Combining PUF with RLUTs: A Two Party Pay Per Device IP Licensing Scheme On FPGAs

DEBAPRIYA BASU ROY, Indian Institute of Technology Kharagpur, India
SHIVAM BHASIN, Temasek Laboratories, Nanyang Technological University, Singapore
IVICA NIKOLIĆ, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
DEBDEEP MUKHOPADHYAY, Indian Institute of Technology Kharagpur , India

With the popularity of modern FPGAs, the business of FPGA-specific intellectual properties (IP) is expanding rapidly. This also brings in the concern of IP protection. FPGA vendors are making serious efforts for IP protection leading to standardization schemes like IEEE P1735. However, efficient techniques to prevent unauthorized overuse of IP still remain an open question. In this paper, we propose a two-party IP protection scheme combining the re-configurable LUT (RLUT) primitive of modern FPGAs with physical unclonable functions (PUF). The proposed scheme works with the assumption that the FPGA vendor provides the assurance of confidentiality and integrity of the developed IP. The proposed scheme is considerably lightweight compared to existing schemes, prevents overuse and does not require any additional functionality of FPGA vendors apart from the service of providing confidentiality and integrity of the developed IP. The validation of the proposed scheme is done on MCNC'91 benchmarks and third party IPs like AES and lightweight MIPS processor.

ACM Reference Format:
Debapriya Basu Roy, Shivam Bhasin, Ivica Nikolić, and Debdeep Mukhopadhyay. 2020. Combining PUF with RLUTs: A Two Party Pay Per Device IP Licensing Scheme On FPGAs. 1, 1 (November 2020), 22 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

FPGAs have come a long way from basic prototyping platforms to effective alternative to ASICs. Due to the increased demand of FPGA based applications, IP integration model is becoming popular. To cope with the increasing constraints on area/performance, designers depend on proven third party IP (intellectual property). This considerably reduces the time to market and allows the system developer to concentrate on other critical parts of the design.

This type of modular design strategy is suitable for FPGAs. For example, FPGA vendor Xilinx provides *Plug-and-Play IP Initiative* [38] which ensures smooth integration of third party IPs into the design netlist. However, due to its flexibility, protecting FPGA based third party IP from over deployment is a challenging problem. Market estimates for losses due to IP counterfeiting is over 10% of the global trade.

Authors' addresses: Debapriya Basu Roy, Indian Institute of Technology Kharagpur, India, deb.basu.roy@cse.iitkgp.ernet.in; Shivam Bhasin, Temasek Laboratories, Nanyang Technological University, Singapore, sbhasin@ntu.edu.sg; Ivica Nikolić, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, inikolic@ntu.edu.sg; Debdeep Mukhopadhyay, Indian Institute of Technology Kharagpur , India, debdeep@cse.iitkgp.ernet.in.

XXXX-XXXX/2020/11-ART $15.00
https://doi.org/0000001.0000001

Generally, the IP protection mechanism is analyzed in two different directions: detection and protection against IP violation. IP violation detection exploits *watermarking* techniques, which a vendor can verify to claim ownership rights. IP violation protection is a far more challenging task, where the IP vendor needs to prevent illegal exploitation and misuse of the supplied IP. The common threats in context to IP violation are reverse engineering, cloning, and overuse. Moreover, threats of malicious hardware Trojan insertion can also cause legal issues for IP vendors.

The prime focus of this paper is to provide an IP protection mechanism. There are various threat scenarios that have to be considered. The first and the obvious threat is reverse engineering. In this scenario, an adversary tries to rebuild the source code from the delivered IP. Once rebuilt, this source code can be sold for a more competitive price, considerably reducing the profit margin of the IP vendor. Another obvious threat to FPGA based IPs is cloning where the malicious user eavesdrops during the FPGA configuration and can directly copy the FPGA bitstream. In the following, IP user and IP client are used interchangeably.

On Xilinx devices, a cyclic redundancy check (CRC) is run on the bitstream during the configuration stage. Additionally, cloning and reverse engineering can be prevented by solutions based on netlist/bitstream encryption and authentication. However, physical attacks on bitstream security have already been practically demonstrated [26]. Moreover, threats like hardware Trojans and counterfeit IC chips are discussed in great details in [27, 28, 34, 36]. To ensure integrity and confidentiality of the third-party IP, FPGA vendors deploy design practices based on the *IEEE P1735* [1] standard. But, these security practices do not address the threat of overproduction or over-deployment of IPs by a malicious user which is the precise objective of the present work. In this scenario, the client has rights for q instances of the concerned IP, but deploys greater than q instances. This threat leads to an inefficient business model where IP providers charge large upfront fees to account for overproduction of the IP. Such large fee imposes a considerable financial constraint on the customers with low to medium volume of IP requirement. This scenario can be considerably improved by introducing a licensing scheme for the IPs which will force the customer to pay the IP provider for usage of each instance of any IP (*pay per device*), allowing the IP provider to sell the IPs at a lower cost as the threat of overproduction gets eliminated.

This paper proposes a two party pay per device licensing scheme for IP protection. We state our proposed scheme as two party because the FPGA vendor does not need to do anything additional to prevent IP overproduction apart from providing assurance of confidentiality and integrity of the IP which an FPGA vendor is already providing for secure third party IP integration. The scheme is composed of: (1) physical unclonable functions (PUF) for device binding or authentication, (2) symmetric key encryption for provable IP protection at low cost and (3) reconfigurable look up tables (RLUT) for IP activation. In this case-study, we focus on the finite state machine (FSM) which is used to design the control units of digital circuits. Nevertheless, the proposed strategy can be applied to any combinational circuits as well. According to our proposed scheme, IP vendors first design the FSM of the IP using RLUT, where RLUT contains an incorrect configuration upon delivery. The client then obtains an authentication tag from the vendor, which is device-bounded due to the built-in PUF. The correct tag must be applied to restore the correct FSM configuration and make that particular device functional. In this case, the user must pay for each authentication tag, which prevents overusing of IP. The advantages of the proposed scheme can be tabulated as below:

- This proposed scheme works in conjunction with the IEEE P1735 protocol and can be easily integrated with existing design flow of FPGA based applications. In our scheme, we assume that the FPGA vendor is trusted to provide confidentiality and integrity of the developed IP

which they already provides for third party IP integration. Apart from this, the only involved parties are IP vendors and clients.
- The scheme has little area overhead on the design with negligible timing overhead. Once the IP is reconfigured by the client with the correct authentication tag, it works without any extra timing requirement.
- The scheme is equally applicable to both combinational and sequential circuits. We concentrate mainly on the FSM of the developed IP. However, the scheme also can be applied to combinational logic.

The rest of the paper is organized as follows: in the next section, we will highlight important literature regarding IP protection mechanism on FPGA and general background on RLUT and PUF. We will focus on the detailed portrayal of the proposed scheme in Section 3, followed by security analysis in Section 4. Section 5 will describe the implementation details and architectural description of the proposed scheme on FPGA. In Section 6 we will show the performance of the proposed scheme in terms of area and timing compared to existing IP protection mechanisms. Finally, in Section 7, we will conclude the paper.

2 GENERAL BACKGROUND

This section provides a concise overview of the related works on IP protection scheme followed by brief descriptions of RLUT, PUF, and the IEEE P1735 protocol.

2.1 Related Work

Eavesdropping and cloning of FPGA bit-stream is a threat to SRAM based FPGA. One straightforward solution to prevent such attack is to replace SRAM with non-volatile Flash ROM. Alternatives include Flash-based FPGA which have higher manufacturing cost [3]. But SRAM based FPGAs make 76.1% of the global sale revenue and hence replacing them is not a viable solution [35]. Some FPGA vendors, to add protection to SRAM FPGA, support symmetric key cipher based bit-stream encryption [7]. However, this does not prevent overuse of IP.

Some previous work address pay per device IP licensing model to prevent overuse. In [18], public key based key derivation function (KDF) is used for secure transmission of the IP decryption key. The advantage of this approach is that it does not create any area overhead as KDF is implemented on a temporary configuration bit-stream and can be removed once the key establishment procedure is completed. This scheme involves trusted FPGA vendors in the protocol establishment. On the other hand, the scheme in [23] involves another party known as the trusted third party (TTP) in the protocol execution. Due to the deployment of TTP, the KDF function can be replaced by a cheaper symmetric key execution. However, assuming the existence of a TTP has its own pitfall as the security of the entire protocol now depends upon TTP, which becomes a *single point of complete failure* [7]. On the contrary, in [7], the author allows FPGA vendors to take responsibility of TTP but does not allow it to have access to the secret information critical for IP protection. The security of this scheme depends on the problem of prime number factorization. Additionally, this scheme requires implementation of modular reduction along with a decryption module which may incur a significant area overhead on the design.

Apart from employing standard cryptographic practices, the researchers have also applied PUFs to ensure IP protection. In [32], authors have employed a PUF along with a decryption and authentication module for IP protection. One advantage of employing PUF is simpler communication between different parties compared to on-chip secret key based protocols [33]. The construction in [32] requires execution of two decryption operations and one hash operation. It is further improved in [19] where the protocol requires execution of only one decryption and one hash

operation. On the other hand, in [20], the authors have employed zero knowledge proof for certification of public keys of FPGA and IP vendors. But, this protocol requires implementation of elliptic curve crypto-system on the FPGA which again imposes significant area overhead on the design. In [4], the authors have provided a methodology where an IP vendor can deploy multiple numbers of IPs and can give the clients access to a subset of those IPs depending upon the requirement. Like other proposals, this proposal also assumes the existence of a TTP.

Recently, in [40], the authors have presented a PUF based IP protection mechanism which does not involve any encryption or decryption module in the design. In this scheme, PUF output is used for FSM transition. Hence, any user without the correct challenge value will not be able to drive the FSM in the correct direction. Thus, the design becomes bound to the device and the user cannot create a copy as PUF on different devices will have different responses to the same challenge. However, this scheme can only be applied on FSMs. Additionally, the absence of any cryptographic primitive in [40] was criticised in [9]. Usage of PUF for providing IP protection scheme was also proposed in [10, 14]. But the solution proposed in [10, 14] requires an enrolment stage for creating challenge-response pair database which is not required in our proposed methodology.

Schemes like *Ending Piracy of Integrated Circuits (EPIC)* [30] have used logic encryption in combination with public key algorithms to provide solutions for preventing IP privacy. This scheme was improved in [21] where they have used an advanced logic encryption technique along with public key cryptography to provide protection from *IP piracy*, *IP overproduction*, and *IP overuse*. The issue of IP overuse has been extensively discussed in [21] from point of view of ASIC design. The proposed solution involves combination of logic encryption, public key cryptosystem for signature generation and IEEE P1735 protocol. Contrary to these works [21, 30], in this paper, we propose a strategy which is exclusive to FPGA based applications. The solution, proposed in [21], assumes the existence of *Trusted Authentication Platform*, which is analogous to trusted third parties, inside a chip and employs computationally extensive public key algorithms (RSA). Our approach, on the other hand is essentially a two party scheme with the assumption that the FPGA vendor is trusted to provide confidentiality and maintain integrity of the IP. We state our proposed scheme as two party because the FPGA vendor does not need to do anything additional to prevent IP overproduction apart from providing assurance of confidentiality and integrity of the IP which they are already providing for secure third party IP integration. Additionally, we use a very lightweight encryption scheme to achieve our goal of preventing IP overuse. It must be noted that the problem of *IP overproduction* does not exist for FPGA based applications as no foundry is involved during the IP integration. Hence, we have focused on the problem of *IP overuse* only.

In [11], the authors have proposed an IP protection strategy: HARPOON for ASIC platform. The proposed methodology is based on obfuscation of netlist. This approach is based on creating an obfuscated FSM which can be enabled by providing the correct obfuscation key. This scheme can be elevated to support user authentication by incorporating signature mechanisms like watermarking. Additionally, the authors have recommended usage of PUFs for instance specific obfuscation and authentication. The main difference from the methodology presented in [11] and our work is the implementation platform. In this work, our focus is on FPGA where implementing pay per device licensing scheme is more challenging compared to ASIC due to the inherent programmability of FPGAs. Furthermore, the work in [11] and [40] are somewhat similar as both are based on obfuscation of FSM. On the other hand, our strategy does not involve any obfuscation methodology. Rather, we generate configuration data of a part of the circuit in run time using the correct combination of PUF response, a lightweight cryptographic algorithm and RLUT feature of the modern FPGAs. In the case studies that we will present in this paper, we have chosen FSM for easy comparison with existing work; however, any combinatorial and sequential logic will serve our purpose. Additionally, the protocol presented in [40] involves the FPGA vendors as it assumes

that the PUF is implemented on the FPGA by the FPGA vendors, which requires modification of standard FPGA manufacturing process and design practices.

Table 1. Comparative analysis of existing pay per device IP licensing schemes

References	On-Chip Asymmetric Cryptography	Two Party	PUF/TRNG	Logic Encryption / Obfuscation	Symmetric Cryptography	IEEE P1735
[18]	√	×	×	×	√	×
[23]	×	×	×	×	√	×
[7]	√	×	×	×	√	×
[32]	×	×	√	×	√	×
[19]	×	×	√	×	√	×
[20]	√	×	√	×	×	×
[4]	×	×	√	√	×	×
[40]	×	×	√	√	×	×
[30]	√	×	√	√	×	×
[21]	√	×	√	√	×	√
[11]	×	×	√	√	×	×
Proposed Work	**×**	**√**[1]	**√**	**×**	**√**	**√**

A comparative study between the existing IP licensing protocol and our proposed protocol is provided in Table 1. All the above-mentioned protocols suffer from two major drawbacks. They either require the implementation of resource consuming public key cryptography (PKC) or assume the existence of trusted third parties. Some of the discussed schemes need both PKC and TTP.

The scheme proposed in this paper uses the industry standard protocol *IEEE P1735* for ensuring confidentiality and integrity of the third party IPs. It must be noted that as per IEEE P1735 protocol, FPGA vendor needs to maintain the confidentiality and integrity of the developed IP. For the execution of the proposed protocol, no additional functionalities are needed to be performed by the FPGA vendor. Apart from this, our protocol involves only two parties: IP vendor and IP user or system integrator (client/user). The proposed scheme exploits PUF, lightweight cryptography, and RLUTs to incur lower area/time overheads.

2.2 Reconfigurable LUT (RLUT)

The RLUT feature can be found in *Xilinx* FPGAs as *CFGLUT5*. It is a 5-input LUT with a single output which can be reconfigured on run time. The reconfiguration mechanism is similar to a shift register or the more popularly known *SRL32* feature of the *Xilinx* FPGAs. It is also possible to realize RLUT like functionality on devices from other FPGA vendors like *Altera* or *Microsemi*. For instance, the *MLAB* primitive from *Altera* can be used to implement an RLUT. Similarly, *Microsemi* and old *Xilinx* also support *SRL* which can be extended to RLUTs. Thus, our results will directly apply to alternatives of *Xilinx* as well.

The basic block diagram of CFGLUT5 is shown in Fig. 1. It can be used as a 5-input and a 1-output LUT or 4-input and 2-output LUT. *INIT* value defines the truth table of the LUT function. In CFGLUT5, this *INIT* can be modified by using the *CE* or the configuration enable port. The new value of *INIT* is fed through CD_I port in a bit by bit fashion serially. Previous *INIT* value is

[1]FPGA vendor provides confidentiality and integrity of the developed IP

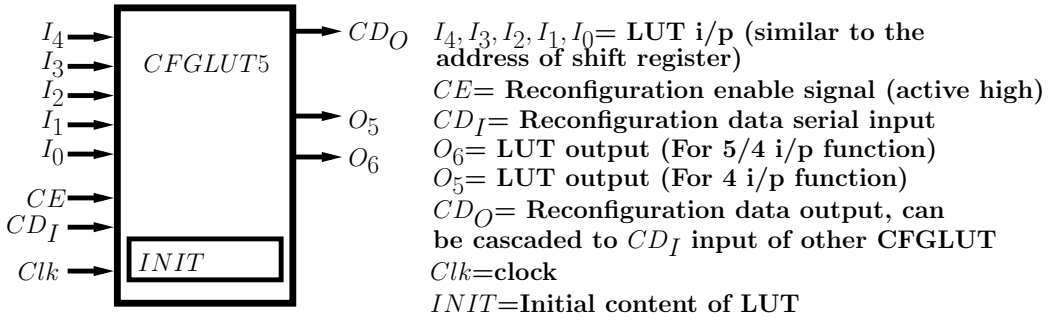

Fig. 1. Block diagram of CFGLUT5 [29]

removed using CD_O port, which can be either reused or flushed out. One bit of new *INIT* can be shifted every clock cycle, thus *INIT* can be changed completely in 32 clock cycles. Moreover, the CFGLUT5 functionality only applies to the *SLICE_M* slices or the memory slices of *Xilinx* FPGA. Several security related applications of RLUT are presented in [29].

2.3 Physical Unclonable Function (PUF)

Physical Unclonable Functions (PUF) can be used as the fingerprint of an integrated circuit. Due to the inherent process variation, certain physical properties of the device varies from one sample to another. The physical difference or the signature can be accessed using a challenge-response protocol. This physical difference results in a different response to a given challenge from one sample to another. To measure the quality of a PUF, three parameters are observed: 1) Randomness: PUF responses should be random enough so that they cannot be predicted by any adversary. 2) Reliability: On a particular device for some particular challenge, PUF response should be identical for all working condition. 3) Uniqueness: Responses of a PUF on a particular device should be unique and distinctive from the responses of other devices. Ideally, a PUF should exhibit 100% reliability. However, due to temperature and voltage variations, reliability never reaches the desired 100%. To mitigate this issue, several techniques like error correcting codes and fuzzy extractors [17] have been proposed which enhances the reliability and maintain the consistent response. In the following, we assume that the used PUF with a stable response is available to the IP vendor.

2.4 IEEE P1735 Protocol

IEEE P1735 is industry adopted protocol for the third party IP integration. The objective of this protocol is to ensure the confidentiality of the third party IP. Like any confidentiality preserving scheme, this protocol employs encryption algorithms to encrypt the source IP. The working of this protocol is shown in Fig. 2.

At the first step, the IP is encrypted by a symmetric key algorithm (AES or DES in CBC mode) with a random session key. This key is then encrypted by an asymmetric key algorithm (RSA) with the public key of the EDA tool platform. Once the session key is encrypted, it is bundled with the encrypted IP to create a single file. This file is then sent to the IP customer. Integration of the encrypted IP with the other design modules is carried out by the same EDA tool. The EDA tool uses its private key to recover the secret session key and decrypt the IP. The protocol ensures that during any step of design integration (simulation, synthesis, mapping, and routing), the internal configuration and design principal of the third party IPs remain secret to the integrator.

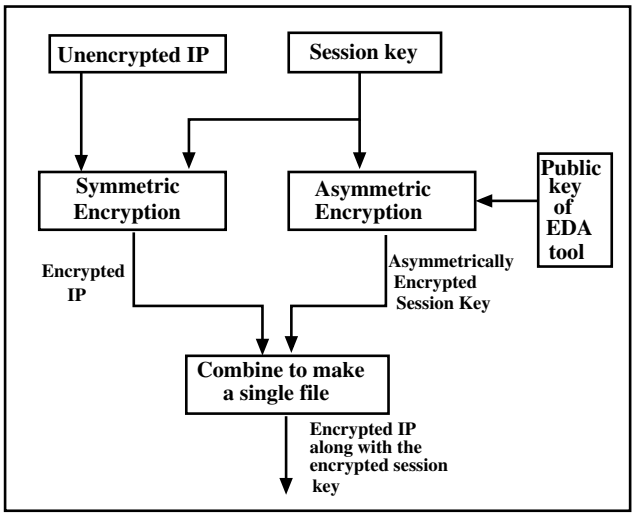

Fig. 2. IEEE P1735 protocol

IEEE P1735 protocol though ensures confidentiality of the IP, it cannot prevent the overuse of it. However, this protocol already employs a combination of symmetric and asymmetric key algorithm which can be extremely useful for the proposed pay per device licensing scheme. Our objective is to integrate already existing IEEE P1735 protocol with the proposed scheme to ensure both the confidentiality of the IP and pay per device licensing.

2.5 Vulnerability of IEEE P1735 Protocol

In the recent work [13], the authors have shown the vulnerability in the construction of the IEEE P1735 protocol. The authors have identified multiple cryptographic mistakes in the protocol structure, allowing an attacker to recover the entire third-party IP in plaintext. Apart from this, the attacker can also develop a standard-compliant third-party IP after including stealthy and malicious hardware Trojans. Thus, the two major objectives of the IEEE P1735 protocol, confidentiality, and integrity of the third-party IPs, are compromised in this scenario. It must be noted that the proposed protocol uses a secured IP encryption algorithm to achieve the overall pay per device framework. However, the protocol is developed in a modular fashion, ensuring that the currently used IEEE P1735 protocol can be replaced by any state of the art secured algorithm without any change in our proposed protocol. We hope that such an algorithm would be soon developed to address the issue of IP security.

3 PROPOSED IP PROTECTION SCHEME

This section provides a detailed description and analysis of the proposed IP protection scheme. It covers the attack model, the actual protocol, and its requirements.

3.1 Attack Model and Assumptions

The proposed scheme considers a scenario where a typical IP vendor licenses q units of an IP to the system designer or end user for a negotiated fee. However, the IP user violates the term of the license and tries to deploy more than q units of the IP without paying for the extra units.

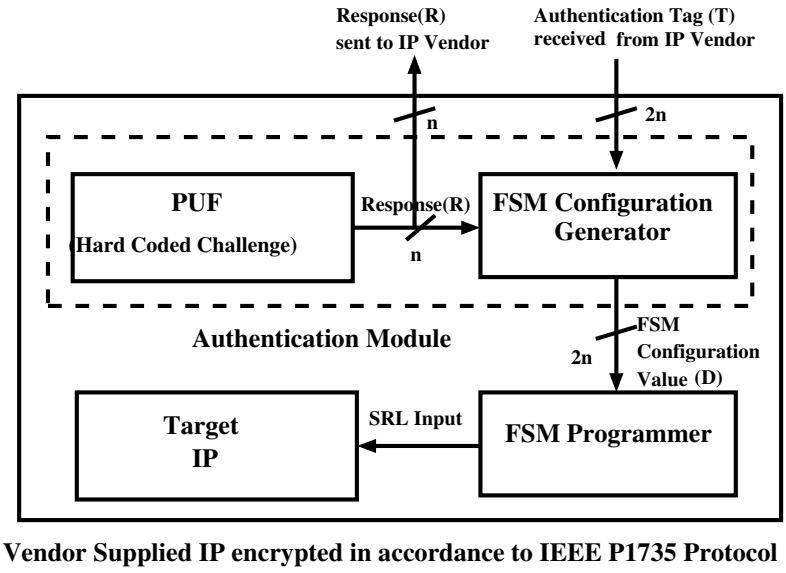

Fig. 3. The proposed IP protection scheme

To prevent such overuse of IP, an IP protection scheme is proposed in the following. The scheme is based on three basic assumptions:

- The IP vendor delivers a protected netlist of the concerned IP to the user. This protected netlist uses industry standard practices like netlist encryption and obfuscation as proposed in IEEE P1735. The adoption of IEEE P1735 ensures proper tool flow for simulation, synthesis, and implementation of the protected IP. The user can only observe the ports of the protected IP, without any visibility of the internal nets.
- IP vendor has access to a PUF with a stable response. The proposed scheme does not depend on a specific kind of PUF and the IP vendor may design the PUF in-house or license it through a third-party vendor. The PUF is expected to resist known attacks and present a stable response.
- The communication between the IP vendor and the IP user is done through a secure channel which provides both confidentiality and integrity. This assumption prevents *man in the middle attack* or *denial of service attack.*

3.2 IP Protection Scheme

This section focuses on the functionality of the developed IP violation protection scheme. As stated earlier, the proposed scheme targets the control unit of the IP. A design of control unit of any digital circuit is essentially based on FSM. The idea is to develop a strategy so that the execution of the control unit becomes dependent on the device fingerprint. Therefore, an IP bounded to a specific device will not work on another device.

The IP vendor designs the IP and tests it for functional correctness. Once the correctness is ensured, the IP vendor chooses a subset of LUTs which are used in the FSM logic of the IP. Let us assume that the *INIT* value of these selected LUTs on concatenation produces a vector D. Now, the IP vendor replaces the chosen LUTs with RLUTs having *INIT* value of zero. The IP vendor then

embeds the IP with other components of the scheme and send it to the end user (Fig. 3). The IP is sent after it is encrypted through IEEE P1735 protocol.

3.2.1 Activation Protocol. Block diagram of the proposed scheme for protecting IP vendors rights is shown in Fig. 3. Once the IP is delivered to the user and implemented on the device, an activation protocol must be followed to restore the functionality of the target IP. The IP activation process unrolls as follows:

(1) The user implements the obtained IP on a particular FPGA device. On initial power up, the IP enters the activation phase. The embedded PUF generates a **response R** which is sent **to the IP vendor**.
(2) The IP vendor computes the corresponding **authentication tag T** and sends it back **to the IP user.**
(3) The user inputs authentication token T to the device, **which produces correct D and activates the target IP.**

Once the FSM is reconfigured, the IP will give functionally correct value. Moreover, this IP cannot be used on other devices as the authentication tag T will be different for another device due to its different value of PUF response. The activation sequence is illustrated in Fig. 4. If an IP user wants to buy licenses for q devices, he needs to acquire q number of tag values from the IP vendor. As we are concentrating on SRAM-based FPGAs, whenever we reconfigure the FPGA, the FSM configuration vector D will be set to zero. However, the corresponding tag value T is already available to the IP user once he buys the licenses and hence no permanent online connection with IP vendor is necessary. The IP user will be supplied with the authentication tag when he buys the license and after that, no communication with the IP vendor is required.

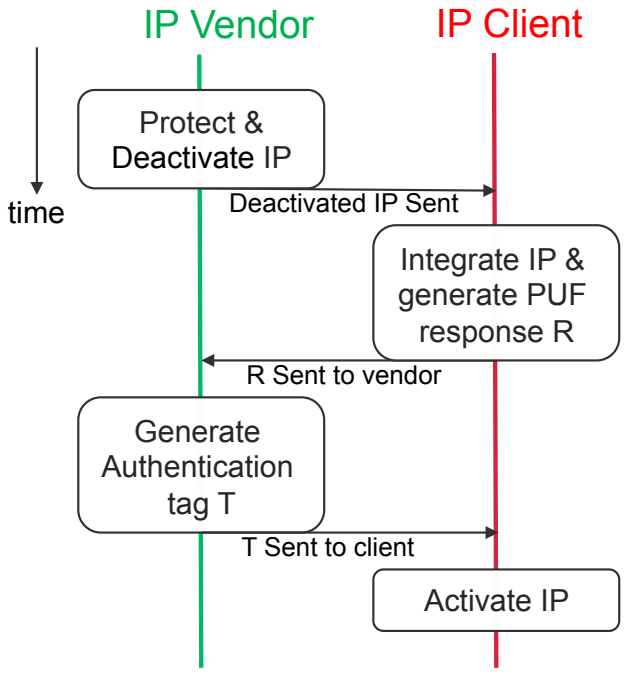

Fig. 4. IP activation protocol

3.2.2 Key Components of the Protection Scheme. The scheme generates the response token R and uses the corresponding authentication tag T in order to activate the IP. This can be done with four distinct components i.e. a PUF, an FSM configuration generator, an FSM Programmer and the target IP. The function of each block is as follows:

- **PUF**: The proposed scheme relies on an available PUF to generate a stable signature that is unique for every device. The challenge to PUF is hard-coded in the IP and does not require any intervention from the vendor after delivery. Upon initial power-up, the PUF generates a unique response R which is readable by the end user. The end user passes this response to the IP vendor for activating the IP. For a correct functionality of the proposed protocol, a stable PUF response is an absolute necessity. Hence, the IP vendor needs to ensure the reliability of the PUF response by mechanisms like error correcting codes or fuzzy extractors. The correctness of the proposed protocol requires high uniqueness and high reliability of the implemented PUF. The PUF need not be secret as long it provides a unique and reliable response. Usage of PUF is required as it will generate a unique device fingerprint. If somehow such unique device fingerprint is already available beforehand, the protocol does not require the deployment of PUF. FPGA vendors actually can generate such device fingerprint, but as our protocol is essentially built on PUF, we do not require the involvement of FPGA vendors. Additionally, the design of the PUF is part of the supplied IP and is protected through the IEEE P1735 protocol. Hence, though the design of the PUF is not secret, the IP user can not replace the PUF in the design. The architectural description along with the implementation details of the PUF design are discussed in Sec. 5.
- **FSM Configuration Generator**: Objective of this module is to create the FSM configuration data which later will be fed into the RLUTs (initialized to zero) by FSM programmer to restore the correct functionality of the target IP's FSM. From a security point of view, this module is important as, together with PUF, it constitutes the authentication module (Fig. 3). The module takes as inputs the PUF response R and the IP vendor supplied authentication tag T and generates the FSM configuration data D. It must be noted that as the device changes, the value of PUF response R and the authentication tag T changes whereas the FSM configuration data D remains same. For the correct realization of the proposed protocol, the FSM Configuration module must be bijective, i.e. for a given D, it must ensure that there are (almost) no two values of R which will generate D for the same value of T. To ensure this property along with secrecy of D, we employ a lightweight cipher $E_K(P)$ which operates on n-bit state and uses k-bit key K. In comparison, R is n-bit long as well, while T and D have $2n$ bits. FSM configuration module computes D in the following way:

$$\begin{aligned} F_C(R,T,K) = D &= D^H||D^L \\ &= E_K(T^H \oplus E_K(R))||E_K(T^L \oplus E_K(\pi(R))) \end{aligned} \tag{1}$$

  Here F_C denotes the FSM configuration generator function and $||$ is a concatenation. The authentication tag T and the FSM configuration data D are divided into two words (T^H, T^L and D^H, D^L, each of n bits). $\pi(R)$ is an involution without fixed points (used for security concerns). The vendor supplied authentication tag T is computed at the vendor side as

$$\begin{aligned} F_V(R,K,D) = T &= T^H||T^L \\ &= [E_K(R)||E_K(\pi(R))] \oplus [E_K^{-1}(D^H)||E_K^{-1}(D^L)] \end{aligned} \tag{2}$$

  Both encryption and decryption are used on vendor's end to compute T. However, given the authentication tag T, to compute D *only encryption module is needed on the client device.* Not observable to the IP client are the values of D and the vendor encryption key K. The

value of K is hard-coded in the FSM configuration generator module. As the entire netlist in encrypted in accordance with IEEE P1735 Protocol, the values of K and D are not observable by the IP client. As we are using the same key K at both on the IP vendor and IP user side, we can generate the FSM configuration data D correctly on the FPGA. Moreover, the key K is hard-coded in the netlist of the IP which is protected by the IEEE P1735 protocol. Hence, to get access to the encryption key K, the IP user needs to first break the cryptographic algorithms involved in the IEEE P1735 protocol. The attacks demonstrated in [26] will only give access to the bitstream, but it will not give access to the netlist. Hence, those attacks are not applicable in this scenario.
Once obtained, the authentication tag T can be stored in an ad-hoc RAM or in some on-chip memory. However, that is not necessary. The authentication tag T is known to the IP client. Therefore, rather than storing it on-chip, he can simply pass the tag T as an additional input to the circuit when it is switched on. Note, the scheme can be simplified at the cost of using a heavier (instead of lightweight) cipher. That is, F_C, F_V can have a simpler form if $E_K(P)$ is a $2n$-bit cipher. However, as we target both high security (higher than n bits) and low implementation cost, we choose to work with an n-bit cipher. The key management of the IP vendors' keys is an important area. In this paper, we assume individual keys for each IP vendor. More advanced key management methodology for FPGA applications can be found in [6].

- **FSM Programmer**: The objective of this module is to restore the functionality of the target IP's FSM. Once the FSM configuration generator completes the computation of configuration data D, this module becomes active and reconfigures the RLUTs of the FSM (which were initialized to zero), with the configuration data D serially.
- **Target IP**: This is the IP, that vendor wants to protect. Some parts of the FSM of this IP is designed using RLUT. The vendor misconfigures the RLUTs in the FSM of the IP before delivery. When presented with correct authentication tag T, the FSM will be configured to the correct value and will make the IP functional.

Our protocol needs to be executed only once on the first start-up. Once the FSM is reconfigured correctly, the protocol does not induce any timing overhead on the IP's timing performance. In contrast, in the existing approaches, some extra states are added in the FSM for the transition depending upon PUF response. Hence, in those cases the timing overhead becomes permanent.

3.3 Simulation Support

It is a common practice to functionally verify through simulations, external and internal IPs. The basic simulation support is provided by the IEEE P1735 protocol. The underlying CAD tools use a proper set of standard cryptographic protocols to ensure this simulation support while guaranteeing the security of the IP. Hence, it is important that such simulation support is guaranteed after incorporating our proposed protocol as well.

For enabling simulation support in the proposed protocol, a functional simulation model is supplied to the IP user. The simulation model has a PUF model built inside it, which will generate a response R_s with the same specification as the original PUF, implemented inside the supplied IP. The IP user will be also supplied with the corresponding authentication tag T_s which will unlock the IP for functional simulation. It must be noted that from the knowledge of T_s, it is not possible to get the knowledge of the actual authentication tag T as the actual PUF response R will not match with R_s which is only used for simulation.

3.4 Integration of Multiple IPs

The proposed scheme can be easily extended from one IP to multiple IP at a minimum overhead. The extended scheme reuses the same components. Only a different key per IP must be considered. Even if the IP comes from different vendors, the same components can be used, given that the key is not leaked from one vendor to another. A single PUF response R can be used to generate different pair (D_i, T_i), for i IPs.

3.5 Role of FPGA Vendor

One of the main features of the proposed IP protection scheme is that it involves only two parties, IP vendor and IP client (refer to Fig. 3). However, it must be noted that a malicious or even an honest and curious FPGA vendor has the complete access over the IP which is supposed to be protected against overproduction and overuse. The proposed IP protection scheme will only work with the following two assumptions:

- The IP vendor supplies the IP to the IP used with compliance to IEEE P1735 protocol (or its alternative).
- The FPGA vendor maintains the confidentiality and integrity of the developed IP. This implies that the FPGA vendor (and its associated EDA tool) neither modifies the design nor leaks it to the IP user. This assumption is realistic as, without this assumption, secure third party IP integration will not be feasible (section 4.4.3 of [2]).

We state our proposed scheme as two party because the FPGA vendor does not need to do anything additional to prevent IP overproduction apart from providing assurance of confidentiality and integrity of the IP which they are already providing for secure third party IP integration. As per [2], FPGA vendor already acts as TTP in the protocol architecture of IEEE P1735. Our proposed scheme takes advantage of this.

4 SECURITY ANALYSIS OF PROPOSED SCHEME

In this section, the security analysis of the proposed scheme is provided.

Brute Force: To break the scheme the user can guess the secret values of T or D. As they are $2n$-bits long each, with a brute force methodology, this requires a computational cost of around 2^{2n} operations, which for larger values of n (we will use $n = 64$), is impractical. Another strategy involves guessing the k-bit secret key K, and thus requires 2^k effort. As a result, the simple brute force will break the scheme in $\min(2^{2n}, 2^k)$ time. To balance the cost, we use $k = 2n$, which makes the overall cost of guessing 2^{2n}.

Security Analysis of the Scheme: Assume the user is malicious and at the cost of paying for 2^t licenses, he tries to produce $2^{t'}$ licenses, where $t' > t$. The 2^t paid licenses can be seen as the online complexity of the attack, i.e., the number of queries and corresponding responses with the vendor. That is, the user pays for 2^t different queries and responses of the type (R_i, T_i), where $i = 1, \ldots, 2^t$. In order to produce unpaid licenses, based on the previous data, the user has to: 1) make sure that for some not queried R_j the value of T_j coincides with some previous response T_i, where $j > 2^t, i \leq 2^t$, or 2) reduce the entropy of T_j, or 3) reduce the entropy of D.

The first scenario corresponds to a so-called collision attack, i.e. the problem of finding two values R_i, R_j such that $F_V(R_i, K, D) = F_V(R_j, K, D)$. From the definition of F_V, in particular from the fact that F_V is injective for fixed values of K and D, it follows that a collision between R_i and R_j will occur only when $R_i = R_j$. As the values of R are random (generated from the PUF), given 2^l values R, there will be around 2^{2l-n} collisions[2]. To put these number into perspective, assume

[2]From 2^l values one can construct around 2^{2l} pairs. A pair collides on n bits with probability 2^{-n}. Thus, given 2^l random values, there will be around 2^{2l-n} collisions.

$n = 64$ and $l = 40$, that is a user has 2^{40} devices. There will be $2^{2\cdot 40-64} = 2^{16}$ collisions, hence the user has to pay for around $2^{40} - 2^{16} \approx 2^{40}$ licenses and obtain only 2^{16} licenses for free. Thus, this is not a real threat to the vendor. In addition, note that 2^{40} is already a very high number compared to typical FPGA IP volumes.

The entropy of T can be reduced by reusing the previous queries (and responses), or by reducing the entropy of K. Recall that F_V is defined as

$$F_V(R, K, D) = E_K(R)||E_K(\pi(R)) \oplus E_K^{-1}(D^H)||E_K^{-1}(D^L) \tag{3}$$

Assume $E_K^{-1}(D^H) = A$ and $E_K^{-1}(D^L) = B$. Then,

$$T_i = T_i^H||T_i^L = E_K(R_i)||E_K(\pi(R_i)) \oplus A||B \tag{4}$$

If for some not queried R_j it holds $R_j = \pi(R_i)$ (where R_i has been queried before), then

$$T_j^H = E_K(R_j) \oplus A = E_K(\pi(R_i)) \oplus A = T_i^L \oplus B \oplus A \tag{5}$$

$$T_j^L = E_K(\pi(R_j)) \oplus B = E_K(R_i) \oplus B = T_i^H \oplus A \oplus B \tag{6}$$

Therefore, $T_j^H \oplus T_j^L = T_i^L \oplus T_i^H$, hence the user can reduce the entropy of the unknown $T_j^H||T_j^L$ from $2n$ bits to n bits. However, as mentioned earlier, the value of R_j cannot be chosen freely, thus this approach, if successful, only leads to create twice as many collisions as before (because π is involution without fixed points), thus the loss of the vendor is negligible. Further, let us focus on reducing the entropy of the secret key, K. To simplify the analysis, assume the user already knows the value of D, that is we give more freedom to the user than he actually has. In this case, the knowledge of 2^t queries and responses is equivalent to the knowledge of $2 \cdot 2^t$ pairs of plaintext-ciphertext of the form $(R_i, E_K(R_i) = T_i^H \oplus A), (\pi(R_i), E_K(\pi(R_i)) = T_i^L \oplus B)$. As the cipher $E_K(P)$ is secure, the provided data cannot help the user to reduce the entropy of the key and thus the user cannot obtain additional free licenses.

Finally, let us focus on the possibility of reducing the entropy of D without the knowledge of the secret key K, but with reusing the previous queries. The user queries the vendor the pair of values $(R_1, R_2) = (R_1, \pi(R_1))$ and receives the responses (T_1, T_2), where

$$\begin{aligned} T_1 &= T_1^H||T_1^L \\ &= E_K(R_1) \oplus A||E_K(\pi(R_1)) \oplus B \end{aligned} \tag{7}$$

and

$$\begin{aligned} T_2 &= T_2^H||T_2^L \\ &= E_K(R_2) \oplus A||E_K(\pi(R_2)) \oplus B \\ &= E_K(\pi(R_1)) \oplus A||E_K(R_1) \oplus B \\ &= (T_1^L \oplus B) \oplus A||(T_1^H \oplus A) \oplus B \end{aligned} \tag{8}$$

$T_2 = T_2^H||T_2^L = E_K(R_2) \oplus A||E_K(\pi(R_2)) \oplus B = E_K(\pi(R_1)) \oplus A||E_K(R_1) \oplus B = (T_1^L \oplus B) \oplus A||(T_1^H \oplus A) \oplus B$. Therefore $T_2^H \oplus T_1^L = A \oplus B$ and $T_2^L \oplus T_1^H = A \oplus B$. As T_1, T_2 are known to the user, he can actually find the xor difference between A and B, i.e. compute $A \oplus B$ at the price of two queries (two licenses). By definition $A \oplus B = E_K^{-1}(D^H) \oplus E_K^{-1}(D^L)$. To be able to reduce the entropy of D based on the knowledge of $A \oplus B$, the user has to break the cipher $E_K(P)$ which by definition is secure. Therefore, he cannot exploit this difference and thus cannot reduce the entropy of D.

The above security analysis, in fact, gives hints about the rationale behind our construction. Let us list explicitly some of the major points.

- We use a complete cipher rather than one-round (or round-reduced) keyed primitive because a weak construction may result in a complete recovery of the secret key K and can be used to produce valid licenses for free. For instance, if $E_K(P)$ is replaced with one-round cipher, then by querying (R_1, R_2) and receiving $(T_1 = E_K(R_1)||E_K(\pi(R_1) \oplus A||B, T_2 = E_K(R_2)||E_K(\pi(R_2)) \oplus A||B$, the user can compute $T_1^H \oplus T_2^H = E_K(R_1) \oplus E_K(R_2)$, and subsequently use differential cryptanalysis to recover the key K (recovering K from the one round differential is trivial, and is possible even when multiple rounds are used).
- We use a cipher with a key K of size $|K| > n$, to make the simple exhaustive search of the key infeasible (when $|K| = n = 64$, the search may still be feasible).
- We choose π to be involution *without fixed points* as otherwise, the number of collisions will increase. Indeed, if π is only an involution, then there exist fixed points, i.e. values of R such that $R = \pi(R)$. For all such values $T^H \oplus T^L = E_K(R) \oplus A \oplus E_K(\pi(R)) \oplus B = A \oplus B$. Thus a user can buy license for fixed point R_1, obtain corresponding licenses for all other fixed points R_i in 2^n (because he has reduced the entropy of T from $2n$ bits to n bits, as guessing n bit T^H allows to compute deterministically T^L as $T^L = A \oplus B \oplus T^H$).
- We define F_V as $F_V(R, K, D) = E_K(R)||E_K(\pi(R)) \oplus E_K^{-1}(D^H)||E_K^{-1}(D^L)$ rather than $F_V(R, K, D) = E_K(R)\ ||\ E_K(\pi(R)) \oplus D^H||D^L$ in order to prevent the user from reducing the entropy of D from $2n$ bits to n bits with the method present above.

Side Channel Attacks: The main target of a side-channel attack is the embedded encryption key K. In practice, the adversary can acquire side-channel traces by varying T^H, T^L and retrieve K. This motivates the use of side-channel countermeasures to protect the encryption core. For instance, a lightweight block cipher like SIMON only uses 36 slices when unprotected and 87 slices when protected with threshold implementation [31]. PUF can also be targeted with modeling attacks. However, with a hard-coded challenge, such attacks are prevented.

Fault Attacks & Reverse Engineering: In the proposed scheme, faults attacks on the encryption core are not possible as the ciphertext is not available to the adversary, which is a primary requirement for fault analysis. Reverse engineering attempts are limited by the available netlist encryption standard as per IEEE P1735. A fault attack on PUF by physical tampering will affect the response R and hence the tag T will not match. Additionally, an adversary can launch a safe error attack on the protocol where he induces stuck at faults on each bit of the encryption key sequentially. For example, let us assume that the adversary introduces a stuck at zero fault at the 0^{th} bit of the key. Now, if the IP gets successfully activated, the adversary can conclude that value of the 0^{th} bit of key is 0, otherwise the value of the 0^{th} bit of key is 1. Subsequently, the adversary has to now repeat this procedure for every key bit to retrieve the entire key. However, injecting stuck at fault at a particular bit position requires expensive and extremely precise fault injection tools. Additionally, in the proposed protocol, the encryption key is hard-coded, making the injection of stuck at fault more difficult.

Confidentiality of D: To find D, an adversary will need to isolate Target IP, remove connection with the authentication module and forge the value of D. However, this involves bypassing security and anti-reverse engineering measure brought in by IEEE P1735.

5 ARCHITECTURAL SPECIFICATION OF PROPOSED IP PROTECTION SCHEME

The key properties of components of the proposed scheme were described in Sec. 3.2.2. The design choices and architecture details of each of these components are described here. The main overhead of the whole scheme comes from these components is a fixed cost. Apart from these, minor overheads are incurred when modifying the FSM on the IP to make it compatible with the scheme (discussed in the next section).

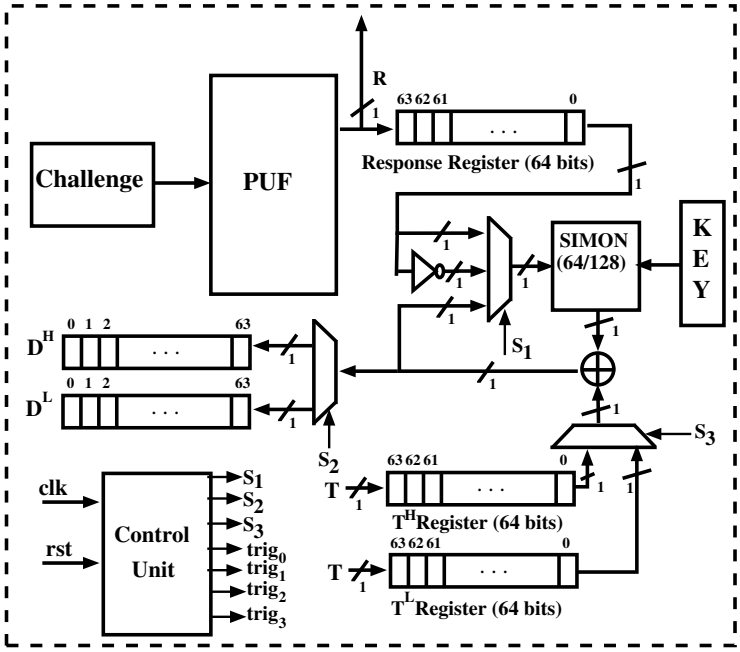

Fig. 5. Authentication module of the proposed IP protection scheme

5.1 Authentication Module

As shown in Fig. 3, the authentication module consists of PUF and FSM configuration generator. The architectural block diagram of the authentication module is shown in Fig. 5.

5.1.1 PUF. For correct functionality of the proposed protocol, existence of a reliable PUF is a necessary requirement. In this section, we will briefly discuss the working of two different PUF that we have considered.

(1) **Arbiter PUF (A-PUF) [33]** A-PUF is a well-studied architecture but suffers from response reliability issues. Multiple variants of A-PUF exist in the literature and among them, we have chosen 5-4 DAPUF [12] for our case study. 5-4 DAPUF is a delay based PUF consisting of 5 equivalent delay chains, followed by 20 arbiters and 5 XORs, which combines the output from the 5 chains and outputs a 4 bit response. It requires 64 bits challenge and as it produces 4 bits as a response for each challenge, we need to record the responses of 16 different challenge values to get the desired 64 bits response register value.
To ensure reliability, the PUF architecture needs to be augmented by error correcting schemes. In this work, we have used a recently published 5-4 DAPUF [12] which when coupled with *BCH* error correcting codes exhibit high reliability. The corresponding reliability variations with different temperatures are shown in Fig. 6. As illustrated in [12], the reliability value is computed for eight different FPGAs (Artix-7 XC7A100T)) at different temperatures (from -20^o to 80^o). Fig. 6 shows that the PUF reliability value is always between 97% to 99%, ensuring the effectiveness of the proposed protocol. It must be noted that using more sophisticated error correction technique like *temporal majority voting*, the error probability can be reduced significantly [15]. The overhead of the 5-4 DAPUF along with the overhead of the BCH encoder and decoder is shown in Table 2.

Issue with error correcting codes. As we have mentioned earlier, we have implemented the 5-4 DAPUF with BCH error correcting scheme which has the capability of handling 16 bit flips. For the completeness of the paper, we have listed the overhead of the error correcting schemes in Table 2. However, we can address the issue of the reliability in the protocol level as shown below:

(a) The IP client sends the PUF response R to the IP vendor
(b) The IP vendor computes the tag value T and also computes the helper data h. He sends the authentication tag T and stores the helper data h in his database.
(c) If due to reliability issue, the PUF response changes from R to $R1$, the IP client informs it to IP vendor and sends the noisy response $R1$ to him.
(d) The IP vendor applies the error correcting schemes on the received response $R1$ with helper data h and if he can retrieve the R, he sends a new tag value $T1$ corresponding to $R1$.

A malicious IP client can buy the IP for one device, and then implement it in another device. It then will send the response of the new device to the IP vendor and will pretend that this response is a noisy response of the first device. However, if the IP vendor can distinguish between the response of the new device from the noisy response of the old device using the helper data, the malicious IP client will fail. The advantage here is that the error correction mechanism is executed in the IP vendor side and does not have any overhead on the supplied IP. It must be noted that the response R is public and is known to the IP client. As long as the PUF response remains R, the IP client can use the same authentication tag T. Only when, the response changes from R to $R1$, the IP client needs to communicate with IP vendor to get $T1$. If the IP vendor has deployed a highly reliable PUF, we can expect that the PUF response will not change every time the system is powered up. Therefore, the IP client need not contact the designer for every power up of the system. The PUF reliability can also be enhanced by simple mechanism like temporal majority voting [24, 37] to reduce the probability of requiring new authentication tag. Another alternative will be to use a sophisticated PUF circuitry which will have high reliability without requiring any error correction codes. We discuss this next.

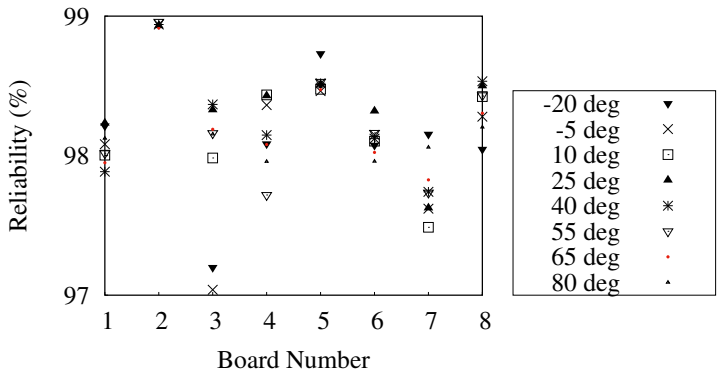

Fig. 6. Variation of reliability of 5-4 DAPUF at different temperature[12]

(2) **MA-PUF**: To improve the reliability of A-PUF, a multi arbiter PUF (MA-PUF) was proposed in [22]. A multi chain MA-PUF circuit is generally implemented as that improves the uniqueness, and reliability of PUF. This architecture was further improved in [39] by replacing the output D flip-flop with 4 D flip-flops. The proposed method could detect and encode

a metastable output to logical 0 or 1. This was shown to improve the overall randomness, uniqueness and stability of the response. Moreover, the overhead was significantly lower than the costly error correcting schemes. We implemented the MA-PUF with proposed 4-DFF arbiter with recoding technique owing to its lower cost in Artix-7 FPGA (XC7A100T). In this scenario, the challenge comes from 128-bit LFSR. The PUF response register is implemented on FPGA using *SRL32* primitives. Once the response register is filled with the PUF response values, it is sent to the output port so that it can be sent to the IP vendor. The overhead of this PUF is shown in Table 2.

It must be noted, that IP vendor is free to choose any existing PUF implementation as long it exhibits high reliability and produces a stable response. In our design, we have chosen the MA-PUF for its lightweightness. The lightweight PUF will reduce the fixed cost.

Table 2. PUF overhead

Circuit	Slice	LUT	Register
5-4 DAPUF	456	887	283
BCH Encoder	41	19	35
BCH Decoder	1236	685	1615
MA-PUF	100	354	712

5.1.2 FSM Configuration Generator. The PUF Response R is read serially by the FSM configuration generator. FSM configuration generator needs to compute four encryptions E_K to generate the FSM configuration data D. We use lightweight encryption algorithm *SIMON* [8] for its low area overhead when implemented on FPGA [5]. A serialized implementation of *SIMON* (128/128) requires only 13 and 36 slices on a Spartan 6 and 3 FPGA respectively. This was extended to a side channel protected version in [31] in 87 slices on Spartan-3. We followed the same design principle of [5] but for a different version of *SIMON* (64/128). Involution of R (i.e., $\pi(R)$) is implemented by simply xoring the response R with 1 (i.e., $\pi(R) = R \oplus 1$). The SIMON-64/128 implementation requires 18 slices on Artix-7 FPGA (XC7A100T). The slight increment in the slice requirement is due to a more complicated key schedule of SIMON-64/128. On the other hand, side-channel protected implementation of SIMON-64/128 requires under 40 slices on the 7-series FPGA. To compute one encryption, the unprotected SIMON core requires 1472 clock cycles and has a critical path of around 3 ns. Further practical validation is done with the unprotected version. The entire architecture of the authentication module is bit-serialized so that area overhead is small. The architecture of this module is shown in Fig. 5. Apart from computing the value of D ($D^H||D^L$), this module also generates control signals $trig_0, trig_1, trig_2, trig_3$ which are used by the FSM programmer module for reconfiguring RLUTs. This serialized architecture of course induces a high timing overhead. However, the overhead is induced only once and it can be considered as the boot-up delay. The total clock cycles consumed during the computation of D is 6020. Assuming a 100 MHz clock, this will take around 60.2 μs, but it is performed only one time. Once the target IP's FSM is reconfigured, this timing overhead will not affect the target IP's timing performance. The entire FSM configuration generator module requires 156 LUTs with 96 flip-flops and occupies 39 logic slices as shown in Table 3.

5.2 FSM Programmer

This function reads the FSM configuration generator outputs which are stored in two registers D^H and D^L. The architectural diagram of this module is shown in Fig. 7. This module reads the

Table 3. Overhead of FSM configuration generator

Circuit	Slice	LUT	Register	Clock cycles	Latency
FSM Configuration Generator	39	156	96	6020	60.2 μs (100 MHz clock freq.)

value from the D^H and D^L register and feeds this value to the RLUTs serially through *CDI* port. Additionally, signals $trig_0$, $trig_1$, $trig_2$, and $trig_3$ are provided to the RLUTs through *CE* port. FSM programmer is actually a simple connection between *CDI* port of RLUTs and D^H and D^L registers and hence has zero area overhead. It requires 64 clock cycles to reconfigure RLUT.

Thus, net fixed cost of the proposed IP protection scheme including PUF, FSM configuration generator, and FSM programmer is **139 slices.**

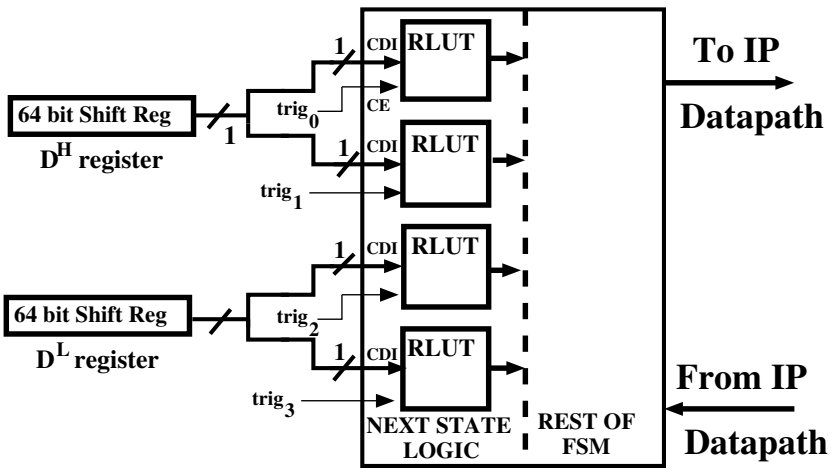

Fig. 7. FSM reconfiguration at the client side

6 EXPERIMENTAL VALIDATION

The proposed IP protection scheme is validated on MCNC'91 benchmark circuits [25]. Additionally, the scheme is also validated on a cryptographic core i.e *Advanced Encryption Standard (AES)* and a MIPS micro-processor [16]. A cryptographic core and micro-processor are chosen as they represent realistic examples of third-party IP. The results are validated on Artix XC7A100T FPGA from Xilinx.

As previously mentioned, the authentication module which consists of PUF and FSM configuration generator has area overhead of 139 slices (without any error correcting codes). The timing overhead of the design can be listed as below:

(1) The MA-PUF will require 64 cycles to produce 64-bit response. However, when the authentication tag T is received by the IP client, the response value is already available in the response register thus it does not contribute to the delay of FSM configuration module.
(2) Once the tag T value is passed to the circuit, the FSM configuration module takes 6020 clock cycles to generate the FSM configuration data.
(3) Subsequently, FSM programmer requires 64 cycles to program the FSM. This overhead is fixed for any IPs and can be considered as boot-up delay as once the FSM is programmed, this delay does not have any effect on the latency of the IP. Additionally, this boot-up delay

Table 4. Performance analysis of FSM programmer on MCNC'91 benchmark circuits, AES and MIPS Processor. (x) represent percentage overhead

IP	Original			With FSM Programmer			FSM Configuration Gen		MA-PUF		5-4 DAPUF	
	LUTs	FFs	Delay (ns)	LUTs	FFs	Delay (ns)	LUTs	FFs	LUTs	FFs	LUTs	FFs
dk16	44	27	1.488	44(0)	27(0)	1.601(7.5)	156	96	354	712	887	283
planet	102	7	2.274	102 (0)	7 (0)	2.627 (15.5)						
S1488	122	11	2.760	121 (−0.8)	11 (0)	2.787 (0.9)						
S1494	110	6	2.149	113 (2.7)	6 (0)	1.992 (−7.3)						
S298	242	218	2.468	246 (1.6)	218 (0)	2.612 (5.9)						
S510	35	6	1.971	35 (0)	6 (0)	1.958 (−0.6)						
S820	79	5	2.568	81 (2.5)	5 (0)	2.413 (−6)						
S832	66	5	2.236	67 (1.5)	5 (0)	2.754 (23)						
sand	110	6	2.409	111 (0.9)	6 (0)	2.583 (7.2)						
styr	101	5	2.404	102 (0.9)	5 (0)	2.666 (10.8)						
AES	1480	260	6.21	1482 (0.1)	260 (0)	6.42 (3.3)						
MIPS	1962	2039	17.5	1965 (0.1)	2039 (0)	17.8 (1.7)						

Table 5. Total Clock cycles Requirement for Programming the FSM

Clock Cycles for PUF Response Generation	Clock Cycles for FSM Configuration Generator	Clock Cycles for FSM Reconfiguration	Total Clock Cycles
64	6020	64	6084[3]

is constant and does not vary with the change of the IP. The timing requirement in terms of clock cycles is shown in Table 5.

Regardless of the nature of the target IP, the overhead of the FSM configuration generator (39 slices) will remain the same as the protocol does not depend upon the nature of the circuit. Rather, it depends on the configuration data stored in some of the LUTs. As long the length of the configuration data (D) does not change, the area overhead of FSM configuration generator will remain the same across different IPs.

In the following, we analyse the overhead incurred when we integrate the FSM programmer with the IPs. As shown in Table 4, the area overhead of the IP integrated with the FSM programmer is almost negligible, 4 LUT in the worst case, while −1 in the best case. In the latter case, the RLUT insertion had simplified further the FSM logic. Similarly, timing overhead can go in either direction. As RLUT in FSM modification contains precomputed D, it can sometimes accelerate the whole FSM. The maximum delay increase is 0.3 ns. The time required for computation of D by FSM configuration generator is only one time and does not affect the timing performance of the target IPs.

It must be noted that the overhead of integrating the FSM programmer with the IP does not reflect the total overhead of the design. Apart from the FSM programmer integrated IP, the total overhead of the proposed scheme includes the overhead of the PUF and FSM configuration generator module as shown in Table 4. However, the overhead of these modules does not change with the IP which is an interesting feature of the proposed scheme.

Previous work [40] reports 52% overhead in LUT, 55% in slices count and 17% of time penalty when implemented on MCNC'91 benchmark FSM.

Compared to that, we have an FSM configuration generator which has fixed cost of 39 slices irrespective of the IP choice which increases to 139 slices when we include the PUF. The FSM programmer when integrated with the IP has only 5.16% timing overhead with negligible area overhead. Moreover, after the initial boot-up delay, the timing performance of the IP remains same.

[3]This does not include the clock cycles required for PUF response generation as when the tag value is received, the response register already has value of the PUF response R

We will also like to point out that our protocol comes with a rigorous security proof which is absent in [40]. Once the IP is activated, it runs independently from PUF and FSM configuration generator. Thus, these modules do not degrade the timing performance of the design. An important contribution of the proposed IP protection scheme is that the area overhead of the FSM configuration generator does not vary with the change in IP. Hence, this protection scheme could be highly efficient for IPs with high resource requirement, where the percentage of additional area overhead due to this scheme will be negligible and the timing performance of the target IP will remain nearly the same.

7 CONCLUSIONS

This paper addresses the issue of IP overuse on FPGA and proposes a novel two party pay per device licensing scheme with the assumption that the FPGA vendor provides confidentiality and integrity of the developed IP. The basic building blocks of the proposed scheme are PUF for device binding, lightweight symmetric encryption for IP protection, and RLUT for IP activation. The scheme comes with a proof of security. It has various advantages over the existing IP protection schemes. First, being a two party scheme, it involves only the IP vendor and the IP user with the assumption that the FPGA vendors provide confidentiality and integrity of the developed IP. Additionally, the proposed scheme can be integrated with the IEEE P1735 standard. The scheme has fixed overhead for the authentication and configuration block of 139 slices on Artix-7. Nevertheless, a lightweight PUF with a stable response can be used to reduce the resource requirement further. The actual IP undergoes negligible overhead in terms of the LUT usage or timing performance. The scheme has been tested on MCNC'91 benchmark circuits, AES and MIPS core. Further research can explore lightweight PUF architectures in FPGA and RLUT alternatives in ASIC.

REFERENCES

[1] 2014. IEEE SA - 1735-2014 - IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP). https://standards.ieee.org/findstds/standard/1735-2014.html. (2014).

[2] 2015. IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP). *IEEE Std 1735-2014 (Incorporates IEEE Std 1735-2014/Cor 1-2015)* (Sept 2015), 1–90. DOI: http://dx.doi.org/10.1109/IEEESTD.2015.7274481

[3] Actel. 2003. Implementation of Security in Actel's ProASIC and ProASIC*PLUS* Flash-Based FPGAs. (2003). http://www.actel.com/documents/Flash_Security_AN.pdf

[4] Yousra Alkabani and Farinaz Koushanfar. Active Control and Digital Rights Management of Integrated Circuit IP Cores *(CASES '08)*. 227–234. DOI: http://dx.doi.org/10.1145/1450095.1450129

[5] Aydin Aysu, Ege Gulcan, and Patrick Schaumont. 2014. SIMON Says, Break the Area Records for Symmetric Key Block Ciphers on FPGAs. Cryptology ePrint Archive, Report 2014/237. (2014).

[6] Arnab Bag, Sikhar Patranabis, Debapriya Basu Roy, and Debdeep Mukhopadhyay. 2018. Cryptographically Secure Multi-Tenant Provisioning of FPGAs. *arXiv preprint arXiv:1802.04136, accepted in DAC-2018 as work in progress poster* (2018).

[7] Abdulrahman Hanoun Bassel Soudan, Wael Adi. IP Protection of FPGA Cores Through a Novel Public/Secret-Key Encryption Mechanism. In *Secure System Design and Trustable Computing*. 369–389.

[8] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. 2013. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404. (2013).

[9] L. Bossuet and B. Colombier. 2016. Comments on "A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-per-Device Licensing". *IEEE Transactions on Information Forensics and Security* 11, 11 (Nov 2016), 2624–2625. DOI: http://dx.doi.org/10.1109/TIFS.2016.2553454

[10] Marek Laban Oto Petura Lilian Bossuet Viktor Fischer Brice Colombier, Ugo Mureddu. 2017. Complete activation scheme for IP design protection. http://www.hostsymposium.org/host2017/hwdemo/HOST_2017_hwdemo_3.pdf. (2017). (Accessed on 03/11/2018).

[11] R. S. Chakraborty and S. Bhunia. 2009. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 10 (Oct 2009), 1493–1502. DOI: http://dx.doi.org/10.1109/TCAD.2009.2028166

[12] Urbi Chatterjee, Vidya Govindan, Rajat Sadhukhan, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, Debashis Mahata, and Mukesh Prabhu. 2017. Building PUF based Authentication and Key Exchange Protocol for IoT without Explicit CRPs in Verifier Database. Cryptology ePrint Archive, Report 2017/422. (2017). https://eprint.iacr.org/2017/422.

[13] Animesh Chhotaray, Adib Nahiyan, Thomas Shrimpton, Domenic Forte, and Mark Tehranipoor. 2017. Standardizing Bad Cryptographic Practice: A Teardown of the IEEE Standard for Protecting Electronic-design Intellectual Property. *IACR Cryptology ePrint Archive* 2017 (2017), 828.

[14] B. Colombier, L. Bossuet, V. Fischer, and D. Hély. 2017. Key Reconciliation Protocols for Error Correction of Silicon PUF Responses. *IEEE Transactions on Information Forensics and Security* 12, 8 (Aug 2017), 1988–2002. DOI: http://dx.doi.org/10.1109/TIFS.2017.2689726

[15] Jeroen Delvaux, Dawu Gu, Dries Schellekens, and Ingrid Verbauwhede. 2015. Helper data algorithms for PUF-based key generation: Overview and analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 6 (2015), 889–902.

[16] Dimitris, Lazaridis. 2012. Mipsr2000. (2012). http://opencores.com/project,mipsr2000.

[17] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. 2004. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *EUROCRYPT*. 523–540.

[18] Saar Drimer, Tim Güneysu, Markus G Kuhn, and Christof Paar. 2008. Protecting multiple cores in a single FPGA design. *Draft available at http://www. cl. cam. ac. uk/sd410/, written May* (2008).

[19] Jorge Guajardo, SandeepS. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems - CHES 2007*. 63–80. DOI: http://dx.doi.org/10.1007/978-3-540-74735-2_5

[20] J. Guajardo, S.S. Kumar, G.-J. Schrijen, and P. Tuyls. 2007. Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection. In *FPL 2007*. 189–195. DOI: http://dx.doi.org/10.1109/FPL.2007.4380646

[21] Ujjwal Guin, Qihang Shi, Domenic Forte, and Mark M. Tehranipoor. 2016. FORTIS: A Comprehensive Solution for Establishing Forward Trust for Protecting IPs and ICs. *ACM Trans. Des. Autom. Electron. Syst.* 21, 4, Article 63 (May 2016), 20 pages. DOI: http://dx.doi.org/10.1145/2893183

[22] Vladimir P Klybik and Alexander A Ivaniuk. 2015. Use of arbiter physical unclonable function to solve identification problem of digital devices. *Automatic Control and Computer Sciences* 49, 3 (2015), 139–147.

[23] R. Maes, D. Schellekens, and I. Verbauwhede. 2012. A Pay-per-Use Licensing Scheme for Hardware IP Cores in Recent SRAM-Based FPGAs. *IEEE TIFS* 7, 1 (Feb 2012), 98–108. DOI: http://dx.doi.org/10.1109/TIFS.2011.2169667

[24] Sanu K Mathew, Sudhir K Satpathy, Mark A Anders, Himanshu Kaul, Steven K Hsu, Amit Agarwal, Gregory K Chen, Rachael J Parker, Ram K Krishnamurthy, and Vivek De. 2014. 16.2 A 0.19 pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*. IEEE, 278–279.

[25] MCNC. 91. benchmarks. Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University (1991). (91).

[26] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs. In *CCS 2011*. 111–124. DOI: http://dx.doi.org/10.1145/2046707.2046722

[27] Shahed E. Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. 2016. A Survey on Chip to System Reverse Engineering. *J. Emerg. Technol. Comput. Syst.* 13, 1, Article 6 (April 2016), 34 pages. DOI: http://dx.doi.org/10.1145/2755563

[28] M. Rostami, F. Koushanfar, and R. Karri. 2014. A Primer on Hardware Security: Models, Methods, and Metrics. *Proc. IEEE* 102, 8 (Aug 2014), 1283–1295. DOI: http://dx.doi.org/10.1109/JPROC.2014.2335155

[29] Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Jean-Luc Danger, Debdeep Mukhopadhyay, Xuan Thuy Ngo, and Zakaria Najm. Reconfigurable LUT: A Double Edged Sword for Security-Critical Applications. In *SPACE 2015*. 248–268. DOI: http://dx.doi.org/10.1007/978-3-319-24126-5_15

[30] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *2008 Design, Automation and Test in Europe*. 1069–1074. DOI: http://dx.doi.org/10.1109/DATE.2008.4484823

[31] Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. 2015. Silent Simon: A Threshold Implementation under 100 Slices. Cryptology ePrint Archive, Report 2015/172. (2015).

[32] Eric Simpson and Patrick Schaumont. 2006. Offline Hardware/Software Authentication for Reconfigurable Platforms. In *Cryptographic Hardware and Embedded Systems - CHES 2006*. 311–323. DOI: http://dx.doi.org/10.1007/11894063_25

[33] G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *DAC 2007*. 9–14.

[34] Mark Mohammad Tehranipoor, Ujjwal Guin, and Domenic Forte. 2015. *Counterfeit Integrated Circuits: Detection and Avoidance*. Springer Publishing Company, Incorporated.

[35] Transparency Market Research. 2014. FPGA Market - Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2014-2020. (2014). https://www.transparencymarketresearch.com/field-programmable-gate-array.html

[36] Kan Xiao, Domenic Forte, Yier Jin, Ramesh Karri, Swarup Bhunia, and Mark Mohammad Tehranipoor. 2016. Hardware Trojans: Lessons Learned after One Decade of Research. *ACM Trans. Design Autom. Electr. Syst.* 22, 1 (2016), 6:1–6:23. DOI: http://dx.doi.org/10.1145/2906147

[37] Kan Xiao, Md Tauhidur Rahman, Domenic Forte, Yu Huang, Mei Su, and Mohammad Tehranipoor. 2014. Bit selection algorithm suitable for high-volume production of SRAM-PUF. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 101–106.

[38] Xilinx. 2014. Xilinx Plug-and-Play IP: Accelerating Productivity and Design Reuse. (2014). https://www.xilinx.com/publications/prod_mktg/ipcenter/Plug_and_Play_IP_backgrounder.pdf

[39] S. S. Zalivaka, A. V. Puchkov, V. P. Klybik, A. A. Ivaniuk, and C. H. Chang. 2016. Multi-valued Arbiters for quality enhancement of PUF responses on FPGA implementation. In *ASP-DAC*. 533–538. DOI: http://dx.doi.org/10.1109/ASPDAC.2016.7428066

[40] Jiliang Zhang, Yaping Lin, Yongqiang Lyu, and Gang Qu. 2015. A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing. *IEEE TIFS* (2015), 1137–1150. DOI: http://dx.doi.org/10.1109/TIFS.2015.2400413