A Lightweight and Secure-Enhanced Strong PUF Design on FPGA

Shen Hou12a, Yang Guo1, Shaoqing Li1, Ding Deng1 and Yan Lei3

Abstract  Physical unclonable function (PUF) is a reliable physical security primitive. The Weak PUF and Strong PUF are two well-known PUF topologies. Strong PUF can be used to authenticate and protect intellectual property on FPGA chips. Classic PUF designs, like arbiter PUF, are hard to implement on FPGA and severely threatened by the machine learning based modeling attacks. In this work, we propose a new Strong PUF on FPGA by combining Weak PUF with obfuscation logic. Experiment results on a 28nm FPGA show that the resistance to modeling attack is good and the hardware overhead is small.

key words: physical unclonable function, lightweight, FPGA, feedback shift register, modeling attack

Classification: Integrated circuits

1. Introduction

In recent years, big data, artificial intelligence, and cloud technologies have developed rapidly. As a hardware reconfigurable architecture, FPGA has strong computing power and sufficient flexibility. As a kind of accelerator that has been paid increased attention in the field of deep learning, FPGA has become a new research and application hotspot. Widespread use of FPGAs brings new security challenges, such as overbuilding, tampering, cloning, and reverse engineering [1]. Physical unclonable function (PUF) can extract some random deviations in the manufacturing process 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. PUF generates a corresponding response only when a special challenge is given. It is called “Challenge-Response” mechanism. According to the relationship between the number of CRPs and the size of physical entities, PUF can be defined as Weak PUF and Strong PUF [2]. PUFs with limited number of CRPs, known as Weak PUFs, are commonly used for key generation in cryptographic functions [3, 4]. Memory-based PUF, like SRAM PUF[5], butterfly PUF [6] is one kind of Weak PUF. It uses asymmetry caused by manufacturing process deviation between the cross-coupling gate devices in most memory cells to produce responses. RO (ring oscillator) PUF is also a Weak PUF. It can generate responses by comparing the oscillation frequencies of two identical RO circuits [7, 8]. The Strong PUFs have exponential number of CRPs and are suitable for authentication [9, 10]. A typical kind of Strong PUF is the arbiter PUF [11, 12]. It contains a multi-level multiplexer chain which has two identical paths. It can output responses by an arbiter behind the chain to announce which path is faster. Due to these features, PUF can provide anti-tamper solutions to protect IP and sensitive data in FPGAs [13, 14].

However, PUF designs on FPGA are few because of the following reasons. Highly regular architecture of FPGA and the automated design method make it very difficult to ensure identical circuit implementation on different chips [15]. For delay-based PUFs, automatic routing and optimizing causes a wire length difference between original design and actual circuit. Furthermore, the security of PUFs has become a serious issue. The machine learning based modeling attacks, which are non-invasive attack method, can successfully break various Strong PUFs [16].

To address some of the issues outlined above, a new lightweight FPGA-based Strong PUF is proposed. The research contributions of this paper are as follows: 1) A new compact FPGA-based time-delay Weak PUF which can generate 2-bit response in 1 slice is implemented on 28 nm FPGA. 2) A lightweight Strong PUF is construct by using some structural features of FPGA. 3) The proposed Strong PUF has small hardware overhead and good resistance to machine learning based attacks.

The rest of the paper is organized as follows. Section 2 reviews related research work on FPGA PUF and machine learning attack. Section 3 details the proposed Strong PUF design. Section 4 gives the experiments.
results and performance analysis. Final is the conclusion section.

2. Related Work

2.1 PUF designs on FPGA

Anderson claimed to implement the PUF structure on the FPGA for the first time [17]. His design refers to the basic idea of the delay-based PUF, and takes advantage of SLICEM. It is implemented on Xilinx Virtex-5 FPGA board and cost 1 SLICEM for each response bit. The PUF identification generator proposed by Gu et al. in [18], [19] uses the flip-flop element in slice as the delay path and the cross-coupled NAND gates as an arbiter. Due to the deviation of the manufacturing process, random responses can be generated. They implemented and analyzed the performance on the Spartan-6 series FPGA evaluation board. The results show that the design has better device uniqueness and reliability, and only one slice is needed to generate a single ID on the 6 series FPGA. The hardware overhead is smaller than Anderson’s design, and according to new architecture of the 7 series FPGA, the overhead can be further reduced.

2.2 Realizing Strong PUF from Weak PUF

As a classic digital Strong PUF design, arbiter PUF has obvious shortcomings. The implementation of time-delay PUF on FPGA chips is difficult and the hardware overhead is relatively large. These problems limit the application of the arbiter PUF in FPGAs authentication. In general, the stability and response uniqueness of Weak PUF is better than the existing Strong PUF design [20, 21]. 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 designed with great sophistication to maintain unclonability of PUF. Several types of obfuscation logic have been proposed, such as AES [22], and neural network [23].

2.3 Strong PUF and machine learning attacks

The basic structure of arbiter PUF with $n$-bit challenge $c[0]$-$c[n-1]$ and 1-bit response $r$ is shown in Fig. 2. The same input is connected to the upper and lower paths, and as the control signals of each stage, challenge determines the two multiplexers status. The process variations result in a slight transition time difference of two paths. A latch behind the chain acts as an arbiter to judge which path is faster. If upper signal arrives earlier, then $r$ is 1, or else $r$ is 0.

![Fig. 1 The underlying architecture of Xilinx 7 series FPGA](image1)

Fig. 1 depicts the underlying architecture of latest Xilinx 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, three multiplexers, 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. 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. This arrangement is called ASMBL architecture. Some compact PUF circuits can be designed based on some features of this architecture.
\[ \bar{W} = [\alpha_0, W_1, \ldots, W_{n-1}, \beta_n], \ W[i] = \alpha_{i+1} + \beta_i \]

\[ \Phi = [\Phi_0, \ldots, \Phi_{n}], \ \Phi[i] = \prod_{j=i}^{n-1} (1 - 2c[j]) \]

For a known CRP, the value of \( r \) and \( \Phi \) are known. By using known CRPs and machine learning algorithms, attackers can train the model and figure out the vector \( \bar{W} \). As a result, the unknown responses of other challenges can be predicted.

Some new design methods are proposed to resist modeling attack like the XOR PUF [9], the lightweight PUF [25]. But They are compromised under new attack algorithms and powerful hardware.

3. Circuit design

3.1 The Weak PUF design

The circuit structure of [18] is shown in Fig. 3. Two flip-flops of the same structure are controlled by the same clock and reset signal. After CLEAR resets the two flip-flops, the clock signal is used as the START signal to simultaneously assign values to the two paths. Due to manufacturing process deviation, the signal propagation time of the two paths is slightly different, and the time of Q0 and Q1 arriving at two NAND is different too. Thus, the output values of the cross-coupled NAND gate Z0 and Z1 are set to “01” or “10”. In such a manner, the two states output logic 0 and logic 1 respectively, controlled by a multiplexer to generate 1-bit response. According to the internal structure of slice, each NAND can be generated by one LUT. The multiplexer and flip-flops can directly use the internal logic elements of slice. Therefore, the 1-bit response can be implemented by one single slice.

![Fig. 3 Gu's 1-bit FPGA ID cell circuit](image)

In the 7 series FPGA, there are 8 flip-flop elements, 4 LUTs, and 3 general-purpose multiplexers in each slice. Each group of four flip-flops falls under the same function and structure. Each LUT can generate 4 NANDs respectively. Therefore, by fine-constrained placement and routing, each slice in 7 series can generate 2-bit PUF response, as shown in Fig. 4. The LUT A, LUT B, and MUX A in the red dashed box of the SLICEL can be directly connected to the two flip-flops on the right to form a 1-bit response generation circuit. The LUT C, LUT D and MUX B on the top can be connected to the two flip-flops on the right to generate another response.

![Fig. 4 2-bit response generate circuit](image)

3.2 The Strong PUF design

In the ASMBL architecture of Xilinx FPGAs, the LUT in SLICEM can also be configured as an SRL with a maximum shifting depth of 16 or 32 bits. When configured as a 32-bit SRL as shown in Fig. 5, the shift depth can be dynamically adjusted from 1 to 32 bits by five input A4~A0 of the LUT.

![Fig. 5 1-bit response generate circuit](image)
FF4 is output as a response bit. Theoretically, only 6 slices can implement core structure of a PUF bit. However, the first problem is that the 10 bits depth control signals only have $2^{10}$ kind of arrangements. It means that we can get 1024 different PUF instances at most, and that is far not enough for actual application of Strong PUF. One solution is to connect more than one SLICEM in series as SLICEM1 and SLICEM2 in Fig. 6. Although the hardware overhead is doubled, it is still acceptable and the PUF instance number is exponential increased.

The second problem is the value of LFSR cannot be all 0s. It can be easily avoided by assign all the SRL32 a non-all-zero initial value.

4. Experimental result

4.1 FPGA implementation

The 2-bit response generating circuit is implemented on an Alinx development board (Zynq-7000 XC7Z020 FPGA). The final implementation on FPGA is shown in Fig. 7. We divide the FPGA side of the Zynq-7000 chip into 16 regions, implementing a 32-bit WPUF and a 32-bit s-PUF in each region. Then we do the same work on 3 development boards and get 48 32-bit PUF instances in total. The challenges and responses are exchanged with PC through the UART interface. The experimental data is recorded and processed using Python.

4.2 Hardware overhead

The core part of a 32-bit s-PUF design uses 1536 of the 53200 LUTs as logic on the Zynq-7000 XC7Z020 FPGA (2.89%), 128 of the 17400 LUTs as memory (0.74%), and 1408 of the 53200 LUTs as logic (2.65%). The experimental results show that the PUF design in this paper has a small hardware overhead and is lightweight. Detailed data are shown in Table I.

Table I Hardware utilization

| PUF type                  | 32 bits s-PUF |
|---------------------------|---------------|
| Logic function Used       | Available     | Util% |
| Slice LUTs                | 1536          | 53200 | 2.89 |
| LUTs as Memory 128        | 17400         | 0.74  |
| LUTs as Logic 1408        | 53200         | 2.65  |
| Slice Registers 1568      | 106400        | 1.47  |

4.3 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 ideal value for uniformity is 50%, meaning that the probability of 0 and 1 in the PUF response is supposed to be identical. We separately calculate the uniformity of the front-end WPUF and the overall s-PUF output response. A total of 1536 bits responses are generated by 48 32-bit PUFs. The number of 1s is 738, and the uniformity is 48%, which is close to the ideal value. After loading 4 32-bit challenges into all the 48 s-PUF instances, 192 32-bit responses are generated, and we get 6144 bits response.
The number of 1s is 3047, and the uniformity of Strong PUF is 49.6%.

2. Uniqueness

A good PUF design should have good uniqueness. When different PUF instances are implemented on different devices, different responses are produced for the same challenge. Uniqueness measures inter-chip variation by evaluating how the design differentiates \( d \) different devices. It can be calculated with the inter-chip Hamming distance (HD) as shown in (3). \( R_i \) and \( R_j \) represent the \( n \)-bit responses generated from two chips \( i \) and \( j \) using the same challenge \( C \).

\[
\text{Uniqueness} = \frac{2}{d(d+1)} \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} \frac{HD(R_i R_j)}{n} \times 100 \quad (3)
\]

Ideally, implemented on different devices, a PUF circuit is expected to 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 48 PUF instances with four challenges and get a total of 4512 HD values. The maximum inter-chip HD is 28, the minimum is 4, and the average is 16.19, which is close to the ideal value of 16. That is, the uniqueness value of the PUF is 50.58%. The probability histogram of the inter-chip HD is shown in Fig. 8.

3. Reliability

Ideally, a PUF design should have a fully reproducible output response. 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 \(-\) \( HD_{\text{intra}} \).

To investigate reliability the responses are measured at 25°C through to 70°C using a temperature chamber while the FPGA supply voltage was varied by 1.2 Volts ±10% using a DC regulated power supply. The average number of bit flips is 2.14 and the \( HD_{\text{intra}} \) is 6.7%. So, the reliability of this WPUF design is 93.3%.

In our designs, the process deviation of the front-end WPUF is used as the entropy source of the whole system, and the back-end obfuscation logic is a stable and mature structure. Therefore, the reliability of the p-SPUF is similar to that of WPUF.

4. Security

Cryptanalysis and machine learning based modeling attack are two mathematical approaches to analyze the security of PUFs.

1) As a broadly applied PRNG, the randomness and unpredictability are very important. The evaluation result shows that the output sequence of p-SPUF has good randomness. As a linear system, LFSR with fixed length and taps is leading to fairly easy cryptanalysis [28]. But in this design, random different seeds are chosen as challenges which is changing every application period. So, the output sequences are changed when a response bit is output. Moreover, only one single random bit is used in output sequence, making the cryptanalysis of PUF instances more difficult.

2) The randomness of p-SPUF mainly comes from three aspects: process variation of the front-end WPUF, challenge sequence value and the shifting cycles number. The 1-bit response generating circuit has identical structure but every bit is actually having different shifting length and taps position. The modeling of p-SPUF is harder than the arbiter PUF. The attack methods in [29] are used to evaluate the capability of p-SPUF in resisting machine learning attack. The experiment is run on PC with Intel i7 3.6GHz CPU. We measure 1000 CRPs of a 32-bit p-SPUF and use three classification algorithms (Logistic Regression, SVM and ANN) to predict response bit in data set. The results of average prediction are 65.6%, 51.7%, 56.8%. In contrast, as a broadly studied Strong PUF, the arbiter PUF has been compromised by modeling attack both on FPGA and ASIC. The modeling algorithm is described in section 2.3. The LR method has the best attack efficiency and the prediction rate of 64-bit arbiter PUF on FPGA is 99% [29]. Therefore, the p-SPUF has better resistance to modeling attack.

4.4 Comparison of performance and overhead

A comparison of our design with two classic Strong PUF designs implemented on same FPGA platform is listed in Table II. Our design has efficient area usage and smallest number of slices per bit than previous Strong PUF designs. The uniqueness of p-SPUF are better.
Table II Comparison with traditional arbiter PUF designs

|                     | arbiter PUF [21] | FF-APUF [30] | p-SPUF |
|---------------------|------------------|--------------|--------|
| Response(bit)       | 64               | 64           | 32     |
| Uniqueness          | 9.42%            | 40%          | 50.58% |
| Reliability         | -                | 97.10%       | 93.3%  |
| Hardware            | Artix-7          | Artix-7      | Artix-7|
| Overhead            | 129x64 slices    | 44x64 slices | 12x32 slices |

5. Conclusion

This paper improves the place and route method of a PUF that utilizes FPGA logic cell structure, then combines it with the SRL architecture to propose a Strong PUF. The PUF has a simple structure and can be conveniently implemented in FPGA. Moreover, compared with other design, its hardware overhead is small and suitable for applications that are sensitive to hardware overhead. Obfuscation method is introduced to increase security. We implement it on a 28nm FPGA evaluation board. Experimental results show that the PUF design is satisfactory in uniformity uniqueness, reliability and security.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No 61832018.

References

[1] S. McNeil: “Solving Today’s Design Security Concerns” (Xilinx, 2012).
[2] R. Maes: “Physically Unclonable Functions: Constructions, Properties and Applications,” Ph.D Dissertation, University of Leuven (2012).
[3] J. Delvaux, et al.: “Helper data algorithms for PUF-based key generation: Overview and analysis,” IEEE Trans. Comput-Aided Des. Integr. Circuits Syst. 34 (2015) 889 (DOI: 10.1109/TCAD.2014.2370531).
[4] M. Yu, et al.: “Performance metrics and empirical results of a PUF cryptographic key generation ASK,” IEEE HOST (2012) 108 (DOI: 10.1109/HST.2012.6224329).
[5] J. Lee, D.-W. Jee, and D. Jeon: “Power-Up Control Techniques for Reliable SRAM PUF,” IEICE Electron. Express 16 (2019) 20190296 (DOI: 10.1587/elex.16.20190296).
[6] X. Xu et al.: “A highly reliable butterfly PUF in SRAM-based FPGAs,” IEICE Electron. Express 14 (2017) 20170551 (DOI: 10.1587/elex.14.20170551).
[7] C.-E. Yin, G. Qu, and Q. Zhou: “Design and Implementation of a Group-based RO PUF,” IEEE/ACM DATE (2013) 416 (DOI: 10.7873/DATE.2013.094).
[8] S. Pei, J. Zhang, and R. Wang: “A low-overhead RO PUF design for Xilinx FPGAs,” IEICE Electron. Express 15 (2018) 20180093 (DOI: 10.1587/elex.15.20180093).
[9] G. E. Suh and S. Devadas: “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” ACM DAC (2007) 9 (DOI: 10.1145/1278480.1278448).
[10] W. Che, F. Saqib, and J. Plusquelle: “PUF-based authentication,” IEEE ICCAD (2015) 337 (DOI: 10.1109/ICCAD.2015.7372589).
[11] Daebyun Lim, et al.: “Extracting secret keys from integrated circuits,” IEEE Trans. Very Large Scale Integr. Syst. 13 (2005) 1200 (DOI: 10.1109/TVLSI.2005.859470).
[12] S. Ghosh, and R. Govindaraj: “A strong arbiter PUF using resistive RAM,” Springer SAMOS (2016) 275 (DOI: 10.1109/SAMOS.2016.7818358).
[13] S. S. Kumar, et al.: “Extended abstract: The butterfly PUF protecting IP on every FPGA,” IEEE HOST (2008) 67 (DOI: 10.1109/HST.2008.4709453).
[14] J. Guajardo, et al.: “FPGA Intrinsic PUFs and Their Use for IP Protection,” Springer CHES (2007) 63 (DOI: 10.1007/978-3-540-74735-2_5).
[15] C. Gu, and M. O’Neill: “Ultra-compact and robust FPGA-based PUF identification generator,” IEEE ISCAS (2015) 934 (DOI: 10.1109/ISCAS.2015.7168788).
[16] U. Rührmair, et al.: “Modeling attacks on physical unclonable functions,” ACM CCS (2010) 237 (DOI: 10.1145/1866307.1866335).
[17] J. H. Anderson: “A PUF design for secure FPGA-based embedded systems,” ACM/IEEE ASP-DAC (2010) 1 (DOI: 10.1109/ASPDAC.2010.5419927).
[18] C. Gu, J. Murphy, and M. O’Neill: “A unique and robust single slice FPGA identification generator,” IEEE ISCAS (2014) 1223 (DOI: 10.1109/ISCAS.2014.6865362).
[19] C. Gu, N. Hanley, and M. O’Neill: “FPGA-based strong PUF with increased uniqueness and entropy properties,” IEEE ISCAS (2017) 1 (DOI: 10.1109/ISCAS.2017.8050838).
[20] Y. Gao, et al.: “Emerging Physical Unclonable Functions with Nanotechnology,” IEEE Access 4 (2016) 61 (DOI: 10.1109/ACCESS.2015.2503432).
[21] Y. Hori, et al.: “Evaluation of Physical Unclonable Functions for 28-nm Process Field-Programmable Gate Arrays,” J. Inf. Process. 22 (2014) 344 (DOI: 10.2197/ipsjjp.22.344).
[22] M. Bhargava, and K. Mai: “An efficient reliable PUF-based cryptographic key generator in 65nm CMOS,” IEEE/ACM DATE (2014) 1 (DOI: 10.7873/DATE.2014.083).
[23] L. Santiago et al.: “Realizing strong PUF from weak PUF via neural computing,” IEEE DFT (2017) 1 (DOI: 10.1109/DFT.2017.8244433).
[24] B. L. P. Gassend: “Physical Random Functions,” Master Dissertation, Massachusetts Institute of Technology (2003).
[25] M. Majzoobi, F. Koushanfar, and M. Potkonjak: “Lightweight secure PUFs,” IEEE/ACM ICCAD (2008) 670 (DOI: 10.1109/ICCAD.2008.4681648).
[26] “Vivado Design Suite User Guide: Using Constraints (UG903)” (2018) 194.
[27] “Vivado Design Suite Tutorial: Using Constraints (UG945)” (2018) 47.
[28] B. Schneier: Applied cryptography: protocols, algorithms, and source code in C (Wiley, New York, 1995) 2nd ed. 40.
[29] U. Rührmair et al.: “PUF modeling attacks on simulated and silicon data,” IEEE Trans. Inf. Forensics Secur. 8 (2013) 1876 (DOI: 10.1109/TIFS.2013.2279798).
[30] C. Gu, et al.: “Novel lightweight FF-APUF design for FPGA,” IEEE SOCC (2016) 75 (DOI: 10.1109/SoCC.2016.7905439).