IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 8, 2019, accepted April 29, 2019, date of publication May 16, 2019, date of current version May 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917259

A Lightweight LFSR-Based Strong Physical Unclonable Function Design on FPGA

SHEN HOU[1,2], YANG GUO[1], AND SHAOQING LI[1]
[1]School of Computer, National University of Defense Technology, Changsha 410073, China
[2]Department of Basic Courses, Information Engineering University, Luoyang 471003, China

Corresponding author: Shen Hou (houshen@outlook.my)

This work was supported by the National Natural Science Foundation of China under Grant 61832018.

ABSTRACT Physical unclonable function (PUF), a reliable physical security primitive, can be implemented in FPGAs and ASICs. Strong PUF is an important PUF classification that provides a large ''Challenge-Response'' pairs (CRP) space for device authentication. However, most of the traditional strong PUF designs represented by the arbiter PUF are difficult to implement on FPGA. We propose a new lightweight strong PUF design that can dynamically reconfigure while maintaining high entropy and large CRP space. We implement the PUF on a 28-nm FPGA. The experimental results show that the uniformity of the PUF is 49.8%, the uniqueness is 49.9%, which is close to the ideal value, and the hardware overhead is very small. This design is easy to implement and suitable for device authentication on FPGA.

INDEX TERMS Hardware security, physical unclonable functions (PUFs), linear feedback shift register, FPGA, lightweight.

I. INTRODUCTION

Field-programmable gate array (FPGAs) have a wide range of applications in embedded system development due to their flexible configurability and relatively low design cost relative to ASICs. In recent years, big data, artificial intelligence, and cloud technologies have developed rapidly [1]. In these application scenarios, CPUs are highly flexible but lack computing power. Accelerators are computationally efficient but not flexible enough. In order to strike a balance between flexibility and efficiency, hardware accelerators plus CPU architectures are becoming more popular. As a hardware reconfigurable architecture, FPGA has powerful computing power and sufficient flexibility. As a kind of accelerator that has been paid more and more attention in the field of deep learning, FPGA has become a new research and application hotspot. Microsoft uses FPGAs to replace traditional CPUs in data centers [2]. Tencent Cloud directly provides FPGA cloud servers, reducing the high cost of developers to purchase equipment [3]. The iPhone has a FPGA chips inside, made by Lattice Semiconductor, according to Chipworks [4].

Widely used of FPGAs brings new security challenges, such as overbuilding, tampering, cloning and reverse engineering [5]. The bitstream that the user uses to configure the FPGA is read and decrypted by these attacks, resulting in leakage of circuit structure, sensitive information, and configuration parameters. FPGA vendors offer designers a spectrum of security solutions to protect intellectual property (IP) and sensitive data, such as encrypting bitstreams, protecting key memories, and bitstreams authenticating. The keys for encryption and authentication algorithms are typically stored in NVM (Non-volatile Memory, typically using mature memory technologies such as EEPROM, Flash, battery-backed SRAM, and fuses, etc.). These methods do not guarantee the security of keys and sensitive information. Keys are easily obtained from NVM through physical invasive attacks. At the same time, for some FPGA applications, the hardware resources are limited, and integrating security modules in them will increase hardware overhead, which may cause some problems. Therefore, the development of new lightweight hardware security primitives as a root of trust (RoT), providing security services such as key generation and authentication for FPGA applications has become a very attractive research field.

In the manufacturing process of digital circuits, due to some uncontrollable reasons, the parameters such as the size, threshold voltage, and gate oxide thickness of each device cannot be exactly the same, and there will be slight and random deviations, that is, process deviations. These deviations do not affect the functionality of the device, nor do they

The associate editor coordinating the review of this manuscript and approving it for publication was Aniello Castiglione.

affect the correctness of the circuit. However, these random deviations can be extracted by special design methods to make a unique ''fingerprint'' of the circuit, so as to accurately identify each circuit and prevent the circuit and chip from being over-manufactured or tampered. This physical system is called the physical unclonable function (PUF). As an emerging hardware security primitive, PUF is very sensitive to physical tampering and has inherent physical disorder. It has unparalleled advantages and broad application prospects in the fields of cryptography and hardware security.

PUF only generates a corresponding response when a challenge is given. This working mechanism is called the ''Challenge-Response'' mechanism, and all the ''Challenge-Response'' pairs of one PUF is called its CRP space. It is also because of this ''Challenge-Response'' security mechanism that keys and private information can be generated in real time without being stored in easy accessing local memories, further reducing the chance of the key being exposed to the adversaries.

According to the relationship between the number of CRPs and the size of physical entities, PUF can be divided into two types: Weak PUF and Strong PUF [6], [7]. PUFs with limited number of CRPs, known as Weak PUFs, are commonly used for key generation in cryptographic functions. Alternatively, Strong PUFs have exponential number of CRPs and are suitable for authentication [8]. Both types of PUF need high uniqueness and reliability to secure their properties. Meanwhile, a well-designed PUF should has small hardware overhead and be easily implemented. Linear feedback shift register (LFSR), as a circuit that can generate pseudo random number sequence, is widely used in key generation and communication fields. It is a high performance, simple and configurable sequential circuit which can provide an excellent random number sequence output with low hardware overhead.

In this paper, we employed LFSR to a FPGA-based Weak PUF design to make a lightweight and easily implemented ''Strong'' PUF. We introduce related works on PUFs and LFSR at first. Then we describe our design and implementation of a lightweight LFSR-based PUF on 28nm FPGA devices, and followed by the experimental data and conclusion.

II. RELATED RESEARCHES

A. PHYSICAL UNCLONABLE FUNCTION

In 2002, Pappu proposed the concept of physical unclonable function [9], and proposed a PUF based on optical system. Thereafter, various kinds of PUFs have been proposed. According to the implementation of the PUF, these PUFs can be classified into three types: non-electronic PUF, analog PUF, and digital PUF.

Non-electronic PUF mainly includes optical PUF [9], paper PUF [10], CD PUF [11], etc. This type of PUF utilizes the uniqueness of the different media under the influence of random factors in the manufacturing process to perform anti-counterfeiting verification on specific media.

Analog PUF mainly includes voltage threshold (VT) PUF [12], LC PUF [13], coating PUF [14], etc. The entities of these PUFs can be integrated into IC devices to measure the amount of electrons that can be used as a response signal in an analog manner, then convert it to digital form to produce the final PUF response.

The digital PUF directly performs response measurements in a digital manner without the need for analog to digital conversion. Compared with the first two types of PUF, this type of PUF is designed to conform to the standard IC design and manufacturing process and can be integrated into IC devices as a whole, so it has the widest research and application prospects. The digital PUF can be further divided into memory-based PUF and delay-based PUF.

There are many types of memory-based PUFs. Fig. 1 is a simple memory-based PUF structure. The memory-based PUF is usually Weak PUF, and it is hardly implemented on FPGA device because many modern FPGA chips initialize memory units at power-up.

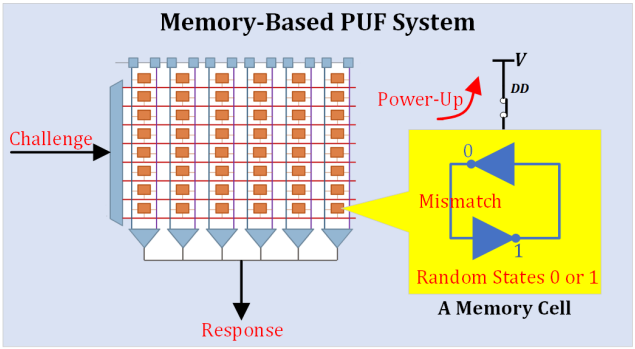

FIGURE 1. A simple memory-based PUF.

Delay-based PUF mainly includes RO (ring oscillator) PUF [15], [16], arbiter-based PUF [17]. Suh and Devadas proposed RO PUF in [16]. RO is a simple circuit that can oscillate at a specific frequency, which becomes unpredictable due to process variations in the manufacturing process. This type of PUF can generate a logic 0 or a logic 1 by comparing the oscillation frequencies of two identical RO circuits. Fig. 2 is a simple RO-based PUF. The design and layout of the two ROs must be identical to ensure that the difference in oscillation frequency depends only on process variations.

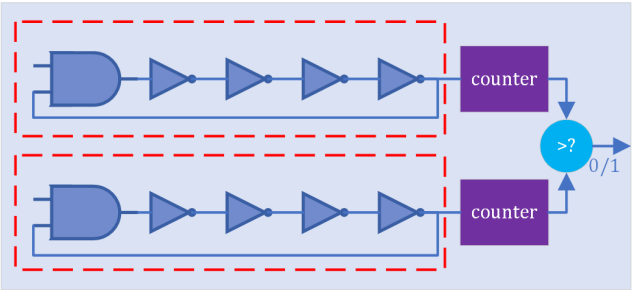

FIGURE 2. The structure of RO PUF.

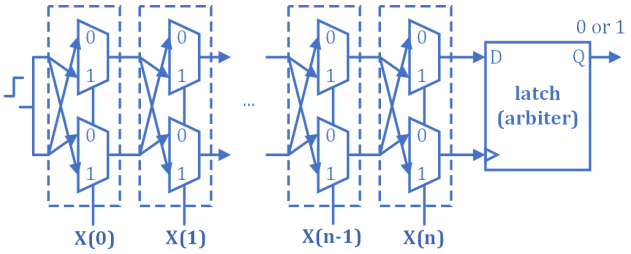

FIGURE 3. Architecture of arbiter PUF.

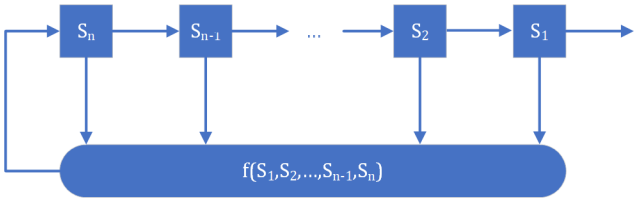

FIGURE 4. Basic structure of LFSR.

Lim *et al.* proposed arbiter-based PUF [17]. A simple arbiter PUF contains a multi-level multiplexer chain, the basic structure of which is shown in Fig. 3. The same input is connected to the upper and lower paths, and the way each path is connected is determined by the control signals of the two multiplexers in each stage. Under the same challenge, although the layout of the two paths is completely identical, the process deviation results in a slight difference in the time delay of two paths. The latch behind the chain acts as an arbiter to announce which path is faster, then outputting logic 0 or logic 1.

B. REALIZING STRONG PUF FROM WEAK PUF

As a classic digital Strong PUF design, arbiter PUF has all the features of a Strong PUF, but its shortcomings are equally prominent. First, the area overhead is in general small, and increase logarithmically. But it is difficult to implement in FPGA. Second, the delay difference depends on the inverter chain and line delay. The poor stability and reliability make it difficult to ensure that the whole PUF circuit is not affected by voltage and temperature fluctuations. Third, from the perspective of some realized arbiter PUF, although the CRPs space is large enough, its response uniqueness is not so good, far lower than the ideal value. Fourth, the path delay is composed by a linear superposition of multiple inverter delays on the path, which makes it easily to be modeled and vulnerable to machine learning attacks.

In general, the stability and response uniqueness of Weak PUF is better than the existing Strong PUF design [7], [18]. Realizing Strong PUF with stable and mature Weak PUF becomes a feasible design method. This method usually uses reliable Weak PUF as an entropy source in the front-end and a structure similar to a random number generator in the back-end. This structure can provide logical obfuscation to increase randomness and expand CRPs space. Notably, the entire circuit must be sophisticated designed to maintain unclonability. Several types of obfuscation logic have been proposed, such as AES [19], neural network [20], etc. However, the hardware overhead of these logics is still too large. As a mostly used pseudo-random number generation circuit, LFSR has good randomness and simple logic structure, which is suitable for designing low-overhead Strong PUF.

C. THE CONCEPT OF LFSR

LFSR is usually composed of two parts: a shift register and a feedback function. The shift register is a sequence of bits, and if it is n bits in length, it is called an n-bit shift register. The basic structure of the feedback shift register is shown in Fig. 4. $f(S_1, S_2, \ldots, Sn_{-1}, S_n)$ is a feedback function.

The initial value of the LFSR is called seed, and we can change the output sequence of the register by changing the value of the seed. The output value of the register is completely determined by its current state (or previous state). A good feedback function selection which is called the maximum length polynomial can generate an output sequence with good randomness and a long repetition period. The rules for selecting feedback polynomial which is given in [21], [22] are as follows:

1) The "1" in the polynomial does not correspond to a tap it corresponds to the input to the first bit.
2) The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and output tap respectively.
3) The LFSR will only be maximum-length if the number of taps is even; just 2 or 4 taps can suffice even for extremely long sequences.
4) The set of taps taken all together, not pairwise (i.e. as pairs of elements) must be relatively prime. In other words, there must be no common divisor to all taps.

III. PROPOSED PUF DESIGN

This section gives a Strong PUF designing method using a relatively mature and reliable Weak PUF. We first discuss the Weak PUF design that provides unclonability. In order to simulate on the FPGA, we select a low-overhead Weak PUF for FPGA and improve it so that it can be implemented on the latest 28nm FPGA chip. Then, the feasibility of LFSR as obfuscation logic is discussed. Finally, the combination of the two parts and the realization of some peripheral functions are confirmed.

A. ARCHITECTURE

The LFSR-based Strong PUF (L-PUF) structure proposed in this paper is shown in Fig. 5. The output response of the front-end Weak PUF provides a unique feedback polynomial for the LFSR, making the LFSR structure of each device different. By providing the same seed as the input challenge for different LFSRs, the L-PUF will generate device's unique output as response after running a fixed number of cycles. Next, we will discuss the specific implementation methods

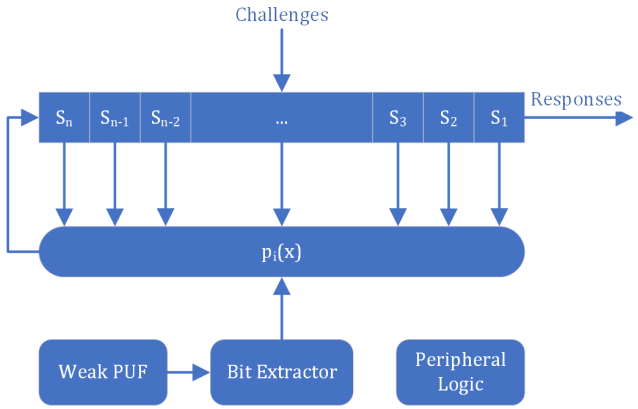

FIGURE 5. The architecture of LFSR-based strong PUF.

of the Weak PUF and LFSR parts, and how to combine the two parts.

B. THE WEAK PUF DESIGN ON FPGA

The most classic Weak PUF is the memory-based PUF as SRAM PUF [23]. However, since the design verification of this paper is performed on the FPGA platform, most FPGA development board initializes the memory on the SoC at startup, so the on-chip memory cannot be used as Weak PUF. Recently, some PUF designs dedicated to FPGA chips have been proposed based on the study of FPGA underlying architecture. Anderson claimed to implement the PUF structure on the FPGA for the first time [24]. Anderson's design refers to the basic idea of the delay-based PUF, and takes advantage of the internal structure of basic cells of the Xilinx FPGA chip called SLICE. It uses the LUTs that can be configured as a shift register and the cascaded multiplexers in the carry logic to build a competing two-way delay path, and uses a flip-flop to output a logic 0 or logic 1 response. These components all can be put in one SLICE. So, the area overhead is very small, and a minimum of 2 SLICEs can implement a 1-bit PUF response.

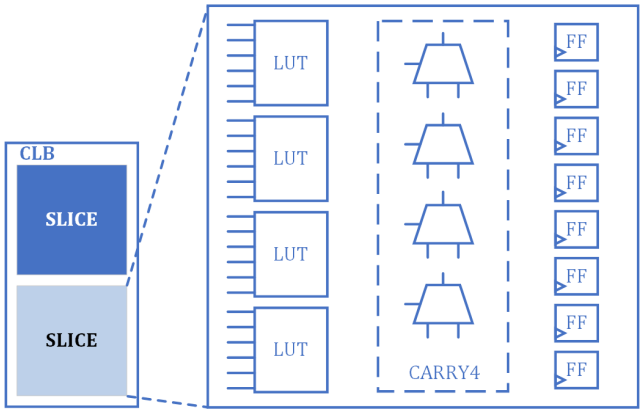

FIGURE 6. The underlying architecture of Xilinx 7 series FPGA.

Fig. 6 depicts the underlying architecture of Xilinx latest 7 series 28nm FPGA chip. The basic constituent logic unit is called CLB (configurable logic block), arranged in a two-dimensional array on the chip, and can be connected through a programmable interconnect matrix. A CLB mainly consists of two SLICEs. Each SLICE mainly consists of four 6-input LUTs, one carry chain (CARRY4 in the dotted line), and eight storage elements (flip-flops). The 6-input LUT can implement any 6-variable logic function, while CARRY4 is used to implement fast arithmetic operations. Each LUT is connected to a corresponding multiplexer in the carryer, and the output of the LUT provides a selection signal for the multiplexer. Four multiplexers are cascaded, the output of the lower multiplexer is one of the inputs of the upper multiplexer, and the other input is obtained outside of SLICE. There are two types of SLICE: the LUT in SLICEM can be configured as shift register logic (SRL) or general logic as needed, while the LUT in SLICEL can only be configured as general logic. These two SLICEs are usually arranged in columns by column.

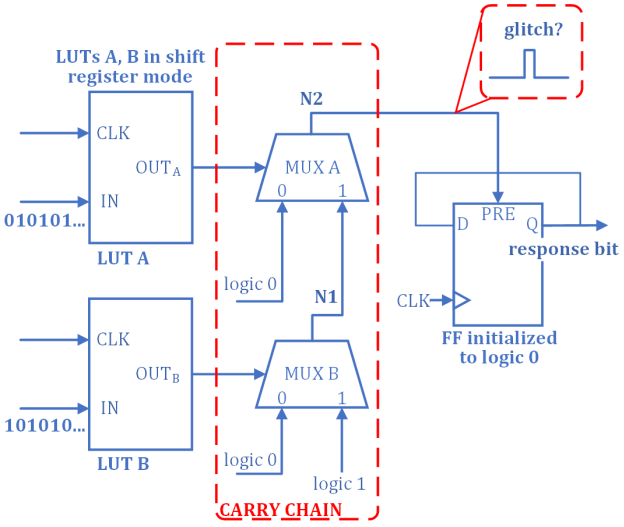

FIGURE 7. A bit Weak PUF response logic in Anderson's design.

In Anderson's design, a bit PUF response logic is shown in Fig. 7. Two LUTs, A and B, in SLICEM are configured in SRL mode. The contents of the register are initialized as:

1) LUT A: 0101010101010101 (0x5555)
2) LUT B: 1010101010101010 (0xAAAA)

The two SRLs are controlled by the same clock. Therefore, the path delay from the input of LUT A to N2 should theoretically be equal to the path delay from the input of LUT B to N1. However, due to process derivation, the two path delays will not be exactly the same. Assuming that the lower path B is faster than upper path A, then on the rising edge of the second cycle, the N1 value first changes to logic 0, while N2 continues to be logic 0. But if path A path is faster, then on the rising edge of the second cycle, N2 is already equal to N1 before N1 turns to logic 0, which makes N2 value having a glitch. Connect N2 to the PRE port of an asynchronous preset flip-flop, initialize the flip-flop state to 0, and output

Q feedback to input D. If N2 has a glitch, the flip-flop state will be set to 1. Otherwise, its value is always 0.

However, due to the earlier completion of the design, its implementation and analysis were performed on Xilinx's previous generation FPGA chip Virtex-5. Xilinx's new generation 7 series FPGA chips (including Virtex-7, Kintex-7, Artix-7, and the Zynq-7000 series used in this article) are different from the 5 series and other previous products. In the original design, when LUT A and LUT B were configured as shift registers, the input ports of MUX A and MUX B could be directly connected to independent logic 0. Therefore, the path formed by LUT A and MUX A can be placed in the same SLICE. However, for the 7 series chips, the input ports of MUX A and MUX B are controlled by the multiplexer in the blue box in Fig. 8. When LUT A and LUT B are configured as SRL, the two inputs of the multiplexer, O5 and AX, cannot always be logic 0 anyway, so the design cannot be implemented on new FPGA chips.

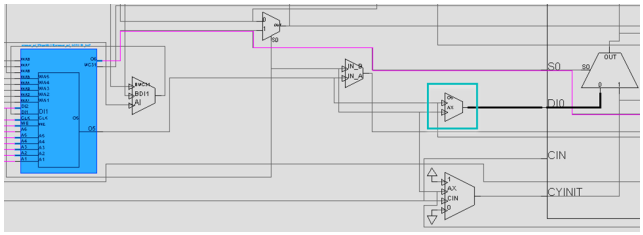

FIGURE 8. Anderson's design cannot be implemented on 7 series FPGA.

In order to solve this problem, we improve the original design. A path consists of a LUT configured as an SRL in a SLICEM, and a corresponding position multiplexer in the adjacent SLICEL. Any one of the 8 flip-flops in SLICEL is used as the output response bit. At the same time, we need to adjust the relative distance between the two paths A and B to improve the randomness of the PUF (short distance will cause the glitch too narrow to be filtered out). The experimental results show that when path A and B are separated by 5 or 6 LUTs, the results meet our requirements. Therefore, the final design is shown in Fig. 9. Four adjacent SLICEs can provide 1-bit PUF response, and the hardware overhead is still small. Our designs can be synthesized, placed and routed automatically without manual intervention.

For LFSR, if the initial value of the register chain is the same, different feedback functions, different tap numbers and positions will generate different shift output sequences. This is the basic idea of our design. We place the tap position at the output of each register and perform a logical AND operation with the response bit of the Weak PUF output to implement different tap position arrangement. For the front-end Weak PUF whose output response is m bits, its n-th output response is puf[n], where $0 < n \leqq m$, if an L-PUF instance's puf[n] $= 0$, the corresponding XOR logic in the LFSR does not work, that is, this position has no tap. Otherwise, if the puf[n] $= 1$, the position has a tap (logic in the dotted box 1 in Fig. 10). If the uniqueness of the Weak PUF is ideal

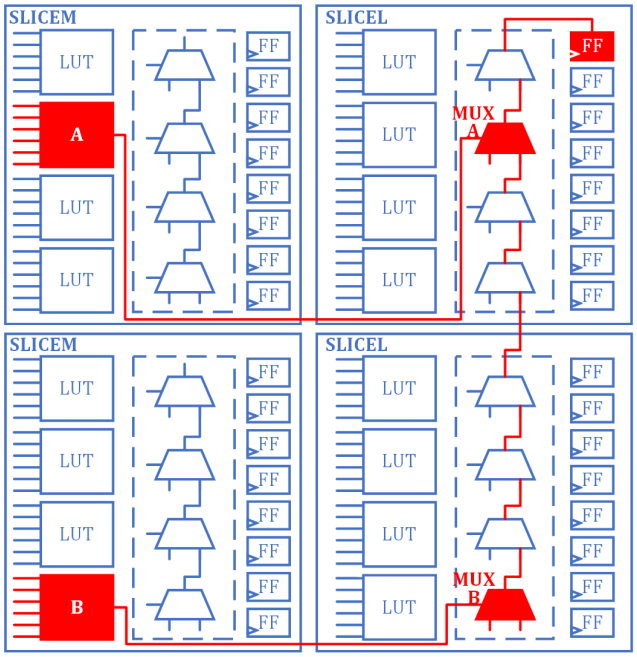

FIGURE 9. 1-bit Weak PUF design on 28nm Xilinx FPGA.

enough, we use different seeds as a challenge to ensure the unclonability (identical circuit) and uniqueness (the response generated by the same challenge is different) of the entire L-PUF.

C. FURTHER DISCUSSION

According to the LFSR feedback polynomial selection rule mentioned above, no matter how large the register chain is, the number of taps reaches 4 or more to meet the randomness requirement. So, the Weak PUF and the LFSR register chain do not have to correspond one-to-one, and the scale of the Weak PUF can be further reduced (the number of 1 in the output response is not less than 4). The randomness requirement for the Weak PUF response can be reduced, and it is not necessary to pursue the ideal value too much. The scale of the Weak PUF needs to be determined according to the specific application environment, and the number of output response bits can be roughly referred to (1).

$$\text{Response bits} \times \text{Uniformity} \geq 4 \tag{1}$$

Uniformity is a parameter that characterizes the randomness of PUF output response. For a 16-bit Strong PUF application, assuming that the uniformity of the front-end Weak PUF is 50%, the response bits of the front-end Weak PUF may be not less than 8. The XOR logic can be evenly distributed to the register chain of the LFSR as in Fig. 11, so that the hardware overhead of L-PUF can be further reduced. We call this improved structure the RL-PUF.

D. PERIPHERAL CIRCUIT

However, there are two problems we need to consider. First, the value of the LFSR in any state cannot be all 0s, otherwise

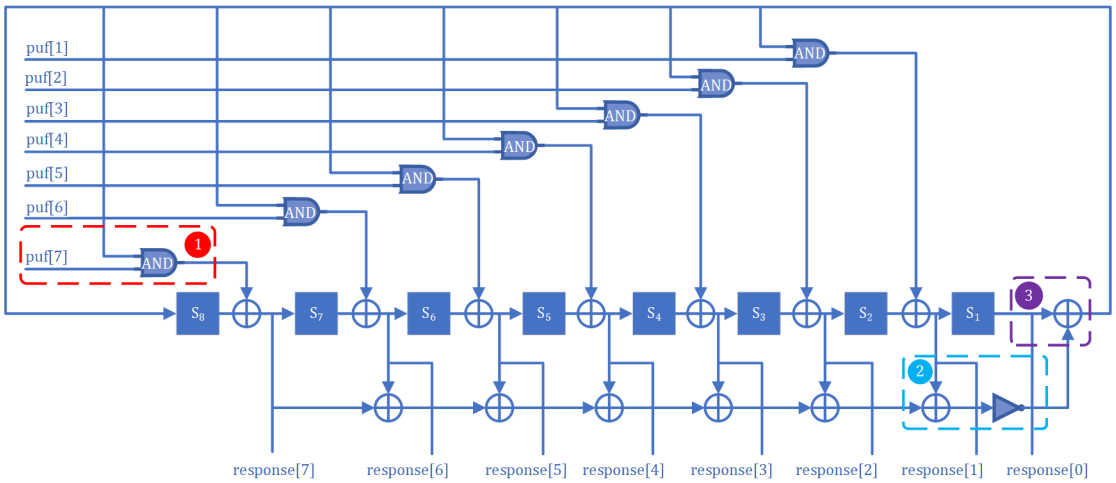

FIGURE 10. 8-bit L-PUF circuit.

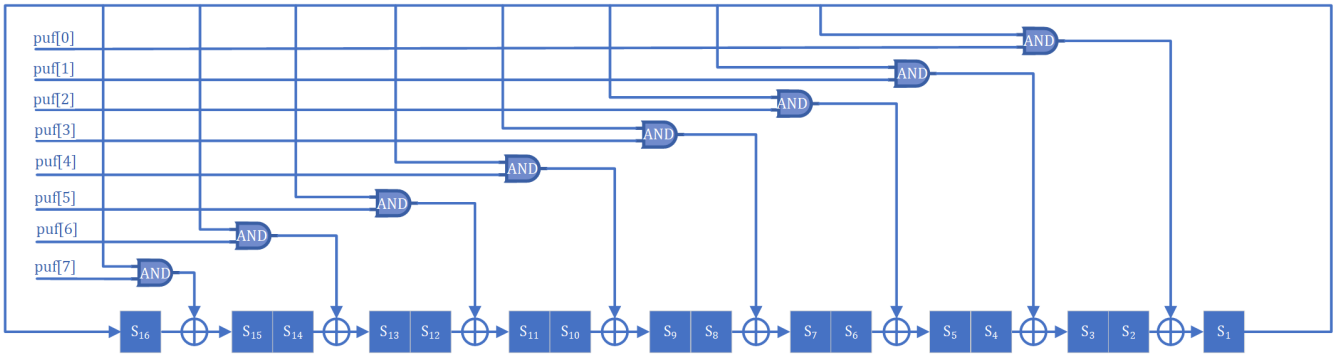

FIGURE 11. 16-bit RL-PUF circuit.

the LFSR will not be able to jump to other states. Second, for this type LFSR we used, when the feedback output is 0, the system will become a normal shifter. Therefore, the leading zero of the seed sequences will cause the output of all L-PUFs to be the same, thus losing the PUF function. For the first problem, we can detect the state of the LFSR in real time through the peripheral logic in the dashed boxes 2, 3 in Fig. 10. If the all-zero state is found, the feedback value is changed and the all-zero state is jumped out. For the second problem, there are two solutions: Option A, adding the leading 0 detection and counting control logic, if the challenge loaded to the L-PUF has n bits leading 0, the L-PUF is controlled to output after n clock cycles. Option B, only the counting control logic is added. For an m-bit L-PUF, it is controlled to output after m-1 clock cycles. For most IoT devices, hardware overhead has a higher priority than latency overhead, and option B is clearly a better choice.

IV. IMPLEMENTATION AND RESULT EVALUATION

A. FPGA IMPLEMENTATION

In this paper, we implement L-PUF and RL-PUF on an Alinx development board (Zynq-7000 XC7Z020 FPGA). Xilinx offers a complete and powerful physical constraint method for multi-level physical location constraints on designs done with VHDL or Verilog. We can precisely place the L-PUF structure by using macros and setting labels. Considering that the process deviation between devices is greater than that between different regions on the same device, if our design is good enough in different regions of a chip, it is certainly has no problem on different devices. Therefore, we implement several PUF instances on the same board to evaluate uniqueness and reliability.

We divide the FPGA side of the Zynq-7000 chip into 16 regions, implementing a 64-bit L-PUF and a 64-bit RL-PUF on each region. We use the RLOC_RANGE attribute in the Vivado 2017.1 development environment to constraint all the implementations. Then, each PUF is packaged as an IP core and connected on the AXI bus inside the Zynq-7000 chip so that we can write the challenge and read the response by the UART interface on the ARM side of the chip. The experimental data is recording and processing on a PC using Python.

B. HARDWARE OVERHEAD

A 64-bit L-PUF design uses 210 of the 53200 LUTs as logic on the Zynq-7000 XC7Z020 FPGA (0.4%), 128 of

TABLE 1. Hardware utilization.

PUF type	**64-bit L-PUF**		
Logic function	64-bit Weak PUF	LFSR and other logic	Total
SLICE LUTs	130	80	210
LUTs as Memory	128	0	128
LUTs as Logic	2	80	82
SLICE	256	60	316
LUT FF Pairs	0	33	33
PUF type	**64-bit RL-PUF**		
Logic function	32-bit Weak PUF	LFSR and other logic	Total
SLICE LUTs	65	64	129
LUTs as Memory	64	0	64
LUTs as Logic	1	64	65
SLICE	128	52	180
LUT FF Pairs	0	23	23

the 17400 LUTs as memory (0.8%), and 316 of the 13,300 SLICEs (2.4%). The three results for a 64-bit RL-PUF are 129 (0.24%), 64 (0.37%), and 180 (1.4%). The experimental results show that the Strong PUF design in this paper has a small hardware overhead and is very lightweight. Detailed data are shown in Table 1.

C. PERFORMANCE ANALYSIS

1) UNIFORMITY

Uniformity characterizes the distribution of 0 and 1 in the PUF response. As the main reference for PUF performance, the value of uniformity reflects the randomness of the PUF response. The better the randomness, the higher the security of the PUF. The ideal value for uniformity is 50%, meaning that the probability of 0 and 1 in the PUF response should be identical.

We separately calculate the uniformity of the front-end Weak PUF and the overall L-PUF output response. A total of 1024 bits responses are generated by 16 64-bit PUFs. When the relative distance between the two paths is 5 LUTs, the number of 1s is 396, and the uniformity is 38.7%, which is close to the ideal value. Moreover, since our design does not require high requirements for Weak PUF uniformity, this value already ensures a good feedback function choice for the LFSR.

The same four 64-bit challenges are loaded to the 16 L-PUFs, and each L-PUF records the output of the first 100 clock cycles. According to our design, all responses after the 63rd clock cycles are taken, and we get $4 \times 64 \times 16 \times 38 = 155648$ bits responses. The number of 1s is 77488, and the uniformity of L-PUF is 49.8%. The number of 1s in the RL-PUF output response is 77522, and the uniformity is 49.8%.

2) UNIQUENESS

A good PUF design should have good uniqueness. When different PUF instances are implemented on different devices, different instances will produce different responses for the same challenge. Uniqueness measures inter-chip variation by evaluating how will design can differentiate d different devices. It can be calculated using the inter-chip Hamming distance (HD) as shown in (2). R_i and R_j represent the n-bit responses generated from two chips i and j using the same challenge C.

$$Uniqueness = \frac{2}{d\,(d+1)} \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{HD\left(R_i, R_j\right)}{n} \times 100 \tag{2}$$

Ideally, when a PUF circuit is implemented on different devices it should produce an average inter-chip HD close to 50% when supplied with the same challenge, implying that half the response bits are different between the two devices even though the same challenge has been used.

We test 16 L-PUF responses for each of the four challenges, and each PUF record the responses of the first 100 clock cycles. There are a total of 480 HD values for the 16 groups of responses per cycle. The response of the 63th clock cycle is counted. The maximum inter-chip HD is 43, the minimum is 22, and the average is 32.16, which is close to the ideal value of 32. That is, the uniqueness value of the L-PUF is 50.25%. We test all the LR-PUF instances under same conditions, and the uniqueness value is 49.66%. The probability histogram of the inter-chip HD is shown in Fig. 12 and Fig. 13.

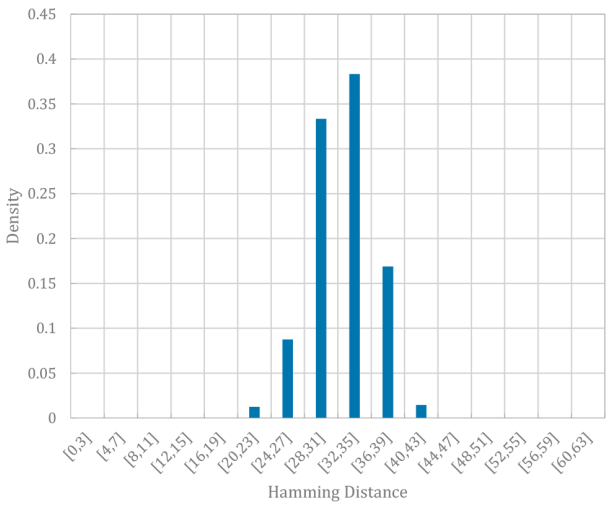

FIGURE 12. Inter-chip hamming distance of L-PUF.

3) RELIABILITY

Ideally, a PUF design, regardless of its implementation on any device, should have a fully reproducible output response. That is, its response to the same challenge should be exactly the same at any time. However, due to environmental factors such as temperature and supply voltage fluctuations, there is noise in the output response of the PUF. Therefore, the reliability of the PUF response can be represented by the ratio of the noise bits in the output response. For a device, reliability can be evaluated by its average intra-chip HD as

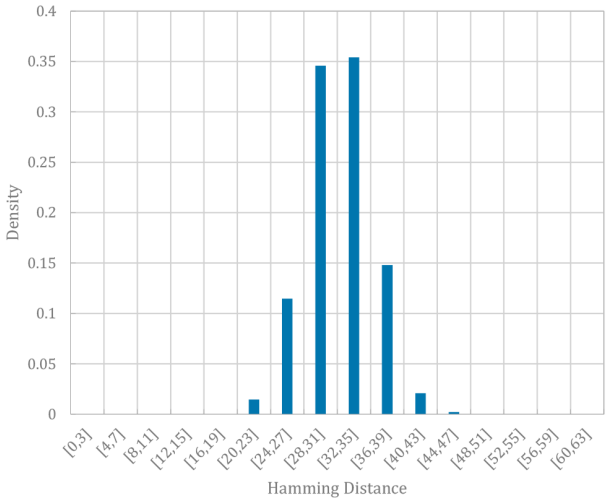

FIGURE 13. Inter-chip hamming distance of RL-PUF.

defined in (3).

$$HD_{intra} = \frac{1}{s}\sum_{t=1}^{s}\frac{HD\left(R_i, R'_{i,t}\right)}{n} \times 100 \qquad (3)$$

We obtain a total of n-bit responses for the s group. For each response, R_i is measured under normal operating conditions, and R'_i is measured at different supply voltages and temperatures. $R'_{i,t}$ is the t-th sample of R'_i. The reliability is equal to $100-HD_{intra}$.

For L-PUF designs that use Weak PUF to construct Strong PUFs, the process deviation of the front-end Weak PUF is used as the entropy source of the whole system, and the back-end obfuscation logic is usually a stable and mature structure. Therefore, the reliability of the L-PUF mainly depends on the reliability of the front-end Weak PUF. Anderson computed the HD between the responses of 36 128-bit PUF that designed on Xilinx Virtex-5 FPGA board at high and low temperature. The result shows that 72% of the responses changed by 5 or fewer bits and no response experienced more than 10 bit flips [24]. The average number of bit flips is 2.38 and the HD_{intra} is 3.7%. So, the reliability of this Weak PUF design is 96.3%.

But in this design the use of LFSR as obfuscation logic will cause reliability problem. The LFSR logic will amplify the unreliability of Weak PUF. So, we must use reconstruction method to ensure that the responses are all regenerated. We should use some Weak PUF design with good stability [19] or some lightweight error correction scheme [25]. Moreover, instead of the error correction mechanism, we can also choose the error tolerance mechanism according to the application scenario. With a delicate error-tolerant mechanism we can avoid using expensive overhead error correction circuit. These tasks are left as future work of this study.

4) MULTI-FUNCTIONALITY

We calculate the average inter-chip HD distribution of the L-PUF and RL-PUF from the 63rd clock cycle to the 100th clock cycle. The results show that the distribution of the average HD is slightly different, and the deviation is small. The average HD, the mean and standard deviation for two PUFs are tabulated in TABLE 2. That result shows that the uniqueness of the PUF design is stable over time. The average inter-chip HD fluctuating with clock cycle is shown in Fig.14.

TABLE 2. Average inter-chip HD results from clock cycle 63 to 100.

Average HD	Min	Max	μ	σ
L-PUF	31.32	32.78	31.97	0.148
LR-PUF	30.6	32.72	31.95	0.246

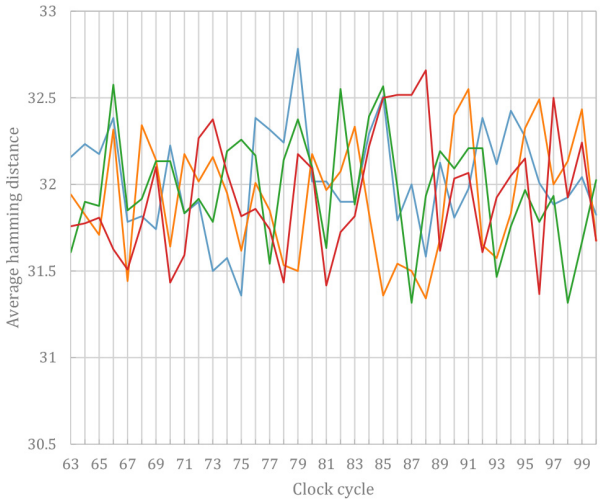

FIGURE 14. The distribution of average inter-chip HD of all the PUFs from clock cycle 63 to 100.

Since the outputs of L-PUF and RL-PUF are different at each clock cycle, we can equivalent one PUF to multiple PUFs by outputting the response after the same period. This feature is suitable for applications that require multiple PUFs to improve system security, and can further reduce area overhead. Moreover, it further enhances the unclonability of PUF since cloning the CRPs spaces for all the possible periods is considered impractical.

5) COMPARISON WITH RELATED WORK

A comparison of L-PUF and LR-PUF with other PUF designs is listed in TABLE 3. The L-PUF and LR-PUF designs achieve better hardware overhead on FPGA than previous Strong PUF designs and nearly have the same uniqueness. Particularly, with the changeable output period both the two PUFs can offer numerous configuration options, which can exponentially increase the CRPs number.

V. SECURITY ANALYSIS

As a security sensitive device, PUF faces a variety of security threats. One of the most powerful threat in this category

TABLE 3. Comparison with conventional Strong PUF design.

Design	Arbiter PUF[26]	Arbiter PUF[18]	Feedforward Arbiter PUF[17]
Response(bit)	64	64	-
Uniqueness	23%	9.42%	38%
Reliability	99.20%	-	90.20%
Hardware	TSMC 180nm	Artix-7	TSMC 180nm
Overhead	1212um×1212um	129×64 SLICEs	-
PUF type	FF-APUF[27]	L-PUF	LR-PUF
Response(bit)	64	64	64
Uniqueness	40%	50.25%,	49.66%
Reliability	97.10%	96.3%*	96.3%*
Hardware	Artix-7	Artix-7	Artix-7
Overhead	44×64 SLICEs	210 SLICEs	129 SLICEs

is modeling attack, where the attacker builds a software model of the PUF and intentionally collects a large set of CRPs to train the model. According to the well-trained model they can accurately predict responses of unknown challenges. The modeling attack on Strong PUF was proposed by Rührmair et al. in 2010. They attacked some classic Strong PUF designs, such as RO, arbiter PUF and various improved designs, using machine learning algorithms such as Logistic Regression and Support Vector Machine. The CRP prediction accuracy of the arbiter PUF is as high as 99% [28]. With the development of hardware level and attack algorithms, the time and accuracy of modeling is getting higher and higher. Although our design has good performance and small overhead, the resilience to modeling attack should be further discussed.

A. SECURITY OF L-PUF

The modeling attacks is mainly used to compromise Strong PUF. A practical Strong PUF design should have a large CRP space. As we mentioned, attackers can predict unknown responses by a well-trained PUF model. According to the attacking principle, for designing a modeling attack resistant Strong PUF, the randomness of CRP space must be increased and the correlation between CRPs must be reduced. It often complicates the PUF design and results in a significantly increased hardware overhead. It's necessary to strike a balance between security and overhead.

As a widely used PRNG, LFSR does have security problems. It is really not safe to use it directly as CSPRNG (cryptographically secure PRNG). The random sequences generating by LFSR are not cryptographically secure. The construction of an LFSR of length n-bits can be easily deduced by observing the $2 \times N$ consecutive bits of its sequence using the Berlekamp-Massey algorithm [29]. Due to its inherent linearity, LFSR-based ciphers are vulnerable to several form of attacking methods, such as machine learning algorithms. However, LFSR is widely used in stream ciphers, because it is simple, fast, and easy to implement for both software and hardware. So, different security enhancement on our design must be considered according to the actual application scenario.

B. SOME ENHANCEMENT METHODS

There are some methods to increase the linear complexity and randomness of LFSR. We discuss three mostly used of them as follows.

1) COMBINATION OF LFSRS

In this method, the output of several LFSRs is combined by a Boolean function f to produce secure and random responses. The function f has to satisfy certain criteria. For low hardware overhead, f can be XOR logic. Fig. 15 illustrates the general construction of this method.

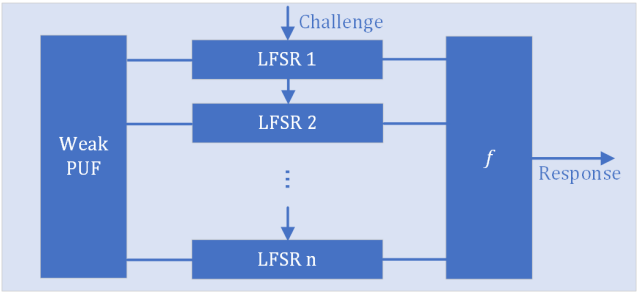

FIGURE 15. The combination of LFSRs.

2) OUTPUT FILTER

Only a single LFSR also can produce secure response by filtering the output. A Boolean function can be used to generate response by filtering and combining the contents of the LFSR.

3) LFSR-BASED LIGHTWEIGHT CIPHER

For better security and attack resistant, the LFSR in our design can be replaced with some LFSR-based lightweight cipher, like Trivium [30], etc. This method may increase the area overhead, but it can greatly improve the cryptographical security and attack resilience of the design.

VI. CONCLUSION

This paper improves the place and route method of a PUF that utilizes FPGA logic cell structure, then combines it with the classic LFSR architecture to propose a new Strong PUF that can be dynamically reconfigured. The PUF has a simple structure and can be conveniently implemented in FPGA, and has a greatly improved performance compared with a traditional Strong PUF. It needs extra error correction circuit or error tolerance scheme to ensure the reliability of front-end Weak PUF, but the high reliability of well-designed Weak PUF will make this overhead as small as possible. Moreover, compared to other design, its area overhead is still very small. This feature is very suitable for applications that are sensitive to hardware overhead. We implement it on a 28nm FPGA evaluation board. Experimental results show that the PUF design is satisfactory in uniformity, uniqueness, and reliability. We analyze the security of the design, and propose some enhancement method to improve the modeling attack resistance. In future, we will use more FPGA devices to build

a larger test set and increase the reliability of evaluation. Also, it is necessary to quantitatively evaluate the resistance of the design to modeling attack.

REFERENCES

[1] A. G. Schmidt, G. Weisz, and M. French, "Evaluating rapid application development with Python for heterogeneous processor-based FPGAs," in *Proc. IEEE 25th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2017, pp. 121–124.

[2] A. Putnam, "Large-scale reconfigurable computing in a microsoft datacenter," in *Proc. IEEE Hot Chips 26 Symp. (HCS)*, Cupertino, CA, USA, Aug. 2014, pp. 1–38.

[3] *Tencent Cloud*. Accessed: Mar. 8, 2019. [Online]. Available: https://intl.cloud.tencent.com/

[4] *Apple iPhone 7 Teardown*. Accessed: May 21, 2019. [Online]. Available: https://www.techinsights.com/blog/apple-iphone-7-teardown

[5] S. McNeil, "Solving today's design security concerns," Xilinx, San Jose, CA, USA, White Paper WP365 (v1.2), 2012, p. 14.

[6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems*, vol. 4727, P. Paillier and I. Verbauwhede, Eds. Berlin, Germany: Springer, 2007, pp. 63–80.

[7] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "Emerging physical unclonable functions with nanotechnology," *IEEE Access*, vol. 4, pp. 61–80, 2016.

[8] J. Zhang, X. Tan, X. Wang, A. Yan, and Z. Qin, "T2FA: Transparent two-factor authentication," *IEEE Access*, vol. 6, pp. 32677–32686, 2018.

[9] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002.

[10] P. Bulens, F.-X. Standaert, and J.-J. Quisquater, "How to strongly link data and its medium: The paper case," *IET Inf. Secur.*, vol. 4, no. 3, pp. 125–136, 2010.

[11] G. Hammouri, A. Dana, and B. Sunar, "CDs have fingerprints too," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2009, pp. 348–362.

[12] K. Lofstrom, W. R. Daasch, and D. Taylor, "IC identification circuit using device mismatch," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2000, pp. 372–373.

[13] J. Guajardo *et al.*, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Inf. Syst. Frontiers*, vol. 11, no. 1, pp. 19–41, Mar. 2009.

[14] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems*, vol. 4249, L. Goubin and M. Matsui, Eds. Berlin, Germany: Springer, 2006, pp. 369–383.

[15] C.-E. Yin, G. Qu, and Q. Zhou, "Design and implementation of a group-based RO PUF," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, 2013, pp. 416–421.

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

[17] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[18] Y. Hori, H. Kang, T. Katashita, A. Satoh, S. Kawamura, and K. Kobara, "Evaluation of physical unclonable functions for 28-nm process field-programmable gate arrays," *J. Inf. Process.*, vol. 22, no. 2, pp. 344–356, 2014.

[19] M. Bhargava and K. Mai, "An efficient reliable PUF-based cryptographic key generator in 65 nm CMOS," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–6.

[20] L. Santiago *et al.*, "Realizing strong PUF from weak PUF via neural computing," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Cambridge, MA, USA, Oct. 2017, pp. 1–6.

[21] M. Goresky and A. M. Klapper, "Fibonacci and Galois representations of feedback-with-carry shift registers," *IEEE Trans. Inf. Theory*, vol. 48, no. 11, pp. 2826–2836, Nov. 2002.

[22] P. L'Ecuyer, "Tables of linear congruential generators of different sizes and good lattice structure," *Math. Comput.*, vol. 68, no. 225, pp. 249–261, Jan. 1999.

[23] W. Liu, Z. Lu, H. Liu, R. Min, Z. Zeng, and Z. Liu, "A novel security key generation method for SRAM PUF based on Fourier analysis," *IEEE Access*, vol. 6, pp. 49576–49587, 2018.

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

[25] Z. He, M. Wan, J. Deng, C. Bai, and K. Dai, "A reliable strong PUF based on switched-capacitor circuit," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 6, pp. 1073–1083, Jun. 2018.

[26] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Proc. Symp. VLSI Circuits. Dig. Tech. Papers*, Honolulu, HI, USA, Jun. 2004, pp. 176–179.

[27] C. Gu, Y. Cui, N. Hanley, and M. O'Neill, "Novel lightweight FF-APUF design for FPGA," in *Proc. 29th IEEE Int. Syst.-Chip Conf. (SOCC)*, Seattle, WA, USA, Sep. 2016, pp. 75–80.

[28] 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)*, Chicago, IL, USA, 2010, p. 237.

[29] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. Hoboken, NJ, USA: Wiley, 1995.

[30] N. Mentens, J. Genoe, B. Preneel, and I. Verbauwhede, "A low-cost implementation of Trivium," in *Proc. SASC*, 2008, p. 8.

SHEN HOU received the B.E. degree in microelectronics from Nanjing University, Jiangsu, China, in 2005, and the M.S. degree in microelectronics from the National University of Defense Technology, Hunan, China, in 2008, where he is currently pursuing the Ph.D. degree in microelectronics. His main research interests include microprocessor design, hardware security, embedded systems, and the IoT application.

YANG GUO received the Ph.D. degree from the National University of Defense Technology, Hunan, China, in 1999. He is currently a Professor with the National University of Defense Technology, where he leads the Digital Signal Processor Group and is the Director of the Integrated Circuits. He has authored or coauthored more than 50 publications on journals and conference proceedings. His primary research interests include low power VLSI circuits, microprocessor design and verification, and electronic design automation (EDA) techniques for VLSI circuits.

SHAOQING LI received the B.S. and the M.S. degrees in computer application from the National University of Defense Technology, where he has been with the School of Computer, since 1984. From 1995 to 2001, he was an Associate Professor. He has been a Professor, since 2002. His research interests include microprocessor design, test, and security.