Uniqueness Enhancement of PUF Responses Based on the Locations of Random Outputting RS Latches

Dai Yamamoto[1], Kazuo Sakiyama[2], Mitsugu Iwamoto[2], Kazuo Ohta[2], Takao Ochiai[1], Masahiko Takenaka[1] and Kouichi Itoh[1]

[1] FUJITSU LABORATORIES LTD.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588, Japan
{ydai,tochiai,takenaka,kito}@labs.fujitsu.com
[2] The University of Electro-Communications
1-5-1, Chofugaoka, Chofu, Tokyo 182-8585, Japan
{saki@ice,mitsugu@quest.is,ota@ice}.uec.ac.jp

Abstract. Physical Unclonable Functions (PUFs) are expected to represent an important solution for secure ID generation and authentication etc. In general, PUFs are considered to be more secure the larger their output entropy. However, the entropy of conventional PUFs is lower than the output bit length, because some output bits are random numbers, which are regarded as unnecessary for ID generation and discarded. We propose a novel PUF structure based on a Butterfly PUF with multiple RS latches, which generates larger entropy by utilizing location information of the RS latches generating random numbers. More specifically, while conventional PUFs generate binary values (0/1), the proposed PUF generates ternary values (0/1/random) in order to increase entropy. We estimate the entropy of the proposed PUF. According to our experiment with 40 FPGAs, a Butterfly PUF with 128 RS latches can improve entropy from 116 bits to 192.7 bits, this being maximized when the frequency of each ternary value is equal. We also show the appropriate RS latch structure for satisfying this condition, and validate it through an FPGA experiment.

Keywords: PUF, Butterfly PUF, RS latch, Metastable, Random number, FPGA, ID Generation, Authentication

1 Introduction

Secure identification/authentication technology using Integrated Circuit (IC) chips is very important for secure information infrastructure. It is used for anti-counterfeiting devices on medical supplies, prepaid-cards and public ID cards such as passports and driver's licenses. The IC card is a well-known solution for this kind of application. Counterfeiting is prevented by storing a secret key on the IC card and using a secure cryptographic protocol to make the key invisible from outside. In theory, however, the possibility of counterfeiting still remains

if its design is revealed and reproduced by the counterfeiter. Naturally, this is difficult because current IC cards are equipped with several highly-developed tamper-proofing technologies. However, further anti-counterfeiting technologies are desirable to meet future developments in reverse-engineering techniques.

Recently, interest has been focused on Physical Unclonable Functions (PUFs) as a solution [1]. In a PUF, the output value (response) to the input value (challenge) is unique for each individual IC. This uniqueness is provided by the process variations of each individual IC [2] [3]. It is expected that PUFs will represent breakthrough in technology for anti-counterfeiting devices, through its use for ID generation, key generation and authentication protocol, which make cloning impossible even when the design is revealed.

The PUFs on ICs are classified into two categories [4]. One uses the characteristics of memory cells such as SRAM-PUFs [5] [6] and Butterfly PUFs (BPUFs) [7]. SRAM-PUFs are based on the unstable power-up values of SRAM cells on ICs such as ASIC and FPGA. However, a device power-up operation is required for the generation of every response. To counter this drawback, BPUFs are composed of cross-coupled latches which behave similarly to an SRAM cell. The output of the BPUF is triggered by a clock edge signal applied to the latches, without an actual device power-up. The other uses the characteristics of delay variations such as Arbiter PUFs [8], Glitch PUFs [9] and Ring Oscillator (RO) PUFs [10]. Arbiter PUFs have an *arbiter* circuit that generates a response determined by the difference in the signal delay between two paths, which is mixed by a challenge. However, a machine learning attack can predict challenge-response pairs by using a large number of past pairs [11]. The Glitch PUF [9] was proposed to solve this problem of ease of prediction. It generates a response by utilizing glitch waveforms and delay variations between logic gates. Since its response to challenges behaves like a non-linear function, machine learning attacks are prevented. RO PUFs derive entropy from the difference in oscillator frequencies.

Today, PUFs in the former category are some of the most feasible and secure because there have already been implementations of error correcting codes (ECCs) and universal hash functions [12] for randomness extraction optimized for the PUFs, which are needed for Fuzzy Extractors [13]. In addition, BPUFs implemented in ASIC seem to have many advantages over SRAM-PUFs, such as not requiring a power-up operation. This paper therefore focuses on BPUFs, which generate n-bit responses based on n outputs from n RS latches.

The PUFs in both categories need to eliminate the randomness of responses in order to generate stable responses. For example, the Glitch PUF can generate very stable responses because it selects available challenges to output stable responses by a masking process. However, as pointed out by the designers of the Glitch PUF, the masking process causes entropy loss. In conventional PUFs, the outputs of random latches are not used to generate stable responses; however, in this paper we make efficient use of random latches.

The responses from PUFs need to have extremely high *uniqueness*. This paper defines uniqueness as the independence among multiple PUFs of responses to the same challenge. In order to prevent clones of cryptographic hardware, it

is important for manufacturers to make sure that multiple PUFs with the same challenge-response pairs do not exist. However, this is very difficult in terms of cost because there are a huge number of manufactured PUFs and challenge-response pairs. Therefore, one of the most practical solutions is to increase the number and range of responses as much as possible. We must note that a large number of responses are not necessarily equivalent to a high level of entropy in those responses. PUFs that output responses with high entropy are capable of generating completely unpredictable responses. Consequently, the probability of multiple PUFs that output unpredictable responses having the same challenge-response pairs is extremely small. Hence, it is also important for PUFs to increase the entropy of responses so as to have extremely high uniqueness.

In addition, the response needs to have high *reliability*. This paper defines reliability as the consistency of PUF challenge-response pairs for repeated measurements. That is, ideally, a PUF always generates the same response to a given challenge. The BPUF has some RS latches that generate random numbers (i.e. "random latches"). This randomness causes a problem in that the reliability of the response is reduced. This is because the values of the response corresponding to the random latches change every time a response is generated. In the conventional approach - in order to maintain the reliability of responses - the outputs of the random latches are discarded, similar to the masking process in the Glitch PUF, which is a widely known technique for the generation of responses. However, the number of responses becomes lower as the number of random latches increases, which reduces the entropy and uniqueness of responses.

Our Contributions This paper proposes a novel PUF structure for generating high-entropy responses using randomness. Note that our proposed methods can be applied to any PUFs. As an example, our paper focuses on a BPUF with random latches. The use of random latches dramatically increases entropy and uniqueness. Also, the construction can maintain the reliability of responses even if random latches are used for the generation of responses. In specific terms, responses are generated based on the location information of the random latches. The proposed PUF generates approximately 3^n responses with ternary value (0/1/random), which is maximized when the frequency of each ternary value is equal. Here, 3^n is not accurate, but is intuitively easy-to-understand, and so a rigorous discussion is given below. We also propose a suitable RS latch structure to satisfy this equality condition to the maximum extent. We evaluate the performance of the proposed PUF with 40 FPGAs. A BPUF with 128 RS latches based on our RS latch construction increases the average number of random latches from 12 to 32, approaching around 43 (=128/3). The proposed PUF with ternary values improves the number of responses from 2^{116} to 2^{196}. From the actual responses generated by 40 PUFs, the entropy of responses is evaluated as 192.7 bits, which indicates that the proposed PUF has extremely high uniqueness.

Organization of the Paper The rest of the paper is organized as follows. Section 2 gives an outline of the BPUF with RS latches, and the conventional methods for implementing RS latches on FPGAs. Section 3 proposes our original BPUF, which generates responses by using the location information of the random latches. In addition, new methods of implementing RS latches are proposed that maximize the performance of our PUF. Section 4 evaluates the performance of our PUF on an FPGA platform. Finally, in Section 5, we give a summary and comment on future directions.

2 Conventional Methods

2.1 Conv. Mtd (1): Generation of Responses from a BPUF

This paper focus on a BPUF using RS latches. First, we describe the circuit and behavior of an RS latch, shown in Fig. 1. An RS latch can be created from two NAND gates, and is in a stable state with output $(B, C) = (1, 1)$ when input $A = 0$. When input A changes from 0 to 1 (= rising edge), the RS latch temporarily enters a metastable state. It then enters a stable state with either output $(B, C) = (1, 0)$ or $(B, C) = (0, 1)$. Ideally, the probability of transition to either of these states is equal. In fact, however, many RS latches have a high probability of entering one specific state. This is because the drive capabilities of the two NAND gates and the wire length between them are not exactly the same. Hence, the output B from RS latches fall into three patterns: all 0's, all 1's, or a mixture of 0's and 1's (= random number) when a clock signal is applied to input A.

We now describe the BPUF, shown in Fig. 2. Challenges to the BPUF are equivalent to choosing $m(\leq n)$ RS latches from n implemented RS latches. The BPUF can generate m-bit responses corresponding to ${}_n\mathrm{C}_m$ challenges. Here, ${}_n\mathrm{C}_m$ is defined as the number of combinations of n elements taken m at a time. The BPUF in Fig. 2 generates an n-bit response $RES[n-1:0]$ because m is set equal to n. Note that, in order to simplify discussion in this paper, the more significant bits of the response correspond to the outputs of RS latches with bigger latch labels. BPUFs, which generate only a response, can be used for applications such as authentication. For example, a random number S is sent from an authentication server to a PUF as a new challenge, and a response R from the PUF is newly defined by equation $R = \mathrm{H}(S \parallel RES)$. Here, H() indicates a mixing function, such as various hash functions. The value of response R changes depending on the challenge S, so BPUFs provide security when used for this application. The PUF in Fig. 2 has some RS latches that generate random numbers such as LATCH_2 and LATCH_{n-2}. These random numbers cause a problem in that the reliability of the response RES is reduced since its value changes every time it is generated.

There are two widely known conventional approaches to response generation aimed at solving this problem. In the first approach ("conventional method (1-A)"), random latches are not used for the generation of responses. This approach maintains the reliability of responses, but reduces their uniqueness, and

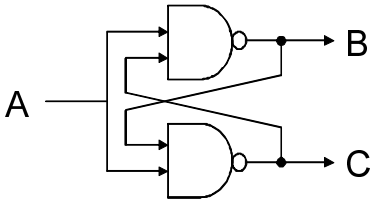


Fig. 1. NAND-based RS latch

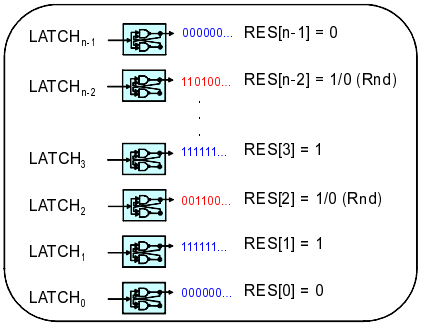


Fig. 2. Butterfly PUF

requires a mechanism to detect random latches. For example, the BPUF with 128 RS latches ($n = 128$) in Fig. 2 has 40 random latches. The bit-length of the responses is reduced from 128 bits to 88 bits, so their entropy and uniqueness are also reduced. Hence, it is necessary to implement extra RS latches in the PUF in accordance with the number of random latches. This PUF is, however, not suitable for embedded systems with limited hardware resources such as smart cards because, while also maintaining the uniqueness of responses, it is necessary for PUFs in embedded systems to have an RS latch area size and peripheral circuit that are as small as possible. In the second approach ("conventional method (1-B)"), ECCs are used to correct the variation in the responses resulting from the random latches. This approach requires larger redundant data for response correction as the number of random latches increases. In addition, it also suffers from the disadvantage of necessitating increased hardware resources and processing time for the ECCs. A BPUF with 128 RS latches generates no more than 2^{128} responses even if ECCs are used. From the above, it can be seen that the first approach, in which random latches are not used for responses, is not suitable. Furthermore, it is not sufficient to use only ECCs, as in the second approach. In Section 3, we propose a method for generating responses based on the locations of random latches. The proposed method maintains the reliability of responses, and dramatically improves their uniqueness.

2.2 Conv. Mtd (2): Implementation of RS Latches on FPGAs

A method for implementing RS latches as a true random number generator on Xilinx FPGAs ("conventional method (2-A)") is proposed in Ref. [14], [15]. Flip-Flops (FFs) are positioned in front of the two NAND gates, as shown in Fig. 3. This minimizes the difference in signal arrival time between the two gates, enabling the RS latch to enter the metastable state more readily and improving the probability of the RS latches outputting random numbers. A Xilinx FPGA consists of a matrix of configurable logic blocks (CLBs). Some kinds of Xilinx FPGA devices have four slices per CLB. A slice includes two pairs of LookUp Tables (LUTs) and FFs. The right and left slices of the CLB are different. The

right slice (SliceL) is available only for logic, while the left one (SliceM) is for both memory and logic. Two types of implementation for an RS latch are reported in Ref. [14]. In one type ("conventional method (2-B1)"), two RS latches are implemented on two CLBs, as shown in Fig. 4(I). In the other ("conventional method (2-B2)"), only one RS latch is implemented on two CLBs, as shown in Fig. 4(II). Both methods implement the NAND gates of an RS latch by using the same kind of slice (SliceL in Fig. 4) on different CLBs. The conventional method (2-B1) uses two CLBs per two RS latches, leading to reasonable circuit efficiency. However, Ref. [14] points out that multiple RS latches which have NAND gates implemented on the same CLB, as shown in Fig. 4(I), have a low probability of outputting random numbers. RS latches based on conventional method (2-B2) have some probability of generating random numbers, but result in low circuit efficiency because an RS latch requires two CLBs. The next section proposes an implementation method that gives the RS latches a high probability of outputting random numbers. In addition, the proposed method gives higher circuit efficiency than in the conventional methods.

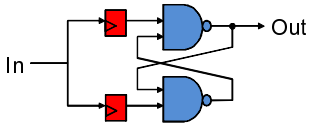
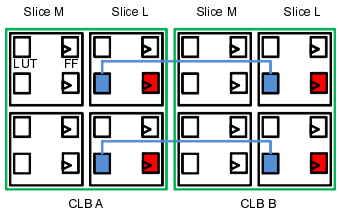
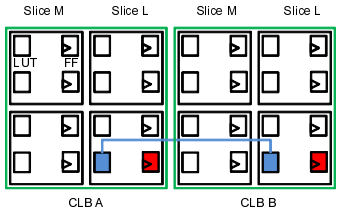


Fig. 3. Conv. mtd (2-A): RS latch circuit [14]



(I) Conventional method (2-B1)



(II) Conventional method (2-B2)

Fig. 4. Implementation of RS latches on Xilinx FPGAs [14]

3 Proposed Methods

3.1 Proposed Mtd (1): Use of the Locations of Random Latches

The conventional BPUF in Fig. 2 generates responses based only on RS latches outputting fixed numbers such as 0's or 1's (i.e. "fixed latches"). Our proposed

BPUF uses the location information of random latch X, rather than the random numbers from the random lathes. If a BPUF with N RS latches has T random latches, then the number of locations of random latches equals to ${}_N\mathrm{C}_T$, which generates the entropy due to random latch locations. Hence, the PUF based on our method utilizes the entropy for uniqueness of responses. However, this kind of BPUF requires complex controls to associate the location of RS latch X with the output number, which leads to a large circuit size. In this paper, we propose a simple and efficient method of solving this problem ("proposed method (1)"). Proposed method (1) regards the three types of output patterns from the RS latches (0's, 1's, and random numbers) as ternary values (00/11/10). Our method can generate responses with much larger patterns than conventional approaches. We describe the details of the proposed method with reference to Fig. 5. When a clock signal is applied to the inputs of the RS latches in our BPUF, they generate three types of outputs: 0's, 1's, and random numbers. The PUF based on our method has new detection circuits - shown in Fig. 6 - located after the RS latches which distinguish these three types. The detection circuit i outputs a 2-bit value (00/11/10) depending on the output of the RS latch i (0's/1's/random numbers). If the output stream of RS latch i includes a transition from 0(1) to 1(0), detection circuit i regards RS latch i as a random latch, and from that point onwards continues outputting the 2-bit value '10' regardless of RS latch i's subsequent output stream. Stated more rigorously, let $S_i[1:0]$ be the 2-bit output of detection circuit i located after RS latch i, and $RES[2n-1:0]$ be the $2n$-bit response of our BPUF. Then

$$RES[2n-1:0] = \sum_{i=0}^{n} \{S_i \cdot 2^{2i}\}. \tag{1}$$

The gate size of the detection circuit shown in Fig. 6 is estimated to be around 28 gates, which is definitely compact enough for embedded systems. Here, we use the equivalencies 1 FF = 12 NAND gate, 1 AND = 1.5 NAND gate, 1 OR = 1.5 NAND gate, and 1 INV = 0.5 NAND gate, introduced in [16]. Naturally, in order to distinguish three types of outputs, CPU-based software approach is able to be used instead of the detection circuit. The reason why we propose the detection circuit as hardware approach is that it is essential when our proposed PUF is implemented on ASIC.

Next, for the PUF based on our proposed method, we theoretically estimate the number of responses. Let N be the number of implemented RS latches, and T be the number of random latches. The number of responses arising from the fixed latches is 2^{N-T}, while the number of responses arising from the random latches is ${}_N\mathrm{C}_T$. Therefore, the number of responses for a given value of T is estimated to be $2^{N-T} \cdot {}_N\mathrm{C}_T$. The PUF based on the proposed method generates ternary values (00/11/10), so the total number of responses is 3^N. This total number is estimated in consideration of all the possible values of T ($0 \leq T \leq N$). However, the value of T is in fact determined by the kind of PUF device and the way in which the RS latches are implemented. Therefore, the PUF generates less than 3^N responses. To be specific, the number of responses for given T corresponds

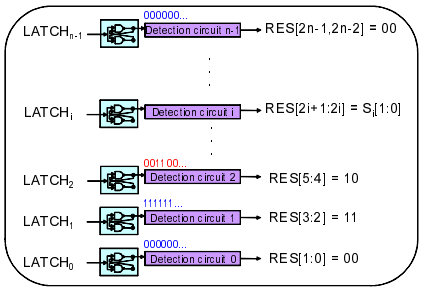


Fig. 5. Proposed method (1)

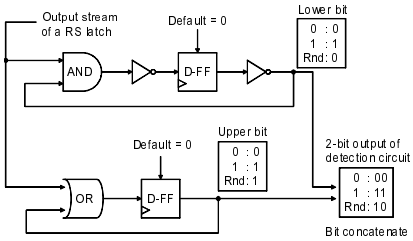


Fig. 6. Proposed detection circuit

to the T-th term of the binomial expansion of $3^N = (2+1)^N$, which is $2^{N-T} \cdot {}_N\mathrm{C}_T$, the same as the above estimate. Figure 7 shows a comparison between the number of responses for the conventional method (1-A) without random latches and the number of responses using our proposed method with various T values and given $N(= 128)$. The conventional method (1-A) generates 2^{N-T} responses, so the number of responses decreases as the number of random latches increases. Even conventional method (1-B), which uses ECCs, generates no more than 2^{128} responses. In contrast, the proposed method (1) dramatically increases the number of responses. The number of responses takes on its maximum value ($\approx 2^{203}$) when the numbers of the three types of RS latches are equal, that is, when T is around 43 ($\approx 128/3$). Hence, the proposed method dramatically improves the uniqueness of responses.

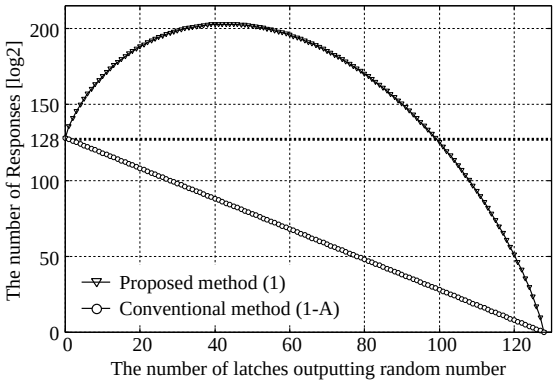


Fig. 7. The number of responses against the number of random latches (Estimate)

3.2 Proposed Mtd (2): Increasing the Number of Random Latches

This section proposes new methods ("proposed methods (2-A) and (2-B)") to give a higher probability of RS latches outputting random numbers than those obtained with the conventional methods in Sect. 2.2. The proposed methods increase the number of random latches to 1/3 of the total number of RS latches, which improves the effectiveness of the proposed method (1).

In proposed method (2-A), a shared FF is positioned in front of two NAND gates, as shown in Fig. 3. This FF sharing between two NAND gates eliminates clock skew in FFs. Consequently, the signal arrival times for the two NAND gates are much closer, allowing the RS latches to become metastable more easily, and increasing the probability of the RS latches outputting random numbers. Proposed method (2-A) also reduces the FF gate size per RS latch by FF sharing.

In proposed method (2-B), one RS latch is implemented on a CLB in a Xilinx FPGA, as shown in Fig. 9. In Ref. [14], an RS latch is implemented on two different CLBs, as described in Sect. 2.2, because FPGA synthesis tools cannot implement two NAND gates of an RS latch on 'different' kinds of Slices (SliceM and SliceL) on the same CLB. To avoid this problem, proposed method (2-B) implements two NAND gates by using the 'same' kind of Slice on the same CLB. Proposed method (2-B) uses only one CLB (two slices) per RS latch, giving high circuit efficiency. In addition, it is anticipated that the probability of RS latches becoming metastable and outputting random numbers would increase since the signal arrival times for the two NAND gates are much closer due to shortening of the wire length between the gates. The concepts behind proposed methods (2-A) and (2-B) can be applied not only to FPGAs but also to ASICs.

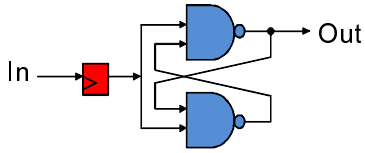


Fig. 8. Prop. mtd (2-A): RS latch circuit

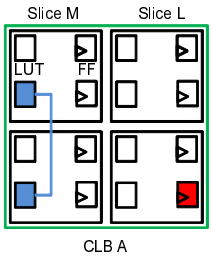


Fig. 9. Prop. mtd (2-B): Implementation of RS latches on Xilinx FPGAs

4 Performance Evaluation

4.1 Experimental Environment

Figure 10 shows our experimental evaluation system, which uses a Spartan-3E starter kit board [17] with a Xilinx FPGA (XC3S500E). A 50-MHz clock signal

generated by an on-board oscillator is applied to a Digital Clock Manager (DCM) primitive, which divides it into a 2.5-MHz clock signal that is applied to 128 RS latches. The output stream from each RS latch is switched by a multiplexer (MUX), and stored into a block RAM. Finally, the raw stream data from all the RS latches are transmitted to the PC through an RS232C port. In our evaluation, a software on the PC detects whether or not the streams contain random numbers rather than this being done with detection circuits. We regard that the detection technique does not influence PUF performance because the latter depends only on the output of the RS latches. We implement 128 RS latches on a 16×8 matrix of FPGA CLBs in accordance with proposed methods (2-A) and (2-B), this being done manually with the FPGA synthesis tools in Xilinx ISE Design Suite 11.1. We regard one FPGA board as four virtual boards, since the RS latches are implemented at four completely different locations in the CLB matrixes for each FPGA. The evaluation uses 10 actual FPGA boards, but in the following discussion, we take the number of FPGA boards to be 40.

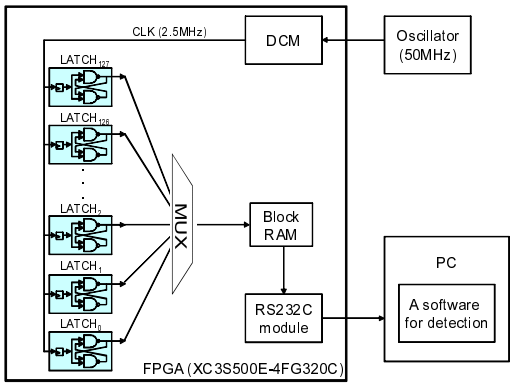


Fig. 10. Experimental evaluation system

4.2 Experimental Results

Reliability and Uniqueness Before we represent an evaluation of the effectiveness of proposed method (1), we show the basic performance of our BPUF, reliability and uniqueness. Our BPUF with 128 RS latches - based on proposed methods (2-A) and (2-B) - gives the results for reliability and uniqueness shown in Fig. 11 and Fig. 12, respectively. In our experiment, the PC is used to measure a 1000-bit output stream from each RS latch. The 2-bit partial response generated by each RS latch is '00(11)' if the 1000-bit bitstream is identically zero (one), or '10' if it includes a transition from 0(1) to 1(0). As a result, our BPUF with 128 RS latches can generate a 256-bit response. The reliability evaluation

generates 40 responses using only a single specific FPGA selected at random. Figure 11 shows a histogram of normalized hamming distance between two arbitrary responses among the 40 responses (i.e. ${}_{40}C_2 = 780$ combinations). The average error rate is approximately 2.4% with a standard deviation of 0.75%, which is much less than the 15% assumed in Ref. [18] for stable responses based on a Fuzzy Extractor with a reasonable size of redundant data. Hence, our PUF gives responses that are of high reliability. Next, the uniqueness evaluation generates a total of 40 responses using all 40 FPGAs (one response per FPGA). Figure 12 shows a histogram of normalized hamming distance between two arbitrary responses among the 40 responses. This evaluation is a general way of showing the extent to which the responses of the chips are different. The difference in the responses of two arbitrary PUFs is approximately 46% with a standard deviation of 3.8%. The ideal difference is 50%, so our PUF gives responses with a high level of uniqueness.

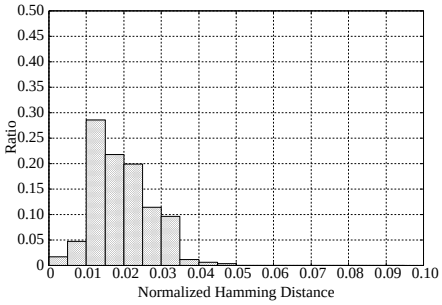


Fig. 11. Reliability

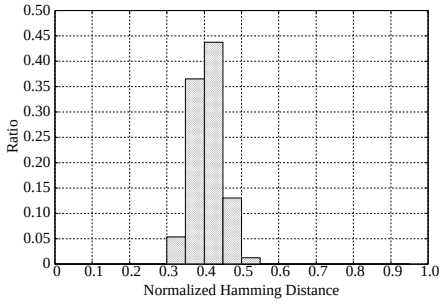


Fig. 12. Uniqueness

Cost Table 10 indicates the gate size and processing time of our PUF evaluation system, shown in Fig. 10. In the FPGA evaluation system, a software on the PC is used instead of detection circuits. Our PUF (not including detection circuits) uses only 5% of the total slices in a FPGA, and the gate size is expected to be very small in ASICs. However, our PUF implemented in ASICs requires 128 detection circuits, and the gate size is estimated to be about 5.4K gates, using the gate equivalencies introduced in [16]. The gate size of our PUF is comparable to that of compact hardware for common key block ciphers such as AES. Hence, our PUF is sufficiently small to be implemented in embedded systems. The gate size can be reduced by a shared detection circuit switched by a multiplexer. The processing time is around 0.4ms, this being the total time taken to generate a response. One way of improving the processing time is to reduce the bitstream length for detection (1000 bits in our experiment). However, too short a length may result in misdetection. For example, RS latches outputting a large number of 0's and very few 1's might be detected not as random, but as fixed latches. This

misdetection leads to loss of reliability, so our PUF makes a tradeoff between reliability and processing time. Our proposed PUF has advantages in terms of low noise because RS latches are allowed to become non-metastable through RS latch clock gating except when generating responses. In addition, our PUF can generate responses at anytime, unlike SRAM PUFs which can only generate them during power activation.

Table 1. Gate size and processing time of our PUF (not including detection circuits)

Gate size	SLICEs used	532/9312
	BRAM16s used	16/20
Processing time		0.4 ms (1000 cycles @ 2.5 MHz)

Effectiveness of Proposed Methods Figure 13, a histogram showing the number of random latches per FPGA, represents an evaluation of the effectiveness of proposed methods (2-A) and (2-B). The results show that the proposed methods increase the number of random latches. This is because these methods allow the RS latches to become readily metastable, and increase their probability of outputting random numbers. In proposed method (1), the number of responses for 128 RS latches takes its maximum value when the number of random latches is around 43. Hence, the proposed methods improve the uniqueness of responses by increasing the number of random latches to as close to 43 as possible.

Table 2 shows the average number of random latches and number of responses for various implementation methods. Here, the number of responses is calculated theoretically based on the average number of random latches and Fig. 7. The number of responses is estimated to be $2^{116}(= 2^{128-12})$ when PUFs implemented by conventional method (2-B1) generate responses without 12 random latches. The PUFs based on proposed method (1) can generate 2^{170} responses using the location information entropy of 12 random latches. Moreover, PUFs based on both proposed methods (1) and (2-B) generate approximately 2^{196} responses with 32 random latches. Our proposed methods therefore dramatically increase the number of responses.

Entropy of Responses Here, we perform a rigorous evaluation of the entropy of responses. The number of responses estimated in Table 2 is based on the assumption that RS latches output 0's, 1's, or random numbers with equal probability. If an RS latch (e.g. LATCH_0) outputs 0's independently of FPGA, then the 2-bit partial response corresponding to that latch must be '00', so the response cannot have a particular value. As a result, the actual number of responses is much smaller than the above estimated number. Table 2 therefore shows the theoretical upper bound on the number of responses. By following the

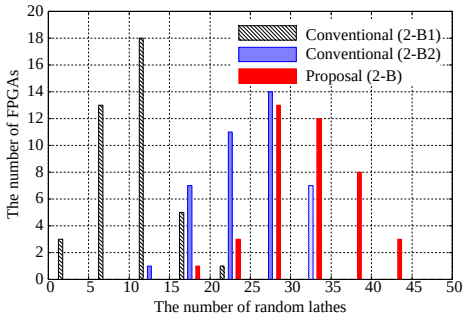


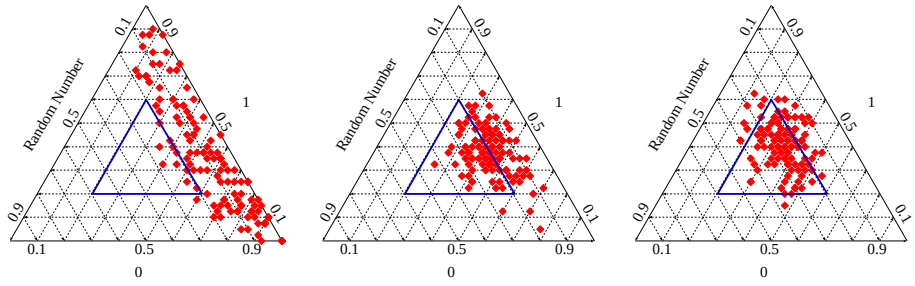
Fig. 13. Histogram for the number of random latches per FPGA

Table 2. The average number of random latches and number of responses

Implementation	Average # of random latches (experimental value)	# of responses (theoretical value)
Conv.(1-A)+Conv.(2-B1)	$\approx 12/128$	$\approx 2^{116}$
Prop.(1)+Conv.(2-B1)	$\approx 12/128$	$\approx 2^{170}$
Prop.(1)+Conv.(2-B2)	$\approx 26/128$	$\approx 2^{192}$
Prop.(1)+Prop.(2-B)	$\approx 32/128$	$\approx 2^{196}$

2 steps below, we rigorously calculate the entropy of responses from our PUFs using the experimental results with 40 FPGAs.

In the first step, we show the ratios of RS latches outputting 0's, 1's, and random numbers, shown in Fig. 14. We explain how to read the figure with the specific example in Fig. 15, as follows. First, the 40 RS latches at the same physical CLB location (e.g. LATCH_0) on the 40 FPGAs are called "a latch group". Hence, in our experiment, there are 128 latch groups corresponding to the range from LATCH_0 to LATCH_{127}. The 40 RS latches labeled as LATCH_0 include 15 latches outputting 0's, 20 outputting 1's, and 5 outputting random numbers. The ratios are therefore 0.375, 0.500 and 0.125, respectively. A plot of LATCH_0 is obtained by relating the ratios to the three sides of a triangle, and 128 plots are obtained, corresponding to the 128 latch groups in Fig. 14. A plot is located at the central point of the triangle if the ratios are equal, which is the ideal. Given the limited number of FPGAs (i.e. 40) in our experiment, it is desirable as a practical criterion that a large proportion of plots are located in the small central triangle illustrated by thick line. If the plot is in the small triangle, the three ratios fall within a range of 0.20 to 0.60. In conventional method (2-B1), it can be seen that all of the RS latches in each latch group have a low probability of outputting random numbers since many of the plots are located on the right side of the triangle. In addition, most RS latches in each latch group have a one-sided probability of outputting 0's or 1's since many of the plots are located throughout the whole of the right side. Conventional



Conv. Mtd (2-B1) (**10** plots in the small triangle)

Conv. Mtd (2-B2) (**65** plots in the small triangle)

Prop. Mtd (2-B) (**93** plots in the small triangle)

Fig. 14. Ratios of RS latches outputting 0's, 1's, or random nums in 128 latch groups

method (2-B2) improves the ratios, making them roughly equal, but requires a large number of CLBs to implement the RS latches shown in Fig. 4. In addition, there are not so many random latches (around 26), so the number of responses is not very large. In contrast, proposed method (2-B) improves the ratios such that they are almost equal since as many as 93 plots are located in the small central triangle. Furthermore, no latch groups have RS latches outputting ternary values at a high (> 0.9) or low (< 0.1) probability. The number of plots in the small triangle is significantly higher than with conventional methods, which implies that the proposed method makes many of the RS latches readily metastable, so that the ratios become almost equal as a favorable side effect. Hence, using the proposed methods, the number of responses is expected to be close to the upper bound shown in Table 2.

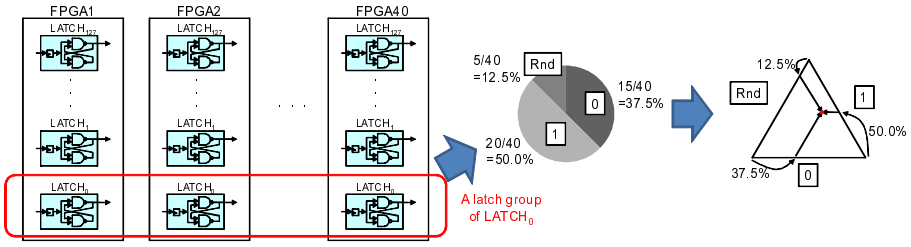


Fig. 15. How to read Fig. 14

In the second step, we rigorously calculate the Shannon entropy of responses based on the ratios discussed in the first step. Let the ratios of the RS latches labeled as $\mathrm{LATCH_n}$ outputting 0's, 1's, or random numbers be $P_n(0)$, $P_n(1)$ and $P_n(R)$, respectively. The Shannon entropy E_n derived from $\mathrm{LATCH_n}$ is defined

as

$$E_n = -P_n(0) \cdot \log_2 P_n(0) - P_n(1) \cdot \log_2 P_n(1) - P_n(R) \cdot \log_2 P_n(R). \quad (2)$$

Hence, the total Shannon entropy derived from LATCH_0 to LATCH_{127} is $\sum_{n=0}^{N-1} E_n$, where $N = 128$. Here, the total entropy can also be given as

$$\sum_{n=0}^{N-1} E_n = \log_2(2^{N-T} \cdot {}_N\mathrm{C}_T) \quad (3)$$

by Stirling's approximation ($\log_2 x! \approx x \log_2 x - x \log_2 e$) under the ideal condition that $P_n(0) = P_n(1) = \frac{1}{2}(1-T/N)$ and $P_n(R) = T/N$ ($0 \leq n \leq N, N = 128$). Equation 3 shows that the total Shannon entropy corresponds to the number of responses estimated in Sect. 3.1. Therefore, in consideration of the ratios of RS latches outputting 0's, 1's, and random numbers in first step, the number of responses can be rigorously calculated on the basis of the Shannon entropy in Eq. 2. Table 3 shows the Shannon entropy for responses. A PUF with 128 RS latches based on conventional method (2-B1) generates $2^{126.6}$ responses even if proposed method (1) is applied. This is because the number of random latches is small, and the ratios are not equal. In contrast, the PUF based on proposed method (2-B) generates $2^{192.7}$ responses, which is almost same as the upper bound in Table 2, and is larger than for PUFs based on conventional methods. Hence, a PUF based on both proposed methods reduces circuit size and dramatically improves the entropy (i.e. uniqueness) of responses.

The entropy per unit area (gate size) of proposed method (1) is expected to be higher than that of conventional methods (1-A) and (1-B). Both proposed and conventional methods (1-A) requires a mechanism to detect random latches, so their area sizes are almost same, while the entropy of proposed method (1) is higher from Table 3. In contrast, conventional method (1-B) does not require the mechanism, so the area size is smaller than proposed method (1). Hence, by implementing more RS latches, the entropy of conventional method (1-B) seems to be higher. In fact, however, conventional method (1-B) needs to correct the variation resulting from all the random latches, which requires larger ECC redundant data for stable responses. In contrast, proposed method (1) regards random numbers as the third stable value, which leads to a reasonable size of redundant data. Therefore, in consideration of the area size for redundant data, the proposed method is expected to generate higher entropy per unit area.

Table 3. Shannon entropy of responses

	Entropy of responses (bits)
Conv.(1-A)+Conv.(2-B1)	97.2
Prop.(1)+Conv.(2-B1)	126.6
Prop.(1)+Conv.(2-B2)	187.7
Prop.(1)+Prop.(2-B)	192.7

Temperature Resistance Figure 16 shows the change in error rate for various temperatures ranging from 0 °C to 85 °C, which is within the rated temperature of the FPGA (XC3S500E-4FG320C). Here, error rate is the ratio of the number of 2-bit partial responses that are different from those at 25 °C. Figure 16 plots the error rates of 40 FPGAs at 0 °C, 25 °C, and 85 °C. The bigger the temperature difference from 25 °C - as the standard temperature - the higher the error rate. The error rate is less than around 15% regardless of temperature, so stable responses are generated based on a Fuzzy Extractor with a reasonable size of redundant data [18].

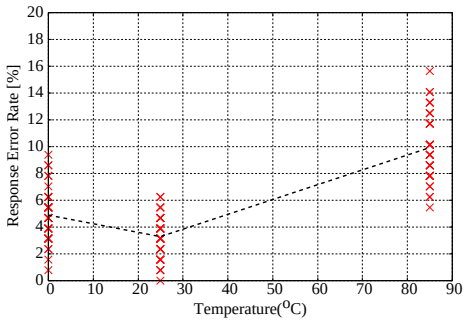


Fig. 16. Error rates against various temperatures

5 Conclusion

This paper proposed a method for generating responses from a BPUF based on the location information of RS latches outputting random numbers. Our proposed detection circuit generates ternary values (00/11/10) in accordance with the three types of output bitstream from RS latches. This increases the number of responses from 2^{128} to around 3^{128} with 128 RS latches, thereby dramatically improving the uniqueness of the responses. In addition, with its small circuit size, the new implementation method increases the number of random latches, and equalizes the ratios of RS latches outputting 0's, 1's, and random numbers, thereby enhancing the effectiveness of the proposed method. According to our experiment with 40 FPGAs, a BPUF with 128 RS latches based on the proposed methods is able to generate responses with 193-bit entropy, which is larger than the 116-bit entropy achieved by conventional methods. The proposed methods can be applied to other PUFs, such as the Arbiter PUF. Unstable (random) outputs from the PUF can be used for generating highly-unique responses without the necessity of selecting available challenges. Future work will include discussion of voltage resistance, performance evaluation on ASIC, and the application of the proposed methods to other kinds of PUFs than BPUFs.

References

1. Ravikanth S. Pappu. Physical one-way functions. PhD thesis, Massachusetts Institute of Technology, March 2001.
2. B.Gassend, D.Clarke, M. van Dijk, and S.Devadas. Silicon physical random functions. In In Proceedings of CCS '02, pages 148–160, 2002.
3. B.Gassend, D.Clarke, D. Lim, M. van Dijk, and S.Devadas. Identification and authentication of integrated circuits. In Concurrency and Computation: Practice and Experiences., pages 1077–1098, 2004.
4. R.Maes and I.Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In Towards Hardware Intrinsic Security: Foundation and Practice, Information Security and Cryptography, pages 3–37. Springer Berlin Heidelberg, 2010.
5. J.Guajardo, S. S. Kumar, G.J.Schrijen, and P.Tuyls. Fpga intrinsic pufs and their use for ip protection. In CHES 2007, pages 63–80, 2007.
6. D. E. Holcomb, W. P. Burleson, and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In Proceedings of the Conference on RFID Security, July 2007.
7. S. S. Kumar, J.Guajardo, R.Maes, G.J.Schrijen, and P.Tuyls. The butterfly puf: Protecting ip on every fpga. In HOST, pages 67–70, 2008.
8. Jae 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 with identification and authentication applications. In In Proceedings of the IEEE VLSI Circuits Symposium, pages 176–179, 2004.
9. D.Suzuki and K.Shimizu. The glitch puf: a new delay-puf architecture exploiting glitch shapes. In CHES 2010, pages 366–382, 2010.
10. G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In Proceedings of DAC '07, pages 9–14, 2007.
11. U.Rührmair, J.Sölter, and F.Sehnke. On the foundations of physical unclonable functions. Cryptology ePrint Archive, Report 2009/277, 2009.
12. J. L. Carter and M. N. Wegman. Universal classes of hash functions. In ACM Symposium on Theory of Computing, 1977.
13. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput., 38:97–139, March 2008.
14. H.Hata and S.Ichikawa. Fpga implementation of metastability-based true random number generator. In IEICE Tech. Rep., RECONF2008-59, Jan. 2009.
15. S.Ichikawa and H.Hata. True random number generator based on metastability of rs latch. In SCIS2009, pages 2F1–5, 2009.
16. L.Batina, J.Lano, N.Mentens, S.BernaÖrs, B.Preneel, and I.Verbauwhede. Energy, performance, area versus security trade-offs for stream ciphers. In The State of the Art of Stream Ciphers, Workshop Record (2004), ECRYPT, pages 302–310, 2004.
17. Spartan-3E starter kit board. http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm.
18. R.Maes, P.Tuyls, and I.Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. In CHES 2009, pages 332–347, 2009.