IMPROVE SYMMETRY OF ARBITER IN APUF

YANG Xu, HUANSHENG Ning* AND LINGFENG Mao
School of Computer & Communication Engineering
University of Science and Technology Beijing
Beijing 100086, China

YOUZHONG Li AND LIJUN Zhang
School of Urban Rail Transportation, Soochow University
Suzhou 215006, China

(Communicated by Zhipeng Cai)

ABSTRACT. Arbiter-based physical unclonable function (APUF) is a classical kind of physical unclonable function (PUF). In APUF-based device authentication, the fairness of traditional APUF is insufficient due to setup time of arbiter. To solve this problem, in this paper we design an arbiter and conduct Monte Carlo simulations to test the performance of the new arbiter. In addition, we present some new evaluation metrics to evaluate the new arbiter quantitatively. Finally, we certify that the new arbiter can work continuously with both one stage racing paths and eight stages racing paths. The new arbiter has good performance in correct rate, stability and fairness. Particularly, it mitigates the setup time problem by reducing the Asymmetry.

1. Introduction. In 2000, Lofstrom et al. got the idea of identification using device mismatch [10]. In 2002, Pappu et al. defined physical one-way function and proposed optical PUF [12]. Gassend et al. implemented PUF in silicon circuit, presented the definition of PUF, and designed an original delay-based PUF [5][2]. In 2003, he improved the definition of PUF. Those works laid the foundation of PUF area [3]. In [3], the definition of PUF is a function that maps challenges to responses. It is embodied by a physical device, easy to evaluate and hard to predict. Silicon PUF uses manufacturing process variations in integrated circuits (IC) with identical masks to uniquely characterize each IC.

In 2004, J. Lee, D. Lim, B. Gassend et al. proposed a new type of PUF called Arbiter-PUF to enhance the robustness as well as reduce the operation time and power dissipation [4][7]. From then on, may kinds of PUFs have been invented. Most PUFs come in two flavors: silicon PUF and non-silicon PUF [16]. Non-silicon PUF includes Optical PUF, paper PUF, CD PUF, acoustical PUF, magnetic PUF, RF-DNA, etc. Silicon PUF, like memory-based PUF, delay-based PUF, refers to PUF that implement in a silicon circuit. And as a security primitive, PUF can work...
together with other security policy like fingerprint and face identification[6]. In this paper, we study only APUF which is a kind of delay-based PUF.

A 64 stages APUF was designed in [4] [7] to improve the original delay-based PUF in [2][3]. The 64 bits of a challenge correspond to 64 switches (also called stage). There are two lines that go through the 64 switches. Every bit in the challenge decides whether the two lines go parallelly or crosswise in the corresponding switch. Even if we feed a rising edge to the two lines simultaneously, the delay time of the two lines will be different, as the delay of switches and lines are affected by uncontrollable intrinsic variation. The line is called delay line, and the two lines together are called a pair of racing paths. All those components that cause the delay difference ($\Delta t$) between the two delay lines is the first part of APUF circuit. The second part of APUF is an arbiter that tests which delay line outputs the rising edge first. If the upper delay line is faster, the arbiter should output a 1, and vice versa. That is, the arbiter transforms the $\Delta t$ to a 1 bit response. Obviously, for different challenges, the switch selects different paths, the $\Delta t$ is different, finally the response is different. A challenge and the corresponding response are termed challenge response pair (CRP). For a 64 stages APUF, each challenge is 64 bits, the corresponding response is 1 bit, and there must be $2^{64}$ CRPs.

There are many kinds of arbiter. In [9], a D latch is used as arbiter (D latch arbiter). Unfortunately, this conventional approach suffers from a couple of shortcomings. An imbalance of 0 and 1 response is caused by setup time of D latch. Further, there is a high possibility of meta stability in D latch. Instead of D latch, D flip-flop (D ff arbiter) is used in [15] to solve the two problems. The arbiter circuit based on a sense amplifier (amplifier arbiter) can also avoid both problems [11]. Another problem is the asymmetry of circuit when an APUF is implemented in a FPGA. This problem leads to a deviation of $\Delta t$. Programmable delay lines were added to arbiter (compensation arbiter) in [11] to compensate the $\Delta t$ deviation. Delay lines can solve the setup time problem of arbiter as well. [11][1][8].

The rest of this paper is organized as follows. Section 2 introduces the methods, the softwares and experimental environment. Section 3 describes the proposed arbiter and presents simulation results. Section 4 concludes the paper.

2. Methods. All the Monte Carlo simulations in this paper is conducted using HSPICE. All the statistical analysis work in this paper is conducted using MATLAB. The theory of Monte Carlo simulations in CAD area is as follow. On the premise that we know the distributions of all the model parameters in circuit, we get the parameter samples by generating random numbers that are in concordance with the distributions. We conduct simulation based on these parameters. This simulation focuses on the generation of random number. A traditional way to generation random numbers that are uniformly distributed is mixing congruential method [13]. For given positive integers A, C and $X_1$, we can get an integer sequence $\{X_i\}$ using recursion formulae Equ.1. And $\{U_i\}$ is the random number sequence in which $U_i=X_i/m$, m is another positive integer. In fact, $\{U_i\}$ are pseudo-random numbers. We need to adjust A,C,$X_1$ and m to maximize the period of $\{U_i\}$.

$$X_k = A \times X_{k-1} + C \quad k = 1, 2, 3,$$

(1)

The width of transistors in circuit are generated by function gauss(220n,0.1,3). The length of transistors in the circuit are generated by function gauss(180n,0.1,3). Other mismatch parameters of transistors are in concordance with the default distributions in model. The temperature is 25 degrees Celsius.
3. Theory.

3.1. Working area of arbiter. We divide APUF into two parts: racing paths and arbiter. The delay difference ($\Delta t$) of racing paths is transmitted to arbiter as the input. So, only when we know what kind of $\Delta t$ does certain racing paths generate, can we decide if our arbiter can work well with this racing paths.

Fig.1 shows the racing paths. The first waveform shows the rising edge that arrives at the beginning. The second and third waveforms show the rising edge propagates in the racing paths. $\Delta T(\text{upper})$ is the propagation delay of upper delay line. $\Delta T(\text{lower})$ is the propagation delay of lower delay line. And $\Delta t = \Delta T(\text{upper}) - \Delta T(\text{lower})$. We need to analyze the $\Delta t$ of different racing paths.

First, we think about the racing paths with only one stage. We conduct 1000 Monte Carlo simulations on a 120nm stage circuit shown in Fig.2 under challenge 0. We feed the same rising edge to $\text{top}_i$ and $\text{bot}_i$ simultaneously. The $\text{top}_{i+1}$ and $\text{bot}_{i+1}$ are the output pins.

Second, we go to the racing paths with 8 stages. We connect 8 stage circuits one by one. We feed the same rising edge to $\text{top}_0$ and $\text{bot}_0$ simultaneously. The $\text{top}_{i+1}$ and $\text{bot}_{i+1}$ of the last stage are the output. The 8 stages circuit has 8 bits challenge. we scan all the 256 possible challenges. For each challenge, we conduct 100 Monte Carlo simulations and calculate a fitting curve.

The statistics and fitting result of the 1 stage circuit simulation are shown in Fig.3. The fitting curve is $y = f(x) = 69.587 \times \exp(-(x - 2.247)/0.003)^2$. In the 8 stages circuit simulation, we get 256 fitting curves as the blue lines shown in Fig.4. Also, we add all these blue lines together and get the red line shown in Fig.3. It is $y = g(x) = 1.6135 \times \exp(-(x + 0.00026609)/0.0197)^2$. From Equ.2, we can assert that in most situations, a one stage circuit will generate a $\Delta t$ in $[-0.007 \text{ns}, +0.007 \text{ns}]$. So, the corresponding arbiter must work accurately when fed by a $\Delta t$ in $[-0.007 \text{ns}, +0.007 \text{ns}]$. The scope $[-0.007 \text{ns}, +0.007 \text{ns}]$ is workspace of the arbiter.

We can jump to a similar conclusion from Equ.3. And obviously, if an arbiter work well under $[-0.04 \text{ns}, 0.04 \text{ns}]$ but has bad performance under $[-0.005 \text{ns}, 0.005 \text{ns}]$, this arbiter can collaborate with only 8 stages racing paths.

![Figure 1. Delay difference](image-url)

\[
\frac{\int_{-0.007}^{+0.007} f(x) \, dx}{\int_{-\infty}^{+\infty} f(x) \, dx} = 0.9959 \tag{2}
\]

\[
\frac{\int_{-0.04}^{+0.04} f(x) \, dx}{\int_{-\infty}^{+\infty} f(x) \, dx} = 0.9959 \tag{3}
\]
3.2. Arbiter circuit design. In [12][3][4][9][14], the setup time problem is emphasized. D latch in Fig. 5 is used here to show this problem. Upper delay line is connected to D pin of D latch, lower delay line is connected to clk pin of D latch. In ideal condition, response is 1 when $\Delta t < 0$, and is 0 when $\Delta t > 0$. However, as a matter of fact, the level variation needs time to take effect. On D pin, this time is called Setup Time ($t_{\text{setup}}$). That is, only when $\Delta t < -t_{\text{setup}}$, can the response turn to 1. This leads to a smaller probability of response 1. So the arbiter is 0-likely. Of course, level variation at clk pin also need Hold Time to take effect. But this hold time will not result in an imbalance of responses, because the initial level of response is 0.
We design a symmetrical arbiter circuit which is shown in Fig.6 to solve the setup time problem. This arbiter consists of two Not-gates (cross coupling inverter) and a reset function. So it is named 2N arbiter. Upper delay line is connected to Upper pin of 2N latch, lower delay line is connected to Lower pin of 2N latch. The arbitration strategy of 2N arbiter is as same as the strategy of D latch arbiter. After arbitration, the Reset pin turns to a low level to activate the reset function. Reset function connects GND to Out and $\overline{\text{Out}}$. Before next arbitration, Reset goes to a high level which will disable the reset function. During arbitration, if the rising edge in upper delay line arrives first, the edge enables the corresponding inverter. This inverter gives Out a high level. And then the rising edge in lower delay line arrives and enables another inverter. This inverter gives $\overline{\text{Out}}$ a low level. Out is the response. Usually, the response depends on the positive or negative of $\Delta t$. But when $|\Delta t|$ is very small, the arbiter becomes a SRAM PUF.

Fig.7 shows the time series of 2N arbiter. The first rising edge in upper delay line arrives at 6.1ns, the second one arrives at 26ns. The first rising edge in lower delay line arrives at 6ns, the second one arrives at 26.1ns. During the first arbitration, the edge in lower delay line arrives first, $\Delta t > 0$. Out signal turn to low level as the result. During the second arbitration, the edge in Upper delay line arrives first. Out signal turn to logic high as the result. Between two arbitrations, Reset...
signal is low to reset $Out$ and $\overline{Out}$ signal. Fig.7 shows that 2N arbiter is able to work continuously. Residual charges from last arbitration have few effect on current arbitration.

We conduct 100 Monte Carlo simulations of 2N arbiter under different $\Delta t$. In each simulation, Upper pin and Lower pin are excited by two rising edges with delay difference $\Delta t$. $Out$ pin is printed as the output. $\Delta t$ scans from -0.050ns to 0.050ns with an interval of 0.001ns. Also, the same simulation is conducted on D latch arbiter as a contrast.

3.3. Results and discussion. The result table is too large to be shown here. To show the result intuitively and to evaluate the arbiters quantitatively, some evaluation metrics are defined based on this result table.

Arbitration probability of an arbiter under a certain $\Delta t$, which is represented symbolically by $PA(\Delta t)$, is the possibility of an arbiter circuit outputting a 1 under a certain $\Delta t$. Ideally, a traditional arbiter outputs a 1 when $\Delta t < 0$ or a 0 when $\Delta t > 0$. This is an analog-digital conversion strategy that converts the analog value $\Delta t$ to the digital value 0 or 1. The arbitration probability of an ideal arbiter, which is represented symbolically by $PS(\Delta t)$, can show the arbitration strategy of this kind of arbiter. In this view, if a circuit is designed to implement an arbitration strategy, this circuit is an instance object of this strategy.

A graph ($PA$ or $PS$ versus $\Delta t$) can show $PA$ or $PS$ under each $\Delta t$. For example, $PA$ of an imaginary arbiter and $PS$ of the corresponding arbitration strategy are
shown as the blue line and red line in Fig.8. The red line should have only discrete values. And the blue line should try to fit the red line.

In the results table, one $\Delta t$ corresponds to 100 voltage values of Out pin. Some values are logic 0, and others are logic 1. PA under this $\Delta t$ is the number of logic 1 divided by 100. In the same way, we can calculate PA under all the $\Delta t$ and further make the graph. The PA graphs of D latch arbiter and 2N arbiter are shown in Fig.9, Fig.10. Obviously, the blue line in Fig.10 fits the red line better. We will analyze the two graphs quantitatively.

![Figure 8. PA of an imaginary arbiter and PS of the corresponding arbitration strategy](image)

![Figure 9. PA of a D latch arbiter and PS of the corresponding arbitration strategy](image)

3.3.1. Correct rate of arbitration. In Fig.11 and Fig.12, these shadows between PA and PS mean the arbiters can’t implement its arbitration strategy perfectly. Correct rate of arbitration (CRA) shows the deviation quantitatively. CRA of an arbiter
under a certain $\Delta t$, which is represented symbolically by $\text{CRA}(\Delta t)$, is the possibility of this arbiter giving a correct output under this $\Delta t$. In experiment, CRA can be calculated by Equ.4. Further, CRA of an arbiter under $[\Delta t_1, \Delta t_2]$ is calculated by Equ.5.

$$\text{CRA}(\Delta t) = \frac{\text{correct output time under } \Delta t}{\text{input time under } \Delta t}$$  \hspace{1cm} (4)

$$\text{CRA}([\Delta t_1, \Delta t_2]) = 1 - \int_{\Delta t_1}^{\Delta t_2} |\text{PA}(\Delta t) - \text{PS}(\Delta t)| \, d\Delta t$$  \hspace{1cm} (5)

CRA represents the ability of an arbiter giving a right arbitration response. CRA gives an answer of how well does an instance object of an arbiter implement its arbitration strategy. For example, CRA of an imaginary arbiter is shown as the shadow area in Fig.11. And $\text{CRA}([-0.05\text{ns}, 0.05\text{ns}]) = 1 - 0.107/0.1 = 0.8925$.

The shadows in Fig.12 and Fig.13 show the CRA of D latch arbiter and 2N arbiter respectively. Obviously, the shadow in Fig.12 is much bigger than the shadow in Fig.13. That means 2N arbiter can implement its arbitration strategy better than D latch arbiter. $\text{CRA}([-0.007\text{ns}, 0.007\text{ns}])$ and $\text{CRA}([-0.040\text{ns}, 0.040\text{ns}])$ of D latch arbiter are 0.5000 and 0.6260 respectively which means D latch arbiter can’t work well with both 1 stage racing paths and 8 stages racing paths. On the contrary, 2N arbiter has a much better performance at least in correct rate.

3.3.2. Instability over time. Ideally, under the same $\Delta t$, in the same environment, an arbiter outputs the same response every time. So, the CRA should be 0 or 1, instead of a number between them. However, in Fig.15 and Fig.16, the CRA($\Delta t$) falls gradually from 1 to 0. That means the arbiter is instable. Instability under $[\Delta t_1, \Delta t_2]$, which is represented symbolically by $\text{Instability}([\Delta t_1, \Delta t_2])$, can evaluate it quantitatively. Instability of the arbiter will affect the stability of APUF. The more CRA close to 1 or 0 (all right or all wrong), the more stable APUF will be. Notice that a stable arbiter doesn’t mean a 100
In experiment, instability can be calculated by Equ.6.

\[
\text{Instability}(\Delta t_1, \Delta t_2) = \int_{\Delta t_1}^{\Delta t_2} |PA(\Delta t) - B(\Delta t)| \, d\Delta t \ast 2 / (|\Delta t_2 - \Delta t_1|)
\]

\[B(\Delta t) = 0 \text{ when } PA(\Delta t) < 0.5\]

\[B(\Delta t) = 1 \text{ when } PA(\Delta t) < 0.5\]

(6)

For example, instability of an imaginary arbiter is shown as the shadow area in Fig.14. And Instability ([ -0.05ns, 0.05ns]) = 0.0097 * 2 / 0.1 = 0.1942

The shadows in Fig.15 and Fig.16 show the instability over time of D latch arbiter and 2N arbiter respectively. The small size of both shadows means that the two arbiters have little inherent instability without regard to environmental changes.
3.3.3. Asymmetry. In Fig. 12, D latch arbiter gives a right response when $\Delta t > 0 \text{ns}$ or $\Delta t < -0.33 \text{ns}$. And when $-0.33 \text{ns} < \Delta t < 0 \text{ns}$, instead of the right response 1, the arbiter gives a 0. Obviously, D latch arbiter responses 0 with a higher possibility than 1. D latch arbiter is asymmetric due to setup time. Asymmetry($|\Delta t|$) which means the difference between $PA(\Delta t)$ and $1 - PA(-\Delta t)$, can evaluate it quantitatively. In experiment, asymmetry can be calculated by Equ.7. Further, Asymmetry of an arbiter under $[|\Delta t_1|,|\Delta t_2|]$ is calculated by Equ.8.

$$\text{Asymmetry}(|\Delta t|) = |APD(\Delta t) + APD(-\Delta t) - 1|$$ (7)

$$\text{Asymmetry}([|\Delta t_1|,|\Delta t_2|]) = \int_{|\Delta t_1|}^{|\Delta t_2|} |1 - PA(\Delta t) - PA(-\Delta t)| \, d\Delta t$$ (8)
An ideal condition is that one line overlaps with another perfectly. *Asymmetry* represents the ability of an arbiter balancing the probability of response 1 and response 0. *Asymmetry* of an arbiter will affect the fairness of APUF. For example, *Asymmetry* of an imaginary arbiter is shown as the shadow area in Fig.17. And *Asymmetry*(\([0 \text{ ns}, 0.05 \text{ ns}]\)) = 0.0016/0.05 = 0.0313.

The shadows in Fig.18 and Fig.19 show the *Asymmetry* of D latch arbiter and 2N arbiter respectively. The area between the two curves in Fig.18 means that setup time problem leads to the imbalance of responses in D latch arbiter. *Asymmetry*(\([0 \text{ ns}, 0.007 \text{ ns}]\)) and *Asymmetry*(\([0 \text{ ns}, 0.040 \text{ ns}]\)) of D latch arbiter are 1.0000 and 0.7480 respectively. Particularly, D latch arbiter always responds a 0 under \(\Delta t\) from -0.007ns to 0.007ns, whatever the challenge is. Obviously, the shadow in Fig.19 is much smaller than the shadow in Fig.18. Also, the *Asymmetry*(\([0 \text{ ns}, 0.007 \text{ ns}]\)) and *Asymmetry*(\([0 \text{ ns}, 0.040 \text{ ns}]\)) of 2N arbiter are 

![Figure 15. Instability of a D latch arbiter](image1)

Instability(\([-0.007 \text{ ns}, 0.007 \text{ ns}]\))=0.0000

Instability(\([-0.040 \text{ ns}, 0.040 \text{ ns}]\))=0.0286

![Figure 16. Instability of a 2N arbiter](image2)

Instability(\([-0.007 \text{ ns}, 0.007 \text{ ns}]\))=0.1928

Instability(\([-0.040 \text{ ns}, 0.040 \text{ ns}]\))=0.0337
0.040ns) declines from 0.7480 to 0.0039 in 2N arbiter. We can draw the conclusion that the setup time problem is mitigated in 2N arbiter.

![Figure 17. Asymmetry of an imaginary arbiter](image1)

Asymmetry([0ns, 0.05ns])=0.0313

![Figure 18. Asymmetry of a D latch arbiter](image2)

Asymmetry([0ns, 0.007ns])=1.0000
Asymmetry([0ns, 0.040ns])=0.7480

4. **Conclusion.** In this paper, we design a new 2N arbiter and conduct Monte Carlo Simulations to prove its usability. A series of new evaluation metrics are defined to evaluate arbiters quantitatively. These metrics show that traditional D latch arbiter has setup time problem and the new 2N arbiter has better performance in correct rate and stability when work with both 1 stage racing paths and 8 stages racing paths. Particularly, 2N arbiter mitigates the setup time problem by reduces the Asymmetry([0ns, 0.040ns]) from 0.7480 to 0.0039.

In this paper, we study the instability over time only. We will concentrate on the instability over temperature and age in the future work. In this paper, our work is
all based on software simulation. We will conduct some FPGA experiments in the future work.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (61471035,61774014).

REFERENCES

[1] K. Fruhashi, M. Shiozaki, A. Fukushima, T. Murayama and T. Fujino, The arbiter-puf with high uniqueness utilizing novel arbiter circuit with delay-time measurement, International Symposium on Circuits and Systems, (2011), 2325–2328.

[2] B. Gassend, D. Clarke, M. Van Dijk and S. Devadas, Silicon physical random functions, Proceedings of the 9th ACM Conference on Computer and Communications Security, (2002), 148–160.

[3] B. Gassend, D. Clarke, M. Van Dijk and S. Devadas, Delay-based circuit authentication and applications, Proceedings of the 2003 ACM Symposium on Applied Computing, (2003), 294–301.

[4] B. Gassend, D. Lim, D. Clarke, M. Van Dijk and S. Devadas, Identification and authentication of integrated circuits, Concurrency and Computation: Practice and Experience, 16 (2004), 1077–1098.

[5] B. Gassend, M. van Dijk, D. Clarke and S. Devadas, Controlled physical random functions, 18th Annual Computer Security Applications Conference, 2002. Proceedings, (2003).

[6] P. F. Hu, H. S. Ning, T. Qiu, H. B. Song, Y. N. Wang and X. X. Yao, Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things, IEEE Internet of Things Journal, 4 (2017), 1143–1155.

[7] 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, 2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525), (2004).

[8] R. X. Li, Improvement of arbiter-based puf, Zhongguo Keji Lunwen Zaixian/ Sciencepaper Online, 6 (2011), 707–710.

[9] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk and S. Devadas, Extracting secret keys from integrated circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13 (2005), 1200–1205.
[10] K. Lofstrom, W.R. Daasch and D. Taylor, *Identification circuit using device mismatch*, 2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056), (2000).

[11] M. Majzoobi, F. Koushanfar and S. Devadas, *FPGA PUF using programmable delay lines*, 2010 IEEE International Workshop on Information Forensics and Security, (2010).

[12] R. Pappu, B. Recht, J. Taylor and N. Gershenfeld, *Physical one-way functions*, Science, 297 (2002), 2026–2030.

[13] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*, Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, 2008.

[14] G. E. Suh and S. Devadas, *Physical unclonable functions for device authentication and secret key generation*, Proceedings of the 44th Annual Design Automation Conference, (2007), 9–14.

[15] J. Q. Zhang, D. W. Gu and F. Y. Hou, Design and analysis of improved arbiter puf, *Computer Engineering*, 3 (2010), 86.

[16] J. L. Zhang, G. Qu, Y. Q. Lv and Q. Zhou, *A survey on silicon pufs and recent advances in ring oscillator pufs*, Journal of Computer Science and Technology, 29 (2014), 664–678.

Received December 2017; revised March 2018.

E-mail address: xuyang.mail@sina.cn
E-mail address: ninghuansheng@ustb.edu.cn
E-mail address: lingfengmao@ustb.edu.cn
E-mail address: lidaqiai@126.com
E-mail address: zhanglijun@suda.edu.cn