Efficient SR-Latch PUF

Bilal Habib, Jens-Peter Kaps, Kris Gaj

Electrical and Computer Engineering Department
George Mason University
Fairfax, VA, USA

Email: {bhabib, jkaps, kgaj}@gmu.edu

Abstract. In this paper we present an efficient SR-Latch based PUF design, with two times improvement in area over the state of the art, thus making it very attractive for low-area designs. This PUF is able to reliably generate a 128-bit cryptographic key. The proposed design is compact and the effect of inter-CLB routing is eliminated. The PUF response is generated by quantifying the number of oscillations during the metastability state for preselected latches. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16). The uniqueness measure is 49.24%. In addition the design has been tested at ± 5% of core voltage and also over the rated temperature range [0-85°C]. The reliability at +5% of nominal voltage is 99.18%, while at -5% of nominal voltage it is 97.54%. We also propose a novel area-efficient error correcting scheme that assures that a key generated in the field, at the extreme values of voltage and temperature supported by the commercial-grade Spartan-6 FPGAs, is the same as the key generated during enrollment at nominal operating conditions.

Keywords: Physical Unclonable Functions, SR-Latch, Metastability, FPGAs.

1 Introduction

Recent research has led to an increased interest in security measures, especially in solutions that are physically unique and unclonable. In this regard, different structures of Physical Unclonable Functions (PUFs) have been developed and investigated to efficiently meet the requirements of these solutions. PUFs are physical primitives that produce unclonable and device-specific measurements of silicon Integrated Circuits (ICs). These measurements are then processed to generate either responses in challenge-response schemes or secure keys for cryptographic functions. Manufacturing process variations give physical uniqueness, but many physical unclonable functions (PUFs) are noisy and exhibit low circuit efficiency. Therefore, while designing new PUF structures, we need to focus on the efficiency and reliability besides the unclonability and uniqueness measures.

PUF structure is incorporated in the silicon devices for targeting two major applications. First one is the identification of silicon devices and another one is secure key

generation for cryptographic functions. In this paper we are targeting the secure key generation application using SR-latch PUF.

In Section 2, we describe the previous work for a better understanding of our study on PUF. Section 3 explains the design methodology. In Section 4, we explain the bit-string generation. Results and analysis are covered in Section 5. The conclusion and future study are described in Section 6.

2 Related Work

Since 2002, silicon based PUFs have been extensively investigated. There are two categories of silicon based PUF circuits: Delay based PUFs and Memory based PUFs. Delay based PUFs include Arbiter PUF [1, 2, 3], Ring-Oscillator (RO) PUF [4, 9], and S-ArbRO PUF [15]. In [14], a PUF based on programmable delay lines is presented.

Another category of silicon PUF is based on memory. It includes SRAM PUF [5], Butterfly PUF [7], and Latch PUF [6]. In [10], the concept of transient effect ring oscillator (TERO) is presented. A true random number generator (TRNG) is developed by counting the oscillations of elements during metastability. The least significant bit of that count is selected as a random bit. It is important to mention that the design proposed in [10] is implemented on Spartan-3 devices which are a 90nm technology while our design is tested on Spartan-6 which is a 45nm technology. The source of entropy is dependent on many factors, like the technology used, design implemented and testing methodology. The TERO approach is further developed in [13], where an element with PUF capability is presented. In [17], PUF design is developed which is based on (TERO) cells. The randomness is harvested by measuring the metastable oscillator counts. Final bits are generated by averaging the oscillation counts and then reading the most significant two (or four) bits of an eight bit counter for each latch. In order to assure the reproducibility of these bits, each latch count is measured 2^{18} times.

In [16], SR-latch based PUF is developed; it has been implemented on Spartan-3 and Spartan-6 devices. In Spartan-6 based design, 128 SR-latches have been implemented. Each latch is configured by using two neighboring CLBs in each column. Inside each CLB a Look up Table (LUT) and a Flip Flop (FF) are used. PUF response is determined by the final state of the latch. All the latches are excited by a 2.5MHz clock signal. For each latch two bits are contributed towards the PUF response. If the final state of the latch is logic ‘0’ in 1000 repetitions of the experiment, the corresponding response bits are ‘00’. If the final state is logic ‘1’ in 1000 repetitions of the experiment, the corresponding response bits are ‘11’. If the final state changes at least once during 1000 repetitions of the experiment, this state is declared random, and encoded as '10'. A detection circuit determines all three cases and generates the response bits in each case. In our proposed design, we tried to improve the circuit efficiency of this design. In addition, our source of entropy is based on the exact number of oscillations at the output of an SR-latch during the metastable state, rather than a final state of each latch, as in [16]. Because of the encoding method, in [16] the goal is to increase the number of random latches, while in our work, we decrease the number of random latches. We include only the most stable latches, i.e., latches generating

consistently the same number of oscillations in the metastable state, in the bit generation step during enrollment. On top of that, the method of [16] does not differentiate between the behavior of latches that generate no oscillations at the output from the behavior of latches that generate an even number of oscillations. This distinction is clearly made in our scheme. Furthermore, the method of [16] is prone to the influence of neighboring logic. We diminished this influence by prohibiting the tool from assigning any resources of the latch CLB to any external logic.

3 Design Methodology

In our design, an SR-latch is made from two LUTs configured as a NAND gate each. Additionally, two flip-flops are used in this latch to reduce the clock skew. Initially, a latch is forced into a metastable state by applying a rising-edge at a 'ctrl' signal, as shown in Fig. 1. Transitory oscillations in the loop start if the following two conditions are fulfilled [10]: A) the circuit must have a positive feedback and B) The RC time constant (defined by the parasitic resistance and capacitance) must be shorter than the total delay of all logic elements involved in the loop. In our design the delay of a loop is equal to the propagation delay of a single LUT. In Spartan 6 FPGAs, this delay is equal to 0.21ns [20]. Adding more elements to the loop increases the propagation delay. The longer loops will be more strongly affected by the routing delays of FPGA fabric; therefore we kept the loop as small as possible.

During the metastable state, the SR-latch oscillates. An eight-bit counter is used to count these oscillations. Once the metastable state is over, the latch stops oscillating, and the counter value is stored into the block RAM (BRAM). Before applying the 'ctrl' signal, the two flip-flops (FF) are reset. It is done to ensure that latches always start oscillations with the same initial state. This reset functionality is not incorporated in either [13] or [17]. Additionally, both [13] and [17] are based on a loop made of AND gates and inverters. Our loop is based only on NAND gates. Since in an FPGA implementation, an additional gate requires an extra LUT, therefore we chose NAND gate to keep the loop as short as possible. This small modification results in the reduction of the propagation delay. It needs to be mentioned that by keeping the loop small has two major advantages. First, we can place four SR-latches inside a single CLB; secondly, propagation delays are minimally affected by routing delays. This way, the randomness due to process variations is the dominant factor, while in a longer loop the routing delays become the dominant factor in the propagation delay [8].

Once the process of the characterization is over for all the latches, the data from the BRAM is read-out to the PC via Enhanced Parallel Port (EPP) protocol as shown in Fig. 2. In our design, a 9-bit multiplexer address line selects a particular latch. This latch is then excited by applying a low to high transition at the 'ctrl' signal, as shown in Fig. 2. The eight-bit counter, available at the output of the multiplexer, counts the number of oscillations during the metastable state. These values are then stored in the neighboring BRAM. The bit generation and analysis are done during post-processing. It must be mentioned that only one latch is characterized at any given time. This is done to prevent any correlation between the neighboring latches and also to save the FPGA logic resources.

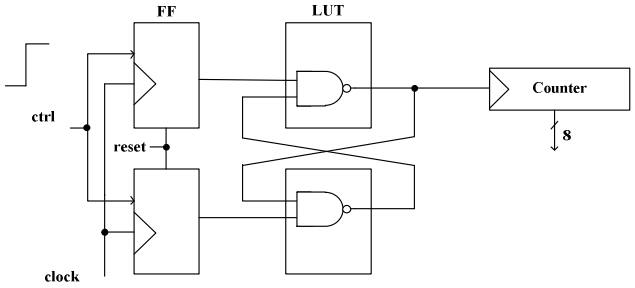

Fig. 1. A single SR-Latch design.

Therefore, only one counter is used to measure the latch-counts during the metastable state. In addition, all the control signals are provided by the FSM. 512 latches are implemented, which requires 128 CLBs. The prototype design requires BRAM and EPP, because we want to analyze all the latch counts. However, in the final product, EPP can be replaced by a different interface and the size of the required BRAM will be reduced. The placement of latches is constrained by the slice location attribute. All the latches are placed in a rectangular matrix of CLBs. The dimensions of this matrix are 16x8 as shown in Fig. 3.

We implemented our design on Spartan-6 (XC6SLX16) device. Four latches are implemented inside a single CLB in our design. These four latches (L1, L2, L3 and L4) are shown in Fig. 4, each with a different color scheme. This design is developed to eliminate the inter-CLB routing. We believe that due to the capacitance of long wires, the variation due to routing delays become significant, as explained in [8].

As evident from the above figure, each latch consists of two LUTs and two FF. All the LUTs inside the CLB are utilized in the implementation of latches, thus, achieving a hardware efficiency of 100% in terms of LUTs. By comparison, the design proposed in Fig. 12 in [16], has 12.5% LUT utilization because, only two LUTs are utilized from the two adjacent CLBs. In addition, the circuit efficiency for FF is 50%, while it is only 6.25% in [16].

4 Bit Generation

Each latch is sampled 100 times and the corresponding latch count values are stored in the BRAM. Once all the latches are characterized, we select highly stable latches and ignore the remaining ones. In our method, a latch is defined to be stable if the latch count value remains the same for all 100 samples. The value 100 is chosen due to the fact that it is easier to interpret it as a percentage of the total. This number is in fact a trade-off between reliability, execution time and the storage resources in BRAM. During enrollment, we store for each latch, a latch number, a corresponding bit showing whether this latch is stable, and the latch count value (i.e., we store: {Latch #, stable, count}).

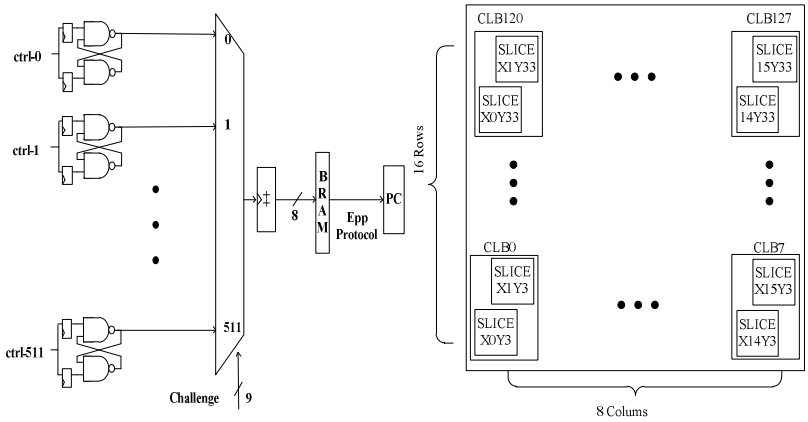

Fig. 2. SR-latch PUF design.

Fig. 3. Layout on FPGA.

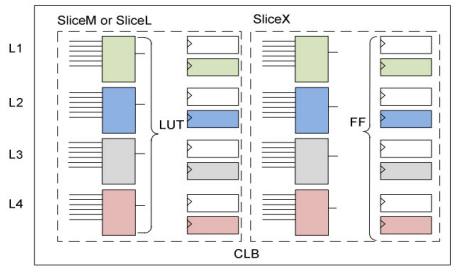

Fig. 4. Our proposed design: Implementation of 4 SR-latches per CLB.

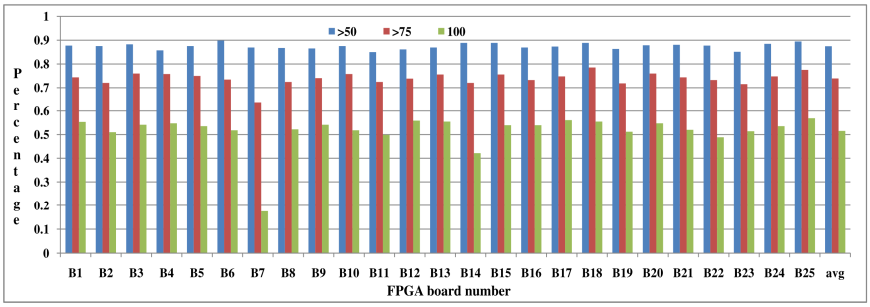

Fig. 5. Proposed design: Percentage of stable latches per board at 1.2V and 25ºC. This result corresponds to the design from Figure 4.

For bit generation during enrollment, only the stable latch count values are considered. In this method the count values for L+1 latches are used. These values are used to generate the L bit PUF response. We number stable latches in a snake-like fashion:

L1, L2, L3, ... LL+1, and then we do the comparisons of neighboring latches [L1, L2], [L2, L3], [L3, L4], ...,[LL,LL+1]. The snake-like comparison is adopted to mitigate the effect of systematic variations [9]. A binary response bit '1' is generated if the count value of latch L_i is greater than the count value of latch L_{i+1}; otherwise it is '0'. In the field, the count value of stable latches is recalculated by sampling each latch 100 times. We get 100 count values between 0 and 255, e.g., {127, 127, 128, 127, 127, 127... 128, 127}. For further calculations, we use the number that appears the largest number of times, e.g., 127 in the example above.

In Fig. 5, the percentage of stable latches per device is shown. On the average, 87% of latches (0.87*512 = 445) in all devices repeat the latch count at least 50 times out of 100 samples during enrollment (i.e., have stability mode "> 50"). This percentage drops to 51.4% when the latch counts are identical 100 times out of 100 samples (i.e., have stability mode "100"). As evident from Fig. 5, the FPGA boards B7, B14 and B22 have the smallest number of stable latches. In such cases, we consider as stable even latches that repeat the count value at least 90 times. Now the question arises, how many latches we should include in the bit generation process. It depends on the reliability of PUF response. In this work, we tested our boards at +5% of nominal voltage and -5 % of nominal voltage. It needs to be mentioned that on our Nexys-3 FPGA boards, the nominal voltage is equal to 1.2V. We believe that the high number of stable latches is achieved due to the fact that external logic has minimal affect on the latch operation. This is accomplished by using all the LUTs of Latch-CLBs for implementing SR latches.

It should be mentioned, that our method for a PUF ID generation is completely different than the method used in [17]. In [17], the most significant bits of a count value for a single TERO loop are used. In our design, all bits of two count values, obtained using two neighboring stable SR latches, are compared with each other to generate a single bit of a PUF ID. As a result, our design allows obtaining the same (or higher) reliability using much fewer repetitions of the count measurement (2^{18} in [17] vs. $100x100 < 2^{14}$ in our design). Because of that, our PUF has a shorter response time than the PUF described in [17].

To prove that a latch is not biased in our design, we analyzed the count value of all the latches and found that 50.12% of them are odd while the remaining ones are even. It proves that the symmetry of latch is not affecting the count value. In addition, it also proves that the difference in the length of two nets connecting the LUTs with each other is insignificant. We need to mention here that our final bit response from each pair of latches is dependent on the latch count values and not on the final state of any particular latch. Therefore, the bit response from a pair of latches will always be the same as long as the differences of the two count values do not change the sign.

5 Results

We compare our results with [16]. This comparison is summarized in Tables 1 and 2. The uniqueness is calculated using the following equation:

$$\text{Uniqueness} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{HD(R_i, R_j)}{L} \times 100\% \qquad (1)$$

Table 1. Details of dataset.

	This work	Yamamoto et al. [16]
No. of Chips (N)	25	20
PUF per Chip	1	2
Samples (T)	100	100
ID size (L) bits	256	256
SR-Latches (M)	512	128
FPGA family (Device used)	Spartan-6(XC6SLX16-3CSG324)	Spartan-6(XC6SLX16-2CSG324)

Similarly, reliability is calculated according to the following equations:

$$HD_{INTRA,i} = \frac{1}{T}\sum_{t=1}^{T}\frac{HD(R_i, R_{i,t})}{L} \times 100\% \quad (2)$$

$$Reliability_i = 100\% - HD_{INTRA,i} \quad (3)$$

$$\text{Average Reliability} = \frac{1}{P}\sum_{j=1}^{P} Reliability_i \quad (4)$$

Table 2. Comparison with Yamamoto et al [16].

	This work	[16]	Ideal
Uniqueness	49.24%	49%	50%
Reliability @ 1.14v	**97.54%**	94.70%	100%
Reliability @ 1.20v	**99.5%**	99.14%	100%
Reliability @ 1.26v	**99.18%**	95.20%	100%

Ri = Response of chip i
Rj = Response of chip j
HD= Hamming distance
T = Total number of samples /ID
t = Index of a sample ($1 \leq t \leq T$)

Reliability shown in Table 2 is the average reliability of five random boards from a set of 25 boards. Average Reliability of P chips can be calculated using equation (4). Equations 1, 2 and 3 are based on the definitions of PUF quantitative metrics proposed in [18]. It needs to mention that Uniqueness shown in Table 2 is calculated for 25 FPGA devices.

5.1 Uniqueness

In Fig. 6, the normalized inter-chip Hamming distance, (HD (Ri, Rj)/L)*100% is shown.

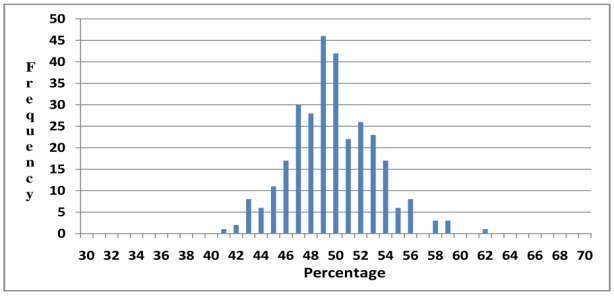

Fig. 6. Normalized inter-chip Hamming distance (Mean=49.24%).

The mean is 49.24%, while the standard deviation is 3.36%. This data is generated from 25 FPGA boards at 25°C and 1.2V supplied as the core voltage. The total number of combinations (i.e., the total number of board pairs {i,j}) is $\binom{25}{2}$= 300. The y-axis (denoted frequency) shows the total number of times a given normalized inter-chip Hamming distance was obtained.

5.2 Reliability

In Table 3, the information on bit flips for five boards is shown. It is clear from this table that the worst case is 13 bit flips. The Normalized intra-chip Hamming distance is defined as (HD (Ri, R'i,t)/L)*100%.

Table 3. Voltage vs. Intra-chip Hamming Distance.

Board No.	No. of stable latches at 1.2V	Bit flips at 1.26V	Bit flips at 1.14V	Maximum HD(R_i,$R'_{i,t}$)	Worse case HD
B2	260	1	4	4	**13**
B4	280	1	**13**	**13**	
B6	264	1	6	6	
B16	275	5	1	5	
B23	262	3	9	9	

We tested the boards at 0°C and 85°C. Table 4 shows the results for 10 boards. From Table 4, it is evident that for reliability the effect of 85°C is always worse than the effect of 0°C. Overall the worse case Hamming distance is 16. We also tested the five FPGA boards at 1.14V and four different temperatures, as shown in Fig. 7.

Table 4. Temperature vs. Intra-chip Hamming Distance.

Board No.	Bit length	Bit flips at 0°C	Bit flips at 85°C	Maximum HD(R_i, $R'_{i,t}$)	Worse case HD
B1	256	4	11	11	**16**
B2	256	7	12	12	
B3	256	2	12	12	
B4	256	0	7	7	
B5	256	3	13	13	
B6	256	2	7	7	
B7	256	3	10	10	
B8	256	7	**16**	**16**	
B9	256	8	8	8	
B10	256	5	8	8	

5.3 Reliability vs. Error Correction

For error correction, we propose a novel method, inspired by the designs described in [19]. This method is based on the use of BCH code. It does the error correc-

tion by removing the noise from PUF response in the field. This method consists of two procedures: Generation and Reproduction. Generation is carried out at the room temperature and nominal voltage, while Reproduction is carried out in the field.

During the generation process, Secure Sketch (SS) is applied to PUF output w, as shown in Fig. 8. The second input to SS is the key K, generated using a True Random Number Generator, RNG1. The output of SS, denoted by s, is stored as helper data in the database. During the reproduction process, the helper data is used to regenerate the key K from a noisy PUF response w'. BCH decoder is used to regenerate the 131 bit key as shown in the figure. We propose to use BCH with the following parameters: (n=255, k=131, t=18) code. The meaning of these parameters is as follows: n=255 is the output block size, k=131 is the input block size (in our case, the size of the key to be encoded), and t=18 is the number of errors that can be corrected by this code. We chose these parameters because the code with these parameters can easily correct the worse case errors shown in Tables 3 and 4, and Fig. 7. Please note that in our scheme, the key K is not a function of the PUF response during the Generation process, but becomes a function of the PUF response during the Reproduction process in the field. This feature differentiates our scheme from two methods presented in [19].

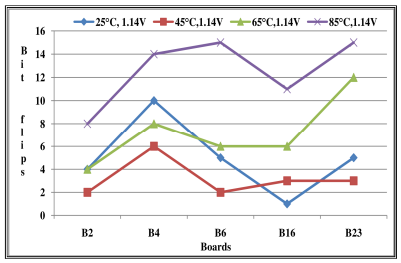

Fig. 7. Bit flips at 1.14V [25°C-85°C].

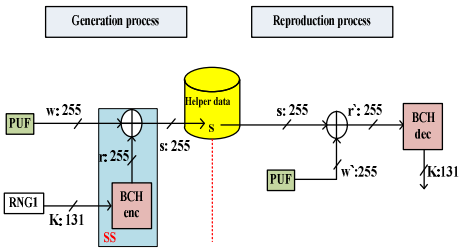

Fig. 8. Error correction scheme.

5.4 Entropy Analysis

Actual entropy of a PUF is a function of complex physical processes. Therefore, it is close to impossible to calculate the actual entropy of a PUF response. Normally, only the estimated upper bounds on the underlying entropy can be calculated. These bounds can be calculated using at least the following two methods.

1. Based on the analysis of single bits
2. Based on the analysis of pairs of bits

Both these methods have been explained in [12]. The first method assumes that an adversary knows a bias for each position of a PUF response. In this method every bit position in a PUF response vector will have its own bias. An adversary knowing these individual bit-dependent biases can make a more accurate prediction by guessing

individual bits in favor of these biases. This upper bound, called the bit-dependent bias entropy bound, can be calculated by using,

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) \quad (5)$$

In equation (5), h(pi) is the binary entropy function, it has been calculated for n = 256 bit positions. We calculated bit-dependent bias entropy bound based on PUF responses of 25 FPGAs. This entropy bound appeared to be equal to 0.959 or 95.9%. This result implies that the 256-bit response contains a maximum of 245 bits of entropy.

The second method is based on the analysis of a pair of PUF response bits. This method assumes that an adversary knows pairwise joint distributions for pairs of consecutive bits i and i+1.This bound is tighter than the bound given by (5). It is called pairwise joint distribution entropy bound. It can be calculated using equations (6) and (7).

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) - \sum_{i=1}^{n-1} I(Y_i, Y_{i+1}) \quad (6)$$

Where,

$$\mathrm{I}((Y_1, Y_2) = \sum_{y1 \in \mathrm{y}1} \sum_{y2 \in \mathrm{y}2} p(y1, y2).\, log_2 \frac{p(y1,y2)}{p(y1)p(y2)} \quad (7)$$

In equation (6), h(pi) is the binary entropy function, while $I(Y_i,Y_{i+1})$ is the mutual information between two random variables Y_i and Y_{i+1}. The mutual information between two random variables is a measure for the amount of information which is shared by both variables. We estimate the pairwise joint distributions of all possible pairs of the considered response bits, by counting the occurrences of each of the four possible pairs ('00', '01', '10', and'11') in the 256 bit response of all 25 devices. We found that for n = 256; the pairwise joint distribution entropy bound is equal to 0.866 or 86.6%. Therefore, a 256-bit response contains approximately 221 bits of pairwise entropy. Finally, we want to add that the entropy of the key is equal to PUF_entropy*131 in our top-level key generation scheme with error-correction code, shown in Fig. 8.

5.5 Cost

It has been stated in [16] that the unused LUTs in each CLB can be used by the tool for other purposes. Therefore it is claimed that the circuit efficiency is still higher. However, we believe that neighboring logic, routed through the CLB, adversely affects the PUF response bits. To prove this claim we implemented ring-oscillators inside the latch-CLB for the design proposed in Fig. 12 in [16]. The result is listed in Table 5. We believe this change can become significant at different voltage.

Table 5. Effect of external signals on SR-latch.

	Without Ring oscillators [16]	With Ring oscillator
Reliability @ 1.2v	99%	92%

Therefore it is highly recommended to prohibit the external signals from being routed through the latch-CLBs. In our design, the external signals cannot be routed through any CLBs designated to be used for SR-latches. We also prohibit the tool from using any resources inside the latch-CLBs for external logic. Therefore, the effect of external signal on the latch performance is eliminated.
As shown in Fig. 1, the latch requires only two LUT input pins. Thus we connect the remaining four input pins of each LUT to logic '0', and lock them. It must be mentioned that we implement our PUF only on even rows of CLBs. This is done to leave one set of rows for additional logic to be implemented by the tool. This logic includes multiplexer, decoder, registers, and counters. We would like to emphasize that the latch in our design is compact and does not share a switch-box with any other external signal (each CLB is associated with a single switch-box). Based on this discussion, the design proposed in [16] for Spartan-6 FPGAs is two times more expensive than the one we are proposing in this work. Table 6 lists the FPGA resources used by both designs.

Table 6. Comparison with [16].

	This work	**[16]**
Total latches configured	512	128
Total CLBs used for PUF (latches only)	128	256
Response bit length	256	256
Latch/#CLB	4	0.5
Response bits/#CLB	2	1
Response bits entropy	221	167.9
Response bits entropy/#CLB†	1.72	0.65
Multiplexer size (CLBs)	16	4
Extra logic (CLBs)*	18	N/A

† Latches only. * Registers, counters, control logic.

5.6 Characterization Time

We record 100 samples of data from each latch. The delay between any two samples is 10,000 clock cycles. This figure comes from the fact that some latches oscillate longer during metastable state. The on-board clock frequency is 100MHz. As a result, the frequency of the ctrl signal is set to 10 kHz. For comparison, in [13], the ctrl signal's frequency is equal to 75 kHz. The total characterization time for each FPGA is equal to (512 x 100 x 10,000)/ (100 MHz) = 5.12 sec.

6 Conclusion and Future Work

We presented a highly reliable SR-latch PUF in this work. The latch is designed to keep the effect of routing at minimum and extract the randomness at the same time. The design is two times more efficient from the circuit efficiency point of view. The PUF responses exhibit high resistance to temperature and voltage variations. The uniqueness is close to the ideal value. We tested the design on a set of 25 FPGA devices and the results were very promising. We plan to incorporate the bit-generation step into the circuit, thus eliminating the requirement for post-processing. In addition, we plan to implement the on-chip error correction scheme.

References.

1. B. Gassend, D. Clarke, M. van Dijk, S. Devadas. “Controlled Physical Random Functions,” in *Proc. ACSAC*, 2002, p. 149-160.
2. J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk, S. Devadas. “A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Application,” in *Proc. Symposium on VLSI Circuits*, 2004, pp. 176–159.
3. D. Lim. “Extracting Secret Keys from Integrated Circuits.” Master’s thesis, MIT, MA, USA, 2004.
4. G.E. Suh, S. Devadas. “Physical unclonable functions for device authentication and secret key g*eneration,” in Proc. DAC, 2007,* pp. 9-14.
5. J. Guajardo, S. S. Kumar, G. Schrijen, and P.Tuyls. “FPGA intrinsic PUFs and their use for IP protection,” in *Proc. CHES*, 2007, pp. 63–80.
6. Y. Su, J. Holleman, B. Otis. “A 1.6pJ/bit 96% stable chip-ID generating circuit using process variations,” in *Proc. ISSCC,* 2007, pp. 406–611.
7. S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. “Extended abstract: The butterfly PUF protecting IP on every FPGA,” in *Proc. HOST,* 2008, pp. 67–70.
8. S. Morozov, A. Maiti, and P. Schaumont. “An analysis of delay based PUF implementations on FPGA,” in *Proc. ARC* 2010, pp. 382–387.
9. A. Maiti and P. Schaumont. “Improved Ring oscillator PUF: An FPGA Friendly Secure Primitive.” *Journal of Cryptology*, vol 24, pp. 375-397, Oct. 2010.
10. M. Varchola, and M. Drutarovsky “New High Entropy Element for FPGA Based True Random Number Generators,” in *Proc. CHES,* 2010, pp. 351-365.
11. G. Hospodar, R. Maes, I. Verbauwhede. “Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability,” in *Proc. WIFS* 2012: pp. 37-*42.*
12. *R. Mae*s. “Physically Unclonable Functions: Constructions, Properties and Applications”. PhD Thesis, KatholiekeUniversiteit Leuven (2012).
13. M. Varchola, M. Drutarov*sky and V.* Fischer. “New Universal Element with Integrated PUF and TRNG Capability,” in *Proc* ReConFig, 2013, pp. 1-6.
14. B. Habib, K. Gaj and J. Kaps. “FPGA PUF Based on Programmable LUT Delays,” in *Proc. DSD*, 2013, pp. 696-704.
15. D. Ganta and L. Nazhandali. “Easy-to-build Arbiter Physical Unclonable Function with enhanced challenge/response set,” in *Proc ISQED* , 2013, pp. 733-738.
16. D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. Takenaka and K. Itoh. “Variety enhancement of PUF responses using the locations of random outputting RS latches.” Journal of Cryptographic Engineering, vol. 3, pp. 197-211, November 2013.
17. L. Bossuet , X. Ngo, Z. Cherif and V. Fischer. “A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon,” in *Proc. TETC* . 2013, pp. 30-36.
18. A. Maiti, V. Gunreddy, and P. Schaumont. “A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions,” in *Embedded Systems Design with FPGAs*, Springer, 2013, pp. 245-267.
19. H. Kang, Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura. "Cryptographic Key Generation from PUF Data Using Efficient Fuzzy Extractors," in *Proc. ICACT*, 2014, pp.23–26.
20. http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf