) for sale
ID(F^i_{PUF})
2. Needs F^i_{PUF} in development
3. Looks up ID(F^i_{PUF}) in Database
F^i_{PUF}
4. Starts HW-IPs / system development with F^i_{PUF}
(a)
CV
(Core Vendor)
SD
(System Developer)
1. Generates ID(HW-IP$_j$), b{HW-IP$_j$}$_{locked}$ = Lock (b{HW-IP$_j$}), stores {ID(HW-IP$_j$), b{HW-IP$_j$}$_{locked}$} in database, and releases ID(HW-IP$_j$) for sale
ID(HW-IP$_j$)
2. Needs HW-IP$_j$ in development
3. Looks up {ID(HW-IP$_j$), b{HW-IP$_j$}$_{locked}$} in Database
b{HW-IP$_j$}$_{locked}$
4. Acquires license to unlocking b{HW-IP$_j$} if required (See Fig.2c)
(b)
CV
(Core Vendor)
FV
(FPGA Vendor)
SD
(System Developer)
{ID(F^i_{PUF}), ID(HW-IP$_j$)}
1. Requires to unlocking b{HW-IP$_j$}$_{locked}$ for device F^i_{PUF}
2. Needs CRPs of F^i_{PUF}
ID(F^i_{PUF})
3. Looks up ID(F^i_{PUF}) in database
4. Stores CRPs in database
CRPs
5. Calculates licenses according to the stored PUF responses and the modified FSM of HW-IP$_j$
licenses
6. b{HW-IP$_j$}$_{unlocked}$= b{HW-IP$_j$}$_{locked}$(Licenses)
(c)
SD
(System Developer)
FV
(FPGA Vendor)
EU
(End User)
{ID(F^i_{PUF}), ID(Product$_j$)}
1. Requires to unlocking Product$_j$ for device F^i_{PUF}
2. Needs CRPs of F^i_{PUF}
ID(F^i_{PUF})
3. Looks up ID(F^i_{PUF}) in database
4. Stores CRPs in database
CRPs
5. Calculates licenses according to the stored PUF responses and the modified FSM of the Product$_j$
licenses
6. b{Product$_j$}$_{unlocked}$= b{Product$_j$}$_{locked}$(Licenses)
(d)

Fig. 2. Hardware IP core binding protocol. (a) FV generates $ID(F^i_{PUF})$ and CRPs and then sells devices to SD and CV. (b) CV generates the locked $HWIP_j$ and distributes it to SD. (c) CV licenses $HWIP_j$ to SD. (d) SD licenses $Product_j$ to EU.

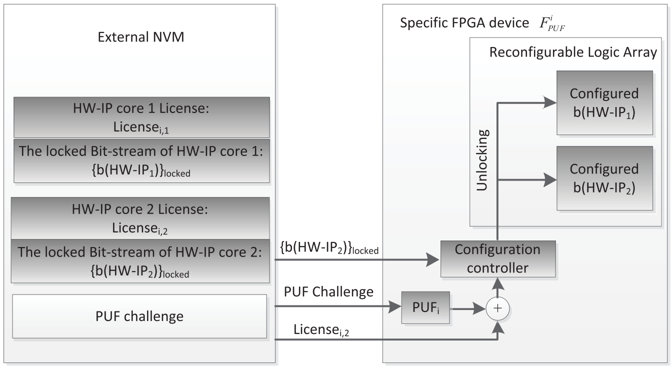

External NVM
HW-IP core 1 License: License$_{i,1}$
The locked Bit-stream of HW-IP core 1: {b(HW-IP$_1$)}$_{locked}$
HW-IP core 2 License: License$_{i,2}$
The locked Bit-stream of HW-IP core 2: {b(HW-IP$_2$)}$_{locked}$
PUF challenge
{b(HW-IP$_2$)}$_{locked}$
PUF Challenge
License$_{i,2}$
Specific FPGA device F^i_{PUF}
Reconfigurable Logic Array
Configured b(HW-IP$_1$)
Configured b(HW-IP$_2$)
Unlocking
Configuration controller
PUF$_i$
+

Fig. 3. Configuration process of an FPGA-based product containing multiple locked hardware IP cores.

4) Product Licensing: As Fig. 2(d) shows, if an end user (EU) would like to buy the products developed by the system developer (SD) to run on a specific FPGA device F^i_{PUF}, it should send $ID(F^i_{PUF})$ and ID($Product_j$) to SD. The SD will send $ID(F^i_{PUF})$ to FV to obtain the corresponding CRPs and then calculate licenses based on the FPGA-PUF responses and the modified FSM. Note that the licenses can be public. Finally, the licenses are sent to EUs to unlock b{$HWIP_j$}. This process can be denoted as $b\{Product_j\}_{unlocked} = b\{Product_j\}_{locked}(Licenses)$.

C. System Integration

The proposed binding scheme can support multiple HWIP cores to be integrated on a single FPGA design. To develop an FPGA-based product, the system developer (SD) obtains an FPGA device with the hard core PUF inside from the FPGA vendor (FV) [following Fig. 2(a)], the required third-party HWIP cores from the core vendors (CVs) [following Fig. 2(b)] and the required licenses for these cores from the CVs [following Fig. 2(c)]. For example, when there are two different HWIP cores from two different CVs; the SD can integrate them into the same FPGA device by putting the PUF challenges from the FV and the authorized licenses from the CVs in a nonvolatile memory (NVM) next to the FPGA device, then our IP protection scheme will work as shown in Fig. 3. When the system is powered on, the activation process checks the PUF-based licenses of the IPs and loads the unlocked IP cores into the reconfigurable FPGA fabric. If a purchased HWIP is copied to an unauthorized FPGA device, even the same license cannot unlock the HWIP because the unauthorized FPGA could not generate the same PUF responses as the authorized one.

V. HWIP Locking Mechanism

A. Locking the Hardware IP

In this section, we describe a prototyping design of the *lock* mechanism proposed in the binding scheme. The *lock* is achieved by exploiting PUF's unique properties (unclonable, persistent and unpredictable). As Fig. 4 shows, we use the PUF response to control the transitions of the FSM in the HWIP. The error corrected PUF response is used to uniquely determine the transitions of the state transition graph (STG) of the HWIP (the IP behavior); without the correct PUF response, the STG would not perform correctly. Therefore, the circuit is kept *locked* until the correct license (formed by

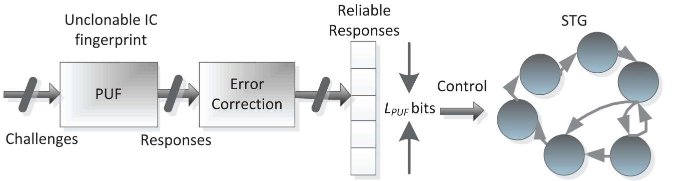


Fig. 4. PUF response is used to uniquely control the transitions of the STG.

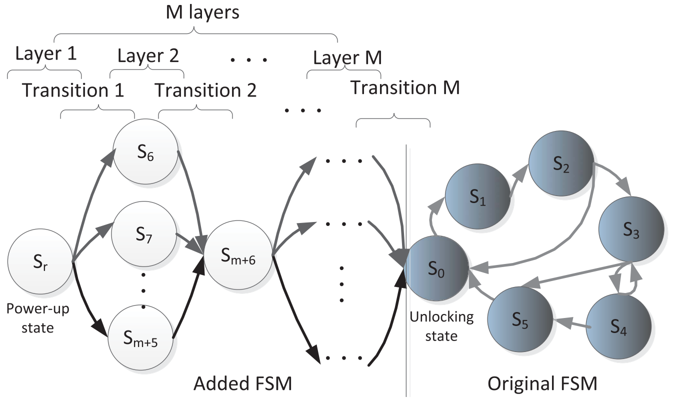


Fig. 5. The binding FSM structure. The original states are shown in dark and the added states are shown in white on the STG of the added FSM.

the correct PUF response) unlocks it. It should be noticed that the computed licenses can also be public and different PUF responses can be used to calculate different licenses. Additionally, the FPGA vendor often computes the error correcting code (ECC) to adjust for any bit-flip to the PUF output (response) because the PUF output is hard to maintain absolutely stable due to the noise or other sources of physical uncertainty.

As an example, considering the original STG with 6 states, $S_0 \sim S_5$, in Fig. 5, the transition from state S_0 to S_1 is excited by a specific input combination. S_0 is called the reset state of the original FSM and S_1 is the next state of this transition. Now we introduce the method to generate a new FSM with additional structure to bind with PUF. We add M (M is an even number) layers of states to form the added FSM. Any even-number layer consists of m states and any odd-number layer only has one state. We define a fixed power-up state S_r for the binding FSM. The first transition step starts from S_r with m transitional edges to each of the other m states. Then the second transition step goes from each of these m states to the next layer (odd layer). After the M-layer (M transition steps) transitions, the state transits to S_0 which is the unlocked state (the reset state).

Assuming the input of the original STG is k-bit long, we define a k-bit input sequence: $\{b_1 b_2 \cdots b_k\}_i^t(\{t|t \in N, 1 \leq t \leq 2k, k \in N\}; i \in N, 1 \leq i \leq M)$ where i denotes the i-th transitional step and t denotes the specific transition in the i-th transitional step. The k-bit input is the function of a L-bit PUF response which determines the transition path in the transitional steps. The odd-number transitional steps are determined by partial bits of the PUF response, and the even-number transitions of the FSM are determined by the license and the rest PUF response.

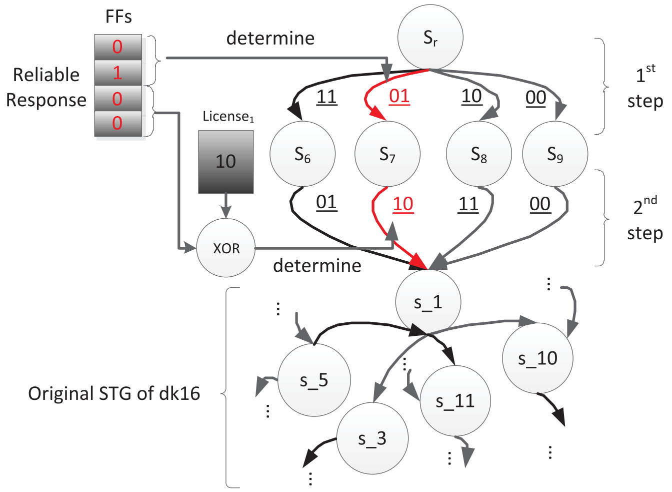


Fig. 6. An example of the lock mechanism and the generation of the licenses for sequential circuit dk16.

For an added FSM structure of M layers, every transitional step needs $\log_2(m)$-bit PUF output. Hence, the length of required PUF bits can be formulated as Eq. (1) shows.

$$L_{PUF} = M \times \log_2(m) \tag{1}$$

In addition, $\log_2(m)$-bit license should also be provided in the even-layer transitional step. The length of license bits can be computed by:

$$L_{license} = \frac{M}{2} \times \log_2(m) \tag{2}$$

Note that L_{PUF} and $L_{license}$ should be sufficiently long in practice in order to guarantee the security of this model.

To illustrate the key idea of our approach, we give an example of the implementation of the *lock* and the generation of the licenses for benchmark circuit *dk*16 shown in Fig. 6. Considering a two-layer added FSM structure composed by two transition steps, assume $k = 2$, hence each transitional step consists of 4 ($2^2 = 4$) edges forming the 4 transition paths. In the step 1, we use $\{b_1 b_2\}_i^t(t = 1, 2, 3, 4)$ which is designed to be 4 different values to distinguish the 4 edges. Then the value of $\{b_1 b_2\}_i^t$ will be decided by a 2-bit PUF output value once the design begins to run, it begins from S_r to one of the four connected states depending on the first 2-bit PUF outputs. As the first 2-bit PUF output value is "01", which equals to the designed $\{01\}_1^2$, thus the 1^{st} step will transition from S_r to S_7. Then in the 2^{nd} step, the design can only possibly transition from S_7 to S_{10} when the first two input bits equal to $\{10\}_2^2$. To possibly enable the transition, the second 2-bit PUF output "00" should be XOR'd with a 2-bit key that is able to generate the result of "10" (in this case the key should be "10"). The FSM can transit from state S_r to s_1 (the original reset state of *dk*16) with the calculated license and the PUF response.

B. Unlocking the Hardware IP

The PUF outputs L_{PUF} bits to determine the transitions of the binding FSM. Now, an attacker with no information about the transition table of the FSM cannot find the correct

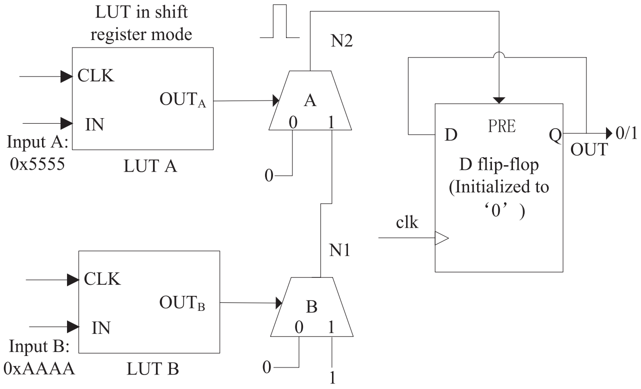


Fig. 7. The structure of the delay-based PUF design.

sequence of the primary input combinations to arrive at the reset state S_0. Hence, the CV is the only one who can compute the license to unlock the $b\{HWIP_j\}_{locked}$.

The unlocking process is stated as follows. The FV provides enrolled PUF-embedded FPGA F^i_{PUF}; each F^i_{PUF} provides a specific set of PUF challenges. If a $b\{HWIP_j\}_{locked}$ is illegally over-used, copied or cloned by a SD, it would be locked into the fixed power-up state, S_r, on the event of powering up the F^i_{PUF} because it does not have the correct PUF responses for the challenges. Hence, in order to unlock the design, the CV must use the received L_{PUF}-PUF responses from the FV who tests the PUF responses on the provided PUF challenges, and then calculates the correct $L_{license}$-bit license for SD. SD can use this license to unlock the $b\{HWIP_j\}_{locked}$ correctly.

VI. The Reference Implementation of PUF

Many kinds of PUF have been proposed in the past decade [42], such as optical PUF, SRAM PUF, arbiter PUF and ring oscillator PUF. Some of them have also been implemented on FPGAs [17], [18], [21], [22]. The proposed binding method can work with any PUF implemented on FPGA that satisfies the properties defined in section III.A. Which PUF to use is up to the FPGA vendor. In this study, we give a concrete implementation based on a delay-based PUF for the designers to refer to. This PUF is designed specifically for FPGAs. It does not need the hard macro with fix routing and is completely described in VHDL with the merits of easy-of-use and low silicon area overhead [41].

A. A Delay-Based PUF

In this paper, we designed and implemented a delay-based PUF on 28nm FPGAs, which takes advantage of the manufactured difference of the switching latencies of two carry-chain multiplexers on the FPGA to produce a positive pulse (glitch) at the output of downstream multiplexer. The glitch can be used to set the output of a D flip-flop to logic-1 from the default logic-0, which forms a one-bit PUF response. The detailed structure of the PUF design is illustrated in Fig. 7. The shift register contents are pre-initialized as follows:

- Input A: 0x5555 (0101010101010101)
- Input B: 0xAAAA (1010101010101010)

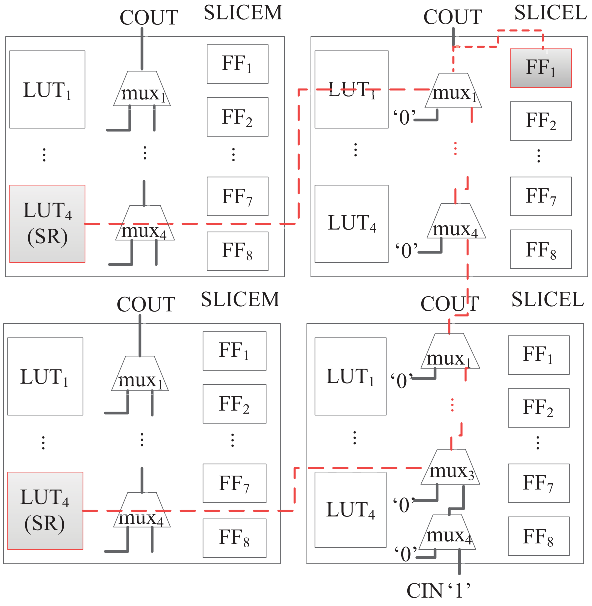


Fig. 8. The new prototype implementation of a primitive PUF on Xilinx Zynq-7000 FPGA.

When the look-up-table (LUT) *A* and its driving multiplexer *A* are faster than the LUT *B* and multiplexer *B*, the output *OUT* would be logic-1.

Note that the current delay-based PUF [41] for FPGAs cannot be directly implemented on the latest Xilinx FPGAs such as Virtex-7, Kintex-7, Artix-7 and Zynq-7000 since the structure of SLICE of the latest Xilinx FPGAs are different from that of the previous FPGA families such as Virtex-5. In the architecture of Virtex-5 FPGAs, once a LUT in SLICEM is configured into a shift register, the logic-0 data input of multiplexer can be connected to a logic-0 signal to meet the design requirement. However, in the SLICEM of Zynq-7000 FPGAs, two optional paths of logic-0 data input of the carry chain multiplexer have been used as output or input of a shift register, which cannot meet the requirement that logic-0 data input of multiplexer should be always connected to a logic-0 signal.

To solve this problem, we use four SLICEs to implement one bit PUF signature. In the layout of Xilinx Zynq-7000 XC7Z020 FPGA, there are two SLICEs in one CLB. A SLICE whose *X* coordinate is even number is SLICEM; then the other SLICE in the same CLB would be SLICEL. Two SLICEMs are configured into two shift registers respectively, while their corresponding SLICELs are configured into a carry chain multiplexer. As shown in Fig. 8, four slices are used to implement a new primitive PUF. The dotted line represents the direction of data flow.

B. Reliability-Enhancing Techniques

Silicon PUF is based on manufacture variation, which may be very sensitive to the operating environment such as voltage and temperature, particularly for delay based PUF [10], [22]. It is very hard for any known PUF to maintain an absolutely stable response. Methods such as error correcting [21], [40], pattern matching [44], [45], and temperature aware collaboration [46] have been proposed to correct bit flips in PUF responses to generate stable PUF output.

These state-of-the-arts have already been very successful in reducing and correcting the PUF bit errors. For example, by using the Index-Based Syndrome coding (IBS), error correction performed on output responses of ring oscillator (RO) PUFs implemented on Virtex-5 FPGAs has an error rate less than 10^{-6} when temperature goes from -55°C to 125°C under 1.0V operating voltage with a ± 10% variation [40]. Maes *et al*. demonstrated that an efficient and extremely low overhead BCH decoder specially for correcting bit flips in PUF responses utilizes merely 112 Slices on a Xilinx Spartan-6 FPGA [21]. Paral and Devadas proposed to use string pattern matching to generate reliable PUF responses, both the false positive and false negative rates can be less than 10^{-9} [44]. Yin and Qu built a temperature aware collaborative RO PUF where they measure the PUF output values at different temperatures and choose the correct one based on the real operating temperature from on-chip temperature sensors, which ideally guarantees no bit error [46].

Moreover, electro-migration, hot carrier injection (HCI), negative bias temperature instability (NBTI) and temperature-dependent dielectric breakdown (TDDB) cause the device aging, which would impact the stability of PUF signatures [47], [48]. For example, Ganta *et al*. [48] observe that around 4% of the RO-PUF bits are prone to instability due to aging in various operating conditions. Recently, the corresponding aging resistant techniques [49], [50] are also developed.

As we will report in Section VIII, our reference delay-based PUF does have bit errors when operating at different temperatures. However, such error will not cause any false positive or false negative, which means that we will be able to distinguish a PUF response with bit errors from a PUF response from a different device. To keep our discussion focused on the new PUF-FSM binding scheme for FPGA IP protection and pay-per-device licensing, we will not elaborate how to improve the reliability of the above reference delay-based PUF and the associated cost. When high reliable PUF responses are critical, one can always use one or more of the reliability-enhancing techniques. It will be a task for the vendors and IP developers to balance the tradeoff between PUF reliability and design overhead.

C. The Integration Architecture

The delay-based PUF in this paper is designed in HDL, and hence has the merits of easy-of-use and high flexibility. The PUF can be implanted into FPGA in the form of soft-core or hard-core. Soft-core PUF is implemented in FPGA fabric while hard-core PUF actually physically implemented as a structure in the silicon connected to the FPGA fabric. Both hard-cores and soft-cores have been widely adopted in FPGA industry. An example of FPGA with hard-cores is the ARM Cortex-A9 dual-core MCU used in the new Xilinx Zynq-7000 System on a Programmable Chip (SOPC). On the other hand, soft-cores are more commonly used in FPGAs such as MicroBlaze, Nios II and OpenRISC.

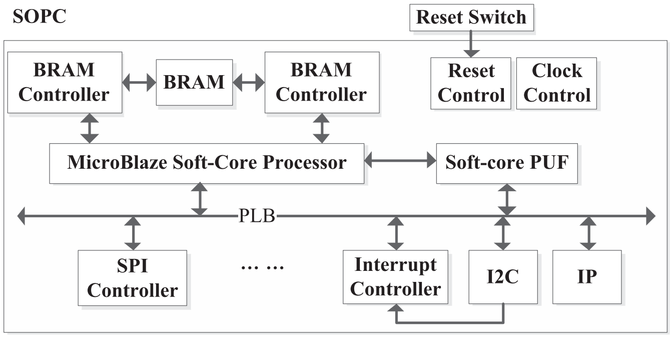


Fig. 9. An example of a soft-core PUF implanted in a SOPC.

Fig. 9 illustrates how a soft-core PUF can be implanted into a SOPC. The PUF is mounted on PLB bus to connect to a Xilinx MicroBlaze soft-core embedded processor. In our proposed binding scheme, the PUF will only be used when there is a need to unlock the IPs, normally during the FPGA power-up process. When the FPGA is running, the PUF will not be needed anymore. Therefore, we propose to power off the PUF unit once the IPs are unlocked. This mechanism can easily be implemented with some control logic and brings several advantages. First, by shutting down the hard-core or soft-core PUF unit, it will not consume unnecessary power; second, when the PUF unit is off, there will not be any leak of timing, power, or electromagnetic emanation from the PUF unit, so it will be more resilient to potential side channel attacks.

VII. The Security Analysis

The objective of the proposed PUF-FSM binding method is to protect the HWIPs from the piracy attacks such as cloning, copying, unauthorized redistribution, over-use, etc. To analyze the security of this method, we consider the following existing attacks:

- *Brute force*. The adversary tries to guess the correct license to unlock the $b\{HWIP_j\}_{locked}$. By using the unclonable PUF responses to control the transition of the added STG, the space of the correct license becomes exponential, making such brute force attack infeasible. For example, when $L_{PUF} = 256$-bit (License = 128-bit), the search space of such brute force attack will be all the 2^{128} possible license values.
- *PUF removal/tampering attack*. The adversary tries to remove/tamper the PUF on the FPGA, for example by replacing the PUF with a SRAM that contains PUF responses from a previously unlocked hardware IP. Then the license for unlocking the previous HWIP can be used to unlock a new HWIP. There are several countermeasures to address this kind of attack, such as adding obfuscated states within the FSM for PUF checking [27]. Our binding scheme can adopt these countermeasures.
- *Simulating PUF*. According to the intrinsic properties of the PUFs described above, it is impractical to duplicate a PUF with functional and timing characteristics identical to another PUF. Although machine learning

techniques [14] have been used to model some strong PUFs with high prediction rate, they need a huge amount of PUF CRPs during the learning phase. Therefore, this attack will not be effective to weak PUFs such as the one used in this paper, SRAM PUF [18], [29] and similar architectures.

- *Tapping PUF responses*. In the binding scheme, the secret PUF response is ephemeral (the response is only used to unlock HWIPs at boot time) and will be immediately cleared after use, and hence it resists tapping PUF responses.
- *Reverse engineering the added FSM*. An adversary tries to extract the STG and separate/remove the added STG from the original STG. However, STG recovery is a computationally intractable problem [15], [24], [28], and there exist effective methods that we can use in our scheme against such attacks such as creating black holes in the added FSM and merging the added FSM with the test and other FSMs [24].
- *Side channel attacks*. These attacks statistically analyze the time, power consumption or electromagnetic emanation of the cryptographic devices to gain knowledge about integrated secrets. Our delay-based glitch PUF architecture (see Fig. 7) uses multiple flip-flops in parallel and leaves little room for side channel attacks. For electromagnetic emanation analysis, it is practically difficult to locate each flip-flop on the die of an FPGA and to focus the EM probe mainly on the radiation of its components. Timing and power analysis attacks are unlikely because all the flip-flops will be on regardless of the PUF bit will be a 0 or a 1 and the PUF will only be used at the IP unlocking phrase. However, our approach is not completely side channel attack free as it is well-known that any PUF-based security mechanism would be vulnerable to side channel attacks unless appropriate countermeasures are taken [21].

VIII. Experimental Results and Analysis

We have performed a set of experiments to evaluate the effectiveness of the proposed new binding method. The experiments include two parts: the reference implementation of the delay-based PUF on the 28nm Xilinx FPGAs, Zynq-7000 FPGAs; and the evaluation of the PUF-bound FSM on FPGAs.

A. Design Evaluation of a Delay-Based PUF

We implemented 16 identical 64-bit PUFs at different locations on a Xilinx Zynq-7000 FPGA. We used range constraints (ROLC_RANGE statement) supported by the Xilinx integrated development kit to place the PUF design to the designated area. Hence, the responses from these PUFs will be independent. It is well-known that the manufacture variation between two chips is normally larger than the variation between different regions on the same chip. Consequently, if the PUFs located in the different regions on a single FPGA produce unique outputs, we would have the strong confidence that the PUF outputs from different chips should also be unique [41]. In this section, we first show the area overhead caused by the PUFs, and then discuss the uniqueness and reliability of the PUF outputs.

1) Area Overhead: The Xilinx Zynq-7000 XC7Z020 FPGA has about 53,200 LUTs, 17400 of which can be used as storage or shift registers. In our experiments, a 128-bit PUF will consume 258 shift register LUTs (utilization: 1%) and 256 flip-flops (utilization: 1%). Hence, the reference PUF implementation's area overhead can be neglected.

2) Uniqueness: The uniqueness shows how uniquely a PUF response can be, which determines the quality of the PUF. It is not acceptable if different PUFs produce the same or very similar responses when fed with the same challenge. We use Hamming Distance (HD) to evaluate the PUF response's uniqueness. For a pair of n-bit PUF responses: P_i and P_j $(i \neq j)$, their HD is the number of bits that P_i and P_j are different. A PUF response is unique as long as it has a non-zero HD with the responses of other challenges. However, due to reliability concerns (see item *3)* below for more details), the PUF responses under different operating environments may have bit errors and thus produce the same response on different challenges when their designate responses have a very small HD. We define, for k n-bit PUF responses: $P_1, P_2, \cdots, P_k$, their average pairwise HD as:

$$u = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(P_i, P_j)}{n} \times 100\% \tag{3}$$

where,

$$HD(P_i, P_j) = \sum_{m=1}^{n} (r_{i,m} \oplus r_{j,m})$$

$r_{i,m}$ is the m-th response bit of the n-bit response string from PUF P_i.

If each PUF response is unique and logic-0 and logic-1 are distributed in responses uniformly, the expectation of HDs between the PUF responses should be 50%. In our experiment, we use $k = 16$ and $n = 64$. From the $(16 * 15)/2 = 120$ data points of pairwise HD, we have $u = 49.6\%$, which means that on average, any pair of PUF responses have a HD of 31.75 bits.

To further investigate the PUF response's uniqueness, we consider the frequency histogram of these 120 pairwise HD which is shown in Fig. 10. These HDs are concentrated around the expected value of 32 (which is half of 64 bits) with maximum equals to 45 and minimum equals to 18. This implies that for two different challenges to generate the same response, at least one of the PUF response has to have 9 out of the total 64 bits flipped, as we will see next, this is almost impossible to happen with the current PUF technologies. Consequently, we conclude that the implemented PUF achieves good response uniqueness.

3) Reliability: Reliability is used to assess the stability of PUF responses in different environments. Ideally, PUF responses challenged by the same input should remain the same in repeated multiple tests. However, PUF responses may change due to factors such as ambient temperature variation and supply voltage fluctuation since these factors

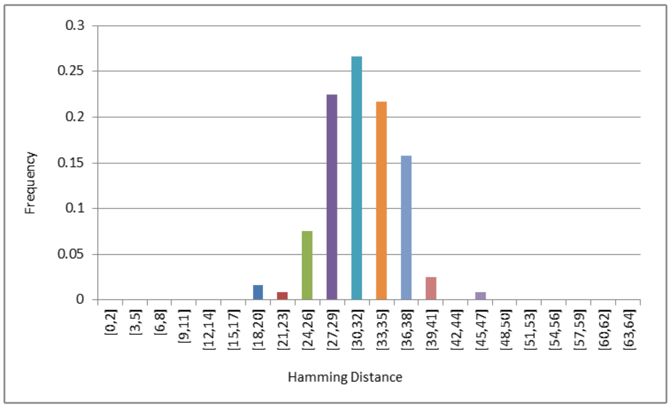


Fig. 10. PUF response uniqueness: Hamming distance distribution.

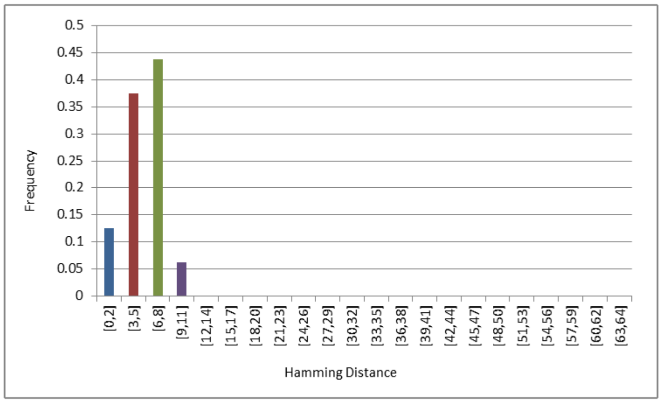


Fig. 11. PUF response variability at high vs. normal temperature.

may affect circuit delay in practice. As a most-effective factor appearing in normal using scenarios, the temperature variation plays very important role to the PUF performance because it can affect the circuit delay. In this study, we select the temperature as the effecting environmental factor to verify the PUF performance. We expect the PUF can also have good uniqueness and reliability when the FPGA runs at high temperature which may be caused by high working load, poor ventilation and high environmental temperature and so on.

We used the Xilinx Chip-Scope to monitor and read the temperature of FPGA chip with PUFs embedded. Room temperature was 15°C, and the normal temperature of the FPGA chip is about 40°C. Then we used an electric hair dryer to raise the FPGA chip temperature to 70°C and tested the PUF responses.

In order to verify the reliability of the PUF at the high temperature, the 64-bit PUF responses were recorded when the FPGA temperatures were about 70°C. The HDs were then calculated between the high-temperature responses and the normal-temperature responses for the 16 testing PUFs as Fig. 11 shows. As for the same PUF under the same challenge at high and normal temperatures, the maximal, minimal and average HD of the responses were 9, 2 and 5.5 respectively; there were 81.25% HDs distributed within the range [3, 8] (small difference). As shown in Fig. 11, when the temperature rose from 40 °C to 70 °C, the reliability degradation was small: the average HD increased from 2.38 to 5.50, and the maximum increased from 6 to 9. From Fig. 10 and Fig. 11, one can see clearly there is no overlap between the number of bit errors at high temperature (maximum to be 9 bits) and the Hamming distance of two different PUF responses (minimum at 18 bits). This phenomenon is known as a vacuum belt: if we use any value x between 10 and 17 as a threshold, when the number of bit difference between the PUF response at run time and the original correct PUF response is less than x, they are errors from the same PUF response; otherwise, they are from different PUF response. Therefore, there will not be any false positive or false negative. As we have discussed earlier in Section VI.B, when the design exhibits large variation to operating environment and there is no vacuum belt, we can apply reliability-enhancing techniques to reduce the bit errors from PUF response.

TABLE II
STATISTICS FOR MCNC'91 BENCHMARKS

Circuit	PI	PO	|S|	T	LUTs	Slices	Delay (ns)	Power (μW)
dk16	2	3	27	108	25	9	1.585	560.25
S832	18	19	25	245	88	44	2.224	560.25
S510	19	7	47	77	38	16	2.461	560.25
S820	18	19	25	232	85	35	2.25	560.49
Styr	9	10	30	166	156	61	2.792	560.25
sand	11	9	32	184	129	38	2.128	560.49
S1488	8	19	48	251	127	54	1.986	560.73
S1494	8	19	48	250	133	60	2.037	560.49
planet	7	19	48	115	140	54	2.377	560.97
S298	3	6	218	1096	245	80	2.895	560.49

B. Overhead Analysis of Modifying FSM

We performed experiments to evaluate the overhead incurred by modifying FSM on the MCNC'91 benchmark sequential circuits and FSM circuits randomly generated by GenFSM [31]. The circuits are described in KISS2 format. Firstly, we use a JAVA program to add the states and transitions for the circuits in KISS2 format, and then use the kiss2vl tool [30] to convert KISS2 to Verilog. Finally, each FSM circuit in Verilog format was synthesized and implemented using the Xilinx ISE 14.1 on the Xilinx Virtex5 FPGA XC5VLX50T, featuring 7200 slices and 28800 Slice LUTs. All experiments were conducted on a 2.4GHz AMD Athlon(tm) 64 Processor 3800+ Dell OptiPlex 740 machine with 1GB RAM.

Table II gives the original synthesis summary conducted on MCNC'91 benchmark designs. The columns "|S|", "PI", "PO" and "T" are the numbers of states, input variables, output variables and transitions, respectively, in each FSM benchmark. The columns "LUTs", "Slices", "Delay" and "Power" are the "Number of Slice LUTs", "Number of occupied Slices", "Minimum period" and the "Estimated power", respectively, of the design with the original FSM as reported by the ISE tools. The Minimum period was obtained by using the Timing Analyzer, and the Power is the estimated power obtained by using the XPower Analyzer.

In our experiment, the number of replicated states m in each odd layer and the number of layers M in the added FSM were

TABLE III

STATISTICS FOR MCNC'91 BENCHMARKS WITH OUR METHOD WHEN $m = 4$ & $M = 4$

Circuit	**LUTs**	**Silices**	**Delay** (ns)	**Power** (μ**W**)	**ΔR-LUTs** (%)	**ΔR-Slices** (%)	**ΔD** (%)	**ΔP** (%)
dk16	45	15	1.830	560.49	80.00	66.67	15.46	0.04
S832	135	70	3.224	561.21	53.41	59.09	44.96	0.17
S510	64	23	2.184	560.49	68.42	43.75	-11.26	0.04
S820	129	55	2.639	560.73	51.76	57.14	17.29	0.04
Styr	200	65	2.38	560.49	28.21	6.56	-14.76	0.04
sand	184	85	3.436	560.49	42.4	123.68	61.47	0
S1488	193	77	2.261	560.97	51.97	42.59	13.85	0.04
S1494	197	78	2.225	560.49	48.12	30.00	9.23	0
planet	182	75	2.383	560.73	30.00	38.89	0.25	-0.04
S298	406	148	4.088	560.49	65.71	85	41.21	0
Mean					52.02	55.34	17.77	0.03

TABLE IV

STATISTICS FOR MCNC'91 BENCHMARKS WITH OUR METHOD WHEN $m = 4$ & $M = 6$

Circuit	**LUTs**	**Silices**	**Delay** (ns)	**Power** (μ**W**)	**ΔR-LUTs** (%)	**ΔR-Slices** (%)	**ΔD** (%)	**ΔP** (%)
dk16	54	17	2.119	560.49	116.00	88.89	33.69	0.04
S832	137	60	2.838	560.49	55.68	36.36	27.61	0.04
S510	75	27	2.125	560.49	97.37	68.75	-13.65	0.04
S820	137	60	2.84	560.49	55.68	36.36	27.61	0.04
Styr	203	58	2.819	560.73	30.13	-4.92	0.97	0.09
sand	205	78	2.47	560.73	58.91	81.58	16.87	0.04
S1488	199	84	2.482	560.73	56.69	55.56	24.97	0
S1494	198	78	2.47	560.49	48.87	30.00	21.26	0
planet	176	78	2.631	560.73	25.71	4.44	10.69	-0.04
S298	256	84	3.214	560.73	4.49	5.00	11.02	0.04
Mean					61.27	49.71	13.91	0.03

TABLE V

STATISTICS FOR LARGE FSMS GENERATED BY GENFSM WITH OUR METHOD WHEN $m = 4$ & $M = 4$

FSM	**Original Details**						**Our Method**							
	|S|	**T**	**LUTs**	**Slices**	**Delay (ns)**	**Power** (μ**W**)	**LUTs**	**Slices**	**Delay (ns)**	**Power** (μ**W**)	**ΔR-LUTs** (%)	**ΔR-Slices** (%)	**ΔD** (%)	**ΔP** (%)
fsm1	300	1200	1105	372	5.621	561.45	1146	405	5.606	561.21	3.71	8.87	-0.27	-0.04
fsm2	350	1400	625	257	4.053	560.97	657	242	4.034	560.97	5.12	-5.84	-0.47	0
fsm3	400	1600	602	203	3.562	560.73	536	207	4.116	560.97	-10.96	1.97	15.55	0.04
fsm4	500	2000	657	247	4.224	560.97	654	247	4.167	561.21	-0.46	0	-1.35	0.04
fsm5	600	2400	1470	524	4.895	560.97	1393	502	4.643	560.73	-5.24	-4.20	-5.15	-0.04
fsm6	800	3200	1992	788	5.163	560.97	1654	585	5.101	561.69	-16.97	-25.76	-1.20	0.13
fsm7	1000	4000	1520	554	5.009	561.69	1574	607	5.284	561.93	3.55	9.57	5.49	0.04
fsm8	1200	4800	2673	962	5.142	561.45	2640	904	5.114	560.73	-1.23	-6.03	-0.54	-0.13
fsm9	1600	6400	2845	976	5.072	561.45	2780	1008	5.11	561.69	-2.28	3.28	0.75	0.04
fsm10	2000	8000	10605	4744	20.558	561.93	10395	2638	19.231	561.45	-1.98	-44.39	-6.45	-0.09
Mean											-2.67	-6.25	0.64	0.01

set as parameters. Table III and Table IV show the synthesis summary on the benchmark circuits processed by our method when ($m = 4$ & $M = 4$) and ($m = 4$ & $M = 6$), respectively. Resources overhead is denoted by the increased "Number of Slice LUTs" and "Number of occupied Slices". Timing overhead is measured by the increased Minimum period. ΔR-LUTs and ΔR-Slices are normalized resources overhead in our proposed scheme in LUTs and Slices, respectively. ΔD, and ΔP are the normalized overhead in delay and power, respectively. We can see from Table III and Table IV that the resources, timing and power overhead due to modifying FSM seems to be independent of the benchmark circuit size. The average resources, power and timing overhead is 52.02% for LUTs (55.34% for Slices), 11.77% and 0.03% when $m = 4$ & $M = 4$; and 61.27% for LUTs (49.21% for Slices), 13.91% and 0.03% when $m = 4$ & $M = 6$. The Table III and Table IV reveal that the power is rather low, and the timing degradation is moderate (11.77% for Table III and 13.91% for Table IV on average) and even negative in some instances. A negative percentage implies that our method has actually improved the performance. The high area overhead and moderate timing overhead on these small benchmark circuits is a direct result of the simplicity of these circuits as they contain only control paths. In practice, an actual HWIP would be much larger with lots of other components in addition to control paths. In those cases, we expect the overhead to be small.

To demonstrate this, we use GenFSM [31] to generate ten random STGs with hundreds of states and hundreds to thousands of state transitions for experimentation by specifying the number of inputs, outputs and states. The experimental results, as shown in Table V, indicate that our method introduces very

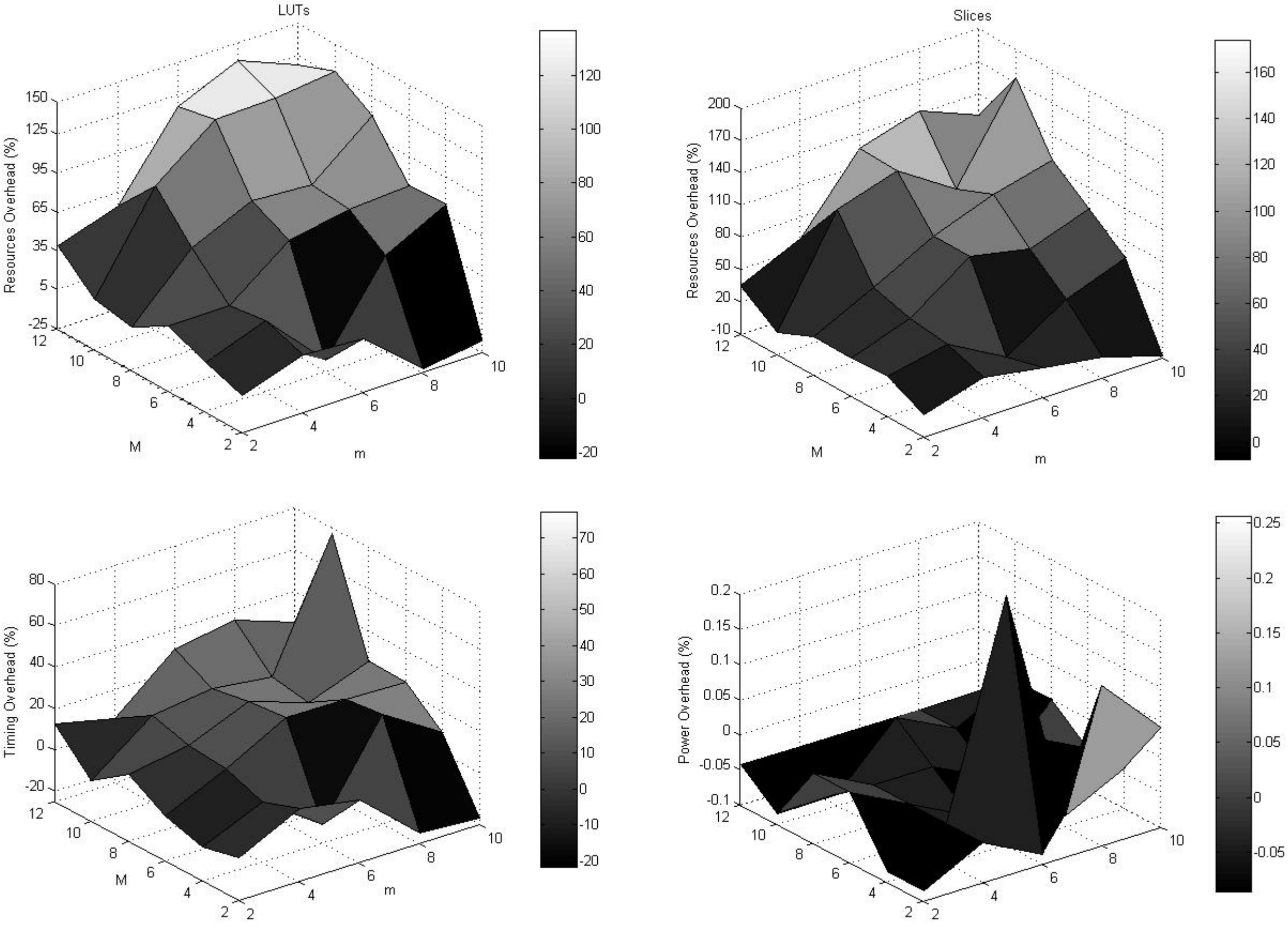


Fig. 12. Area, delay, and power overhead with different M = (2,4,6,8,10,12) and m = (2,4,6,8,10) for circuit *planet*.

low "ΔR-LUTs", "ΔR-Slices", "ΔD" and "ΔP" for large FSM designs, the average of resources, timing and power overhead are −2.67% for LUTs (−6.25% for Slices), 0.64% and 0.01% respectively. In addition, it must be noted that the overhead could be much less in large designs where there are many components other than control paths, such as memory and I/O peripheries. Furthermore, the control path realized by the binding FSM on each benchmark is only a small part of the overall size of the design ($\leq$ 1%) [23]. Therefore, adding more control paths to the binding FSM would be acceptable with low overhead on area, timing and power in practice.

And finally, we discuss the impact of various m and M on resources, timing and power overhead for benchmarks. Fig. 12 shows the impact of various M and various m on resources, timing and power overhead for benchmark *planet* when M was assigned to 2, 4, 6, 8, 10, 12 and m was assigned to 2, 4, 6, 8, 10, successively. It can be seen that the power overhead is negligible, and the resources and timing overhead are roughly positive-correlated to both M and m, but nonlinear due to the optimization of the circuits during synthesis.

From the above results, we see that the PUF-FSM binding scheme can control the overhead on the area, power and timing especially in large designs. Proper m and M values can also be considered near the empirically proper values.

IX. Conclusion and Future Work

This article presents a new binding method that enables binding hardware IPs to specific FPGAs by utilizing the PUF and the FSM of circuits. The method is fundamentally different from the traditional encryption-based HWIP protection methods and offers the following advantages: 1) it can be used to protect the third-party FPGA IP cores in addition to the single FPGA configuration bitstream; 2) it does not need any third parties or permanent storage for secret keys in the FPGA; 3) it supports the pay-per-device licensing mechanism; and 4) it has low hardware cost. Experimental results on a reference implementation of the binding scheme show that a 128-bit delay-based PUF utilizes only 258 RAM-LUTs and 256 flip-flops on 28nm Xilinx FPGAs and the modified FSM only introduces 0.64% timing overhead and 0.01% power overhead on average for ten FSM designs randomly generated by GenGSM.

We conclude with a discussion on the limitations of our PUF-FSM binding scheme which lead to several future research directions. First, in our approach, we modify the FSM of a design to lock it and use the PUF response to unlock it. This effectively protects the whole design, not only the FSM, from attacks such as cloning, copying, misusing and unauthorized integration. However, the design components without bound FSMs will still be vulnerable to tamping attacks. Anti-tampering is beyond the scope of this article, but it will be interesting to study how our approach can be combined with anti-tamper methods such as those described in [12]. Second, although we have argued that our approach is more resilient again physical attacks than encryption-based IP protection method. From the physical security perspective, it is well-known that any implementation of a cryptographic primitive and PUF-based security mechanism would be vulnerable to side channel attacks when no appropriate countermeasures are taken [21]. It will be of high interest to develop effective countermeasures to enhance resiliency against various types of physical attacks.

Acknowledgment

The authors would like to thank Dr. Qiang Wu, Dr. Qiang Zhou, Wenjie Che and Kecheng Yang for reviewing this article and providing us feedback. We would also like to thank the anonymous reviewers for their insightful suggestions and comments.

References

[1] S. Drimer, “Security for volatile FPGAs,” Ph.D. dissertation, Dept. Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-763, Nov. 2009.

[2] R. Maes, D. Schellekens, and I. Verbauwhede, “A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 98–108, Feb. 2012.

[3] T. Kean, “Cryptographic rights management of FPGA intellectual property cores,” in *Proc. ACM/SIGDA 10th Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2002, pp. 113–118.

[4] T. Güneysu, B. Möller, and C. Paar, “Dynamic intellectual property protection for reconfigurable devices,” in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2007, pp. 169–176.

[5] S. Drimer, T. Güneysu, M. G. Kuhn, and C. Paar. (2008). *Protecting Multiple Cores in a Single FPGA Design*. [Online]. Available: http://www.cl.cam.ac.uk/~sd410/papers/protect_many_cores.pdf

[6] “Design security in Stratix III devices (v1.5),” Altera, San Jose, CA, USA, White Paper 01010, Sep. 2009.

[7] “Using high security features in Virtex-II series FPGAs (v1.0),” Xilinx, San Jose, CA, USA, Appl. Note 766, Jul. 2004.

[8] S. Trimberger, J. Moore, and W. Lu, “Authenticated encryption for FPGA bitstreams,” in *Proc. 19th ACM/SIGDA Symp. Field-Program. Gate Arrays (FPGA)*, 2011, pp. 83–86.

[9] “Protecting the FPGA design from common threats (v1.0),” Altera, San Jose, CA, USA, White Paper 01111, Jun. 2009.

[10] G. E. Suh and S. Devadas, “Physical unclonable functions for device authentication and secret key generation,” in *Proc. 44th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2007, pp. 9–14.

[11] M. A. Gora, A. Maiti, and P. Schaumont, “A flexible design flow for software IP binding in FPGA,” *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 719–728, Nov. 2010.

[12] S. J. Stone, “Anti-tamper method for field programmable gate arrays through dynamic reconfiguration and decoy circuits,” M.S. thesis, Dept. Elect. Comput. Eng., Air Force Inst. Technol., Wright-Patterson Air Force Base, OH, USA, 2008.

[13] K. Kepa, F. Morgan, and K. Kosciuszkiewicz, “IP protection in partially reconfigurable FPGAs,” in *Proc. IEEE Int. Conf. Field-Program. Logic Appl. (FPL)*, Aug./Sep. 2009, pp. 403–409.

[14] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 237–249.

[15] A. L. Oliveira, “Robust techniques for watermarking sequential circuit designs,” in *Proc. 36th Annu. ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 1999, pp. 837–842.

[16] E. Simpson and P. Schaumont, “Offline hardware/software authentication for reconfigurable platforms,” in *Proc. 8th Int. Conf. Cryptogr. Hardw. Embedded Syst. (CHES)*, 2006, pp. 311–323.

[17] M. Majzoobi and F. Koushanfar, “Time-bounded authentication of FPGAs,” *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 1123–1135, Sep. 2011.

[18] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “FPGA intrinsic PUFs and their use for IP protection,” in *Proc. 9th Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, 2007, pp. 63–80.

[19] “Security solutions using Spartan-3 generation FPGAs (v1.1),” Xilinx, San Jose, CA, USA, White Paper 266, Apr. 2008.

[20] J. A. Roy, F. Koushanfar, and I. L. Markov, “EPIC: Ending piracy of integrated circuits,” in *Proc. Eur. Design Test Conf. (DATE)*, 2008, pp. 1069–1074.

[21] R. Maes, A. Van Herrewege, and I. Verbauwhede, “PUFKY: A fully functional PUF-based cryptographic key generator,” in *Proc. 14th Int. Conf. Cryptogr. Hardw. Embedded Syst. (CHES)*, 2012, pp. 302–319.

[22] M. Majzoobi, A. Kharaya, F. Koushanfar, and S. Devadas. (2014). “Automated design, implementation, and evaluation of arbiter-based PUF on FPGA using programmable delay lines,” Rice Univ., Houston, TX, USA, Tech. Rep. 2014/639. [Online]. Available: http://eprint.iacr.org/

[23] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 4th ed. San Mateo, CA, USA: Morgan Kaufmann, 2006.

[24] Y. M. Alkabani and F. Koushanfar, “Active hardware metering for intellectual property protection and security,” in *Proc. 16th USENIX Secur. Symp.*, 2007, pp. 291–306.

[25] Y. Alkabani, F. Koushanfar, and M. Potkonjak, “Remote activation of ICs for piracy prevention and digital right management,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2007, pp. 674–677.

[26] Y. Alkabani and F. Koushanfar, “Active control and digital rights management of integrated circuit IP cores,” in *Proc. Int. Conf. Compil., Archit., Synth. Embedded Syst.*, 2008, pp. 227–234.

[27] F. Koushanfar, “Provably secure active IC metering techniques for piracy avoidance and digital rights management,” *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 51–63, Feb. 2012.

[28] A. Cui, C.-H. Chang, S. Tahar, and A. T. Abdel-Hamid, “A robust FSM watermarking scheme for IP protection of sequential circuit design,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 5, pp. 678–690, May 2011.

[29] D. E. Holcomb, W. P. Burleson, and K. Fu, “Power-up SRAM state as an identifying fingerprint and source of true random numbers,” *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[30] C. Pruteanu. (2000). *Kiss to Verilog FSM Converter*. [Online]. Available: http://codrin.freeshell.org/

[31] C. Pruteanu and C.-G. Haba, “GenFSM: A finite state machine generation tool,” in *Proc. 9th Int. Conf. Develop. Appl. Syst.*, 2008, pp. 165–168.

[32] A. B. Kahng *et al.*, “Watermarking techniques for intellectual property protection,” in *Proc. 35th Annu. Design Autom. Conf. (DAC)*, 1998, pp. 776–781.

[33] G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Designs: Theory and Practice*. Boston, MA, USA: Kluwer, 2003.

[34] J. Zhang, Y. Lin, Q. Wu, and W. Che, “Watermarking FPGA bitfile for intellectual property protection,” *Radioengineering*, vol. 21, no. 2, pp. 764–771, 2012.

[35] J. Zhang, Y. Lin, W. Che, Q. Wu, Y. Lyu, and K. Zhao, “Efficient verification of IP watermarks in FPGA designs through lookup table content extracting,” *IEICE Electron. Exp.*, vol. 9, no. 22, pp. 1735–1741, 2012.

[36] J. Zhang, Y. Lin, and G. Qu, “Reconfigurable binding against FPGA replay attacks,” *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 2, Feb. 2015, Art. ID 33.

[37] J. Zhang *et al.*, “FPGA IP protection by binding finite state machine to physical unclonable function,” in *Proc. 23rd Int. Conf. Field-Program. Logic Appl. (FPL)*, Sep. 2013, pp. 1–4.

[38] F. Koushanfar and G. Qu, “Hardware metering,” in *Proc. 38th Annu. Design Autom. Conf. (DAC)*, 2001, pp. 490–493.

[39] J.-L. Zhang, G. Qu, Y.-Q. Lv, and Q. Zhou, “A survey on silicon PUFs and recent advances in ring oscillator PUFs,” *J. Comput. Sci. Technol.*, vol. 29, no. 4, pp. 664–678, 2014.

[40] M.-D. Yu and S. Devadas, “Secure and robust error correction for physical unclonable functions,” *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan./Feb. 2010.

[41] J. H. Anderson, “A PUF design for secure FPGA-based embedded systems,” in *Proc. 15th Asia South Pacific, Design Autom. Conf. (ASP-DAC)*, 2010, pp. 1–6.

[42] R. Maes and I. Verbauwhede, “Physically unclonable functions: A study on the state of the art and future research directions,” in *Towards Hardware-Intrinsic Security*. Berlin, Germany: Springer-Verlag, 2010, pp. 3–37.

[43] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Proc. Int. Conf. Field Program. Logic Appl. (FPL)*, Aug. 2007, pp. 189–195.

[44] Z. Paral and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, Jun. 2011, pp. 128–133.

[45] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proc. IEEE Symp. Secur. Privacy Workshops (SPW)*, May 2012, pp. 33–44.

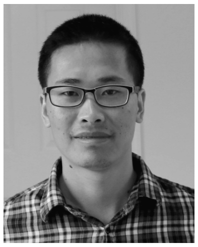
[46] G. Qu and C.-E. Yin, "Temperature-aware cooperative ring oscillator PUF," in *Proc. IEEE Int. Workshop Hardw.-Oriented Secur. Trust (HOST)*, Jul. 2009, pp. 36–42.

[47] A. Maiti, L. McDougall, and P. Schaumont, "The impact of aging on an FPGA-based physical unclonable function," in *Proc. Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2011, pp. 151–156.

[48] D. Ganta and L. Nazhandali, "Study of IC aging on ring oscillator physical unclonable functions," in *Proc. 15th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2014, pp. 461–466.

[49] M. Rahman, D. Forte, J. Fahrny, and M. Tehranipoor, "ARO-PUF: An aging-resistant ring oscillator PUF design," in *Proc. Design, Autom., Test Eur. Conf. Exhibit. (DATE)*, 2014, pp. 1–6.

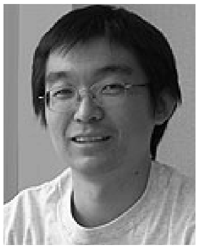
[50] R. Maes and V. van der Leest, "Countering the effects of silicon aging on SRAM PUFs," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2014, pp. 148–153.



Jiliang Zhang received the B.E. degree in chemical engineering and technology from the Shandong University of Science and Technology, Qingdao, China, in 2009, and the Ph.D. degree in computer application technology from Hunan University, Changsha, China, in 2015. From 2013 to 2014, he was a Research Scholar with the Maryland Embedded Systems and Hardware Security Laboratory, University of Maryland, College Park, MD, USA. His research interests include hardware security, such as security for field-programmable gate arrays, PUF and PUF-related applications, IC obfuscation, and IP protection.

Yaping Lin received the B.S. degree from Hunan University, Changsha, China, in 1982, the M.S. degree from the University of Defense Technology, Changsha, in 1985, and the Ph.D. degree from Hunan University, in 2000. From 2004 to 2005, he was a Visiting Scholar with the University of Texas at Arlington, Arlington, TX, USA. He is currently a Professor with the College of Information Science and Engineering, Hunan University. His primary research interests are in the area of computer networking and information security with a focus on sensor networks, cloud security, and hardware related security.

Yongqiang Lyu received the B.S. degree in computer science from Xidian University, Xi'an, China, in 2001, and the M.S. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2003 and 2006, respectively. He is currently an Assistant Professor with the Research Institute of Information Technology, Tsinghua University. His research interest focuses on the hardware–software fusion architecture in emerging computing systems.



Gang Qu (SM'07) received the B.S. and M.S. degrees in mathematics from the University of Science and Technology of China, Hefei, China, in 1992 and 1994, respectively, and the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2000. Upon graduation, he joined the University of Maryland, College Park, MD, USA, where he is currently a Professor with the Department of Electrical and Computer Engineering and the Institute for Systems Research. He is a member of the Maryland Cybersecurity Center and the Maryland Energy Research Center. He is the Director of Maryland Embedded Systems and Hardware Security Laboratory, College Park, and the Wireless Sensors Laboratory.

His primary research interests are in the area of embedded systems and very large scale integration (VLSI) computer aided design (CAD) with a focus on low power system design and hardware related security and trust. He studies optimization and combinatorial problems and applies his theoretical discovery to applications in VLSI CAD, wireless sensor network, bioinformatics, and cybersecurity. He has received many awards for his academic achievements, teaching, and service to the research community. He serves as an Associate Editor of the IEEE EMBEDDED SYSTEMS LETTERS, and the *Integration, the VLSI Journal*.