FPGA Based Arbiter Physical Unclonable Function Implementation with Reduced Hardware Overhead

Alexander A. Ivaniuk and Siarhei S. Zalivaka(B)

Belarusian State University of Informatics and Radioelectronics, 220013 Minsk, Belarus
{ivaniuk,zalivako}@bsuir.by

Abstract. The existing implementations of the arbiter physical unclonable function (PUF) are based on synthesis of configurable symmetric paths, each link of which is a pair of two-input multiplexers providing two configurations of test signal translation: straight and exchange. To build a single link on FPGA, it is necessary to use two built-in LUT blocks, providing the implementation of two multiplexers, and the hardware resources of the LUT blocks are not fully utilized. The paper presents a new architecture of symmetric paths of the arbiter PUF, providing efficient use of the hardware resources of LUT blocks for various Xilinx Artix-7 FPGA family.

Keywords: Physical Unclonable Function · Arbiter · FPGA · LUT · Symmetrical path

1 Introduction

Protection of digital devices against unauthorized use, copying and modification can be achieved by various methods, algorithms and technical means (e.g. encryption, watermarking and fingerprinting, active and passive metering, formal verification, etc.). Relatively novel scientific area named physical cryptography can be highlighted among many mentioned above methods. This direction is mainly based on so called Physical Unclonable Functions (PUFs) [1]. The main idea of PUFs is in extraction of unique physical characteristics from a fabricated digital device. Manufacturing process variations of integrated circuits bring a lot of random, unpredictable changes to its structure. Therefore, each instance of a manufactured integrated circuit becomes unique and irreproducible with unpredictable physical characteristics. These unique changes can be extracted by designing special digital circuits which can produce unique digital responses as a result of applying specific digital challenges to the device. In general, circuit implementation of a PUF can be represented as a block with n digital inputs getting n-bit challenge C from all possible 2^n combinations and producing one single-bit output value R (Response). The behaviour of such circuit can be represented as a boolean function mapping $\{0,1\}^n \rightarrow \{0,1\}$. The randomness of this function can be explained by the fact that the exact mapping of the set of challenges to the set of responses is unknown until the device is fabricated. This property also depends on uncontrollable variations of all manufacturing stages. Thus, the set of all possible challenge-response pairs of a PUF $CR_\alpha = \{c_0 r_0, c_1 r_1, \ldots, c_{2^n-1} r_{2^n-1}\}$ determines the uniqueness of an instance α of the digital device and $r_i = \text{PUF}_\alpha(c_i)$, $i = \{0, 1, \ldots, 2^{n-1}\}$ determines unique dependency between a response r_i and a challenge c_i.

PUF should meet the following criteria to be utilized in physical cryptography [1, 2].

1. Hardware overhead for PUF implementation should not exceed the hardware cost of a protected device.
2. Collection, storage and analysis of the set CR_α should be physically infeasible using modern equipment within reasonable time. The PUF, which matches this criterion, is called a strong PUF for parameter $n \geq 64$. To store the whole set of R_α, the 16 exabyte memory device is required. At the same time, collection of all possible values of R_α takes approximately 580 years if the response time of a memory device is 1 ns.
3. Knowing information about the challenge-response pair $c_i r_i$ for a particular instance of a device α it is impossible to calculate, simulate or design a mathematical model to predict value of a pair $c_j r_j$, $i = j$, or any other subset of such pairs. If a PUF satisfies mentioned condition, it can be considered *random* and *unpredictable*.
4. For a particular device α the set of responses R^*_α, $|R^*_\alpha| < |R_\alpha|$, can be extracted many times with a high degree of reliability by applying corresponding set of challenges C^*_α, $|C^*_\alpha| < |C_\alpha|$ to the inputs of a PUF. The PUF which has this property can be considered stable (*reliable*).
5. The set $D = \{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$, $|D| = m$, for different instances of a digital device with embedded PUF circuit and fabricated using the same technology, should match the following condition $CR_{\alpha_0} = CR_{\alpha_1} = \ldots = CR_{\alpha_{m-1}}$. This condition can be strengthened by additional condition $R^*_{\alpha_0} = R^*_{\alpha_1} = \ldots = R^*_{\alpha_{m-1}}$ for a corresponding set of challenges $C_{\alpha_0} = C_{\alpha_0} = \ldots = C_{\alpha_{m-1}}$, $|C_{\alpha_0}| = |C_{\alpha_1}| = \ldots = |C_{\alpha_{m-1}}| = \lceil \log_2 m \rceil$. The PUF which has this property can be considered *unique*.

The PUF design which matches criteria described above can be efficiently used as a cryptographic primitive for following tasks.

– unclonable identification of digital devices;
– reliable authentication of digital devices;
– irreproducible random number sequences generation;
– hardware implementation of hash functions;
– hardware implementation of watermarks and fingerprints;
– protection digital devices against illegal cloning and modifying.

There are many circuit implementation of PUFs for digital device [1, 2]: arbiter PUF, ring oscillator PUF, butterfly PUF, memory based PUFs, etc.

Almost all of mentioned PUF designs are based on measuring delay differences between signals propagated by symmetrical paths formed by a set of sequentially connected digital elements.

One of the most widely used methods of a PUF implementation for digital devices based on measuring delay differences is called Arbiter PUF (A-PUF) [3–6]. In contrast to the other mentioned PUFs, this type can be classified as a strong PUF and involves acceptable hardware overhead. However, practical implementation of this PUF has some disadvantages, which are under investigation of many developers and researchers in physical cryptography area. One main issue of A-PUF design is its low reliability of challenge-response pairs set and a vulnerability to an accurate modeling using known subset of pairs CR^*_α, $|CR^*_\alpha| \ll |CR_\alpha|$ as circuit structure has linear dependency between challenge and response values.

2 Classical Arbiter PUF Design

Classical Arbiter PUF (A-PUF) implementation usually contains three blocks, namely Test Pulse Generator (TPG), Symmetrical Paths Block (SPB) and Arbiter Block (AB). This structure is shown in Fig. 1.

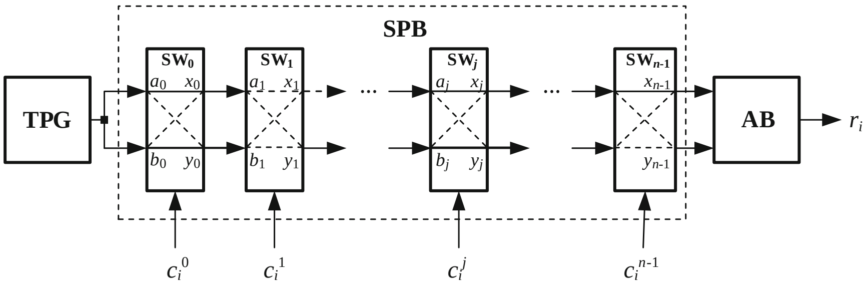
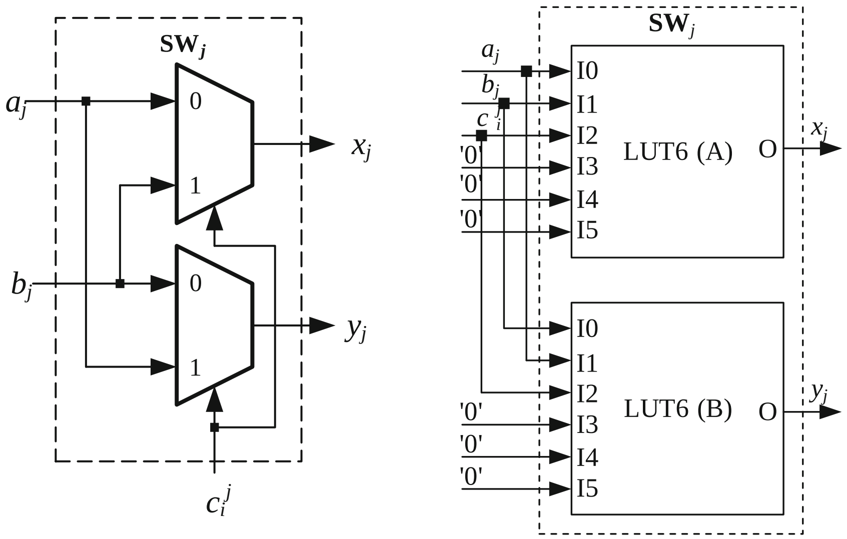


Fig. 1. Classical Arbiter PUF block level design.

SPB usually embeds n paths switching blocks (SW$_j$) controlled by external signals $c_i^j \in \{0, 1\}, j = \{0, 1, 2, \ldots, n-1\}$, which correspond to the input challenge bits. Each switch SW$_j$ has two inputs a_j, b_j and two outputs x_j, y_j. If $c_i^j = 0$ paths are configured in a straight mode, i.e. signal goes from input a_j to output x_j and from b_j to y_j, correspondingly. Otherwise ($c_i^j = 1$), switch SW$_j$ has an exchanging configuration, so input a_j is now connected to y_j and b_j to x_j. Thus, n switching blocks provide 2^n possible path configurations for propagation of two test pulses generated by TPG block. Arbiter block determines which of two signals arrives earlier, i.e. if signal from output x_{n-1} came faster than the one from output y_{n-1} AB produces response value $r_i = 1$. Otherwise, it outputs $r_i = 0$.

AB of classical A-PUF design usually triggered by rising edge of a test pulse produced by TPG. Therefore, synchronous D Flip-Flop (DFF) is used for response generation. Signal from x_{n-1} is propagated to the clock input of DFF and y_{n-1} – to the data input. One main problem of this circuit is metastability state of the DFF, which degrade overall stability of the A-PUF design. This issue can be resolved by implementing AB as latches (e.g. RS latch) or multiple flip-flops [4].

Major hardware overhead in A-PUF design with $n \geq 8$ switches is located in SPB. Consider a detailed implementation of SW components. Basic approach to implement one switching block usually utilizes a two-multiplexers circuit shown in Fig. 2a.



(a) RTL level diagram (b) LUT-block level diagram

Fig. 2. Circuit implementation of SW block.

The circuit in Fig. 2a can be synthesized as two LUT6 blocks as shown in Fig. 2b, which correspond to the combinational circuit of 6-input multiplexer and configurable memory. Input signals a_j and b_j are fed into the corresponding input ports I0 and I1 of the upper LUT6 (A) block and to the I1 and I0 ports of the bottom LUT6 (B) block, respectively. The challenge signal c_i^j is passed to the I2 input ports of both LUT6 blocks. The other input ports (I3, I4, I5) are not utilized if the standard synthesis configuration is used. The output signals x_j and y_j correspond to the output port O of the LUT6 block.

FPGA chips fabricated by Xilinx usually have 4- to 6-input LUT blocks. The number of inputs in hardware blocks depends on the FPGA architecture [7]. LUT block contains a configuration memory and a set of multiplexers for translation values stored in this memory. Address for the memory is formed by signals from multiplexer select inputs. Redundant inputs are usually fed with constant '0' value.

There are four types of LUT6 blocks in Xilinx Artix-7 FPGA, LUT6 (general output), LUT6_D (general and local output), LUT6_L (local output), LUT6_2 (two outputs). Figure 3 shows the structure of LUT6_L block which is configured as a multiplexer x_j from block SW$_j$ (see Fig. 2). As shown in Fig. 3, only 12.5% of LUT6 block resources are utilized. Therefore, if number of switches is significant, it will lead to a substantial LUT overhead to implement A-PUF circuit.

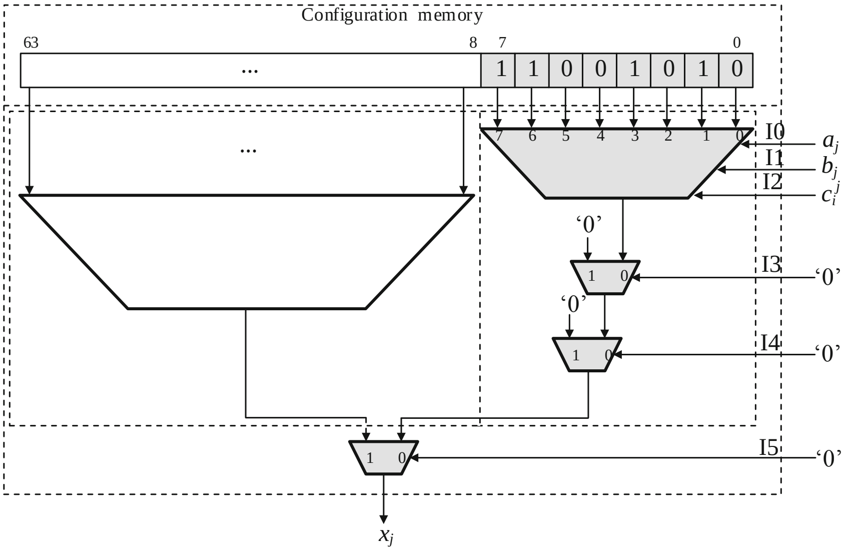


Fig. 3. Circuit implementation of a multiplexer using LUT6_L.

For example, for $n = 128$ implementation of the SPB requires 256 LUT blocks of Xilinx Artix-7 FPGA, which is around 1% of all resources available in XC7A100T chip [7].

3 Proposed Symmetrical Path Design

Classical implementation of A-PUF in FPGA provides SPB with unique parameters, e.g. delays of internal multiplexers providing translation of a chosen signal to an output port [8].

Assume time required for rising edge of a test signal to reach output x_j from input a_j ($c_i^j = 0$) be denoted as $\delta(x_j, a_j)$ and as $\delta(x_j, b_j)$ for input b_j, output x_j ($c_i^j = 1$), respectively. Similarly denote these characteristics for a second multiplexer, $\delta(y_j, a_j)$ ($c_i^j = 1$) and $\delta(y_j, b_j)$ ($c_i^j = 0$). To estimate these parameters the post place-route modeling of SW$_j$ has been performed for

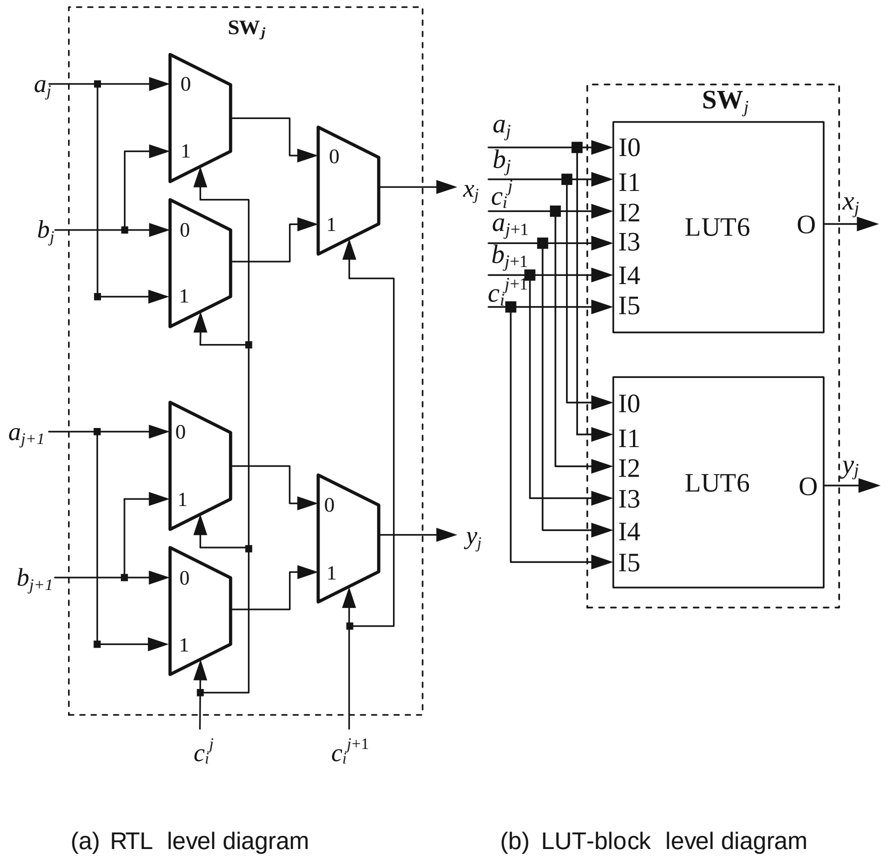


Fig. 4. Circuit implementation of SW block based on full utilization of LUT6 FPGA block.

XC7A100T FPGA platform. As a result, following values for SW$_0$ block have been obtained, $\delta(x_0, a_0) = 0.809$ ns, $\delta(x_0, b_0) = 0.309$ ns, $\delta(y_0, a_0) = 1.022$ ns, $\delta(y_0, b_0) = 1.022$ ns. For the neighbour block SW$_1$ these values differ significantly, $\delta(x_1, a_1) = 0.719$ ns, $\delta(x_1, b_1) = 0.506$ ns, $\delta(y_1, a_1) = 0.296$ ns, $\delta(y_1, b_1) = 0.083$ ns. These results can be explained by uniqueness of utilized LUT blocks and asymmetrical configuration of its connections. Unique values of the delays prove that the second copy of the switching block can be implemented using unutilized resources of a LUT block. Therefore, all unused inputs (I3, I4 and I5) can be reconfigured for an additional switching block. Figure 4 shows both, circuit implementation of the block containing SW$_j$ (see Fig. 4a) and the synthesized scheme on LUT6 blocks (see Fig. 4b).

This circuit provides four configurations of inputs $a_j, b_j, a_{j+1}, b_{j+1}$ to outputs x_j, y_j: two straight connections when $c_i^j = 0, c_i^{j+1} = 0$ and $c_i^j = 1, c_i^{j+1} = 1$, and two exchanging connections when $c_i^j = 1, c_i^{j+1} = 0$ and $c_i^j = 0$, $c_i^{j+1} = 1$.

Table 1 presents the rising edge delays for test signal for different combinations of inputs and outputs depending on c_i^j, c_i^{j+1}. These values have been obtained using post place-route modeling for $n = 16$ A-PUF implemented in Xilinx Artix-7 XC7A100T FPGA.

Table 1. Delay values for two different switching blocks.

Challenge bits $c_i^j c_i^{j+1}$	Delay type	Delay value for SW_1 block, ns
00	$\delta(x_j, a_j)$	0.443
	$\delta(y_j, b_j)$	0.481
01	$\delta(x_j, a_{j+1})$	0.392
	$\delta(y_j, b_{j+1})$	0.497
10	$\delta(x_j, a_j)$	0.609
	$\delta(y_j, b_j)$	0.654
11	$\delta(x_j, a_{j+1})$	0.660
	$\delta(y_j, b_{j+1})$	0.565

Shown results proves the high potential of using proposed design as A-PUF implemented in FPGA.

4 Hardware Overhead Analysis

As shown above, proposed structure of an SPB fully fits the LUT6 architecture. This approach provides significant reduction of hardware resources in FPGA chip. TPG and AB blocks bring negligible hardware overhead to the overall design comparing to the SPB. Implementation of the TPG block usually requires three LUT blocks and three flip-flops, AB uses only one flip-flop if implemented as in [4]. The major hardware resources are required for SPB. Proposed architecture requires double less LUT blocks comparing to the classical A-PUF design.

Table 2 presents a comparison for two implementations of A-PUF in XC7A100T FPGA, namely classical A-PUF (G_0) and a proposed one (G_1). As shown in table, proposed A-PUF switching blocks architecture requires double less hardware for its implementation. This design can be implemented in FPGA based on LUT blocks with more than 6 inputs [7].

Table 2. Hardware overhead for A-PUF implementation in Xilinx-7 FPGA ($n = 128$).

FPGA resource	# Utilized blocks		Total # of blocks	% of hardware overhead	
	G_0	G_1		G_0	G_1
Slices	131	69	15,850	0.83	0.44
Flip-Flops	4	4	126,800	0.0003	0.0003
LUT6	259	131	15,850	1.63	0.83

5 Arbiter PUF Figures of Merit Analysis

The modeling of two approaches for A-PUF implementation for XC7A100T FPGA have been conducted using Xilinx ISE 14.7 CAD [9]. These approaches have been post place-route simulated as 16-bit A-PUFs. AB has been implemented as synchronous D Flip-Flop (technological component FDC).

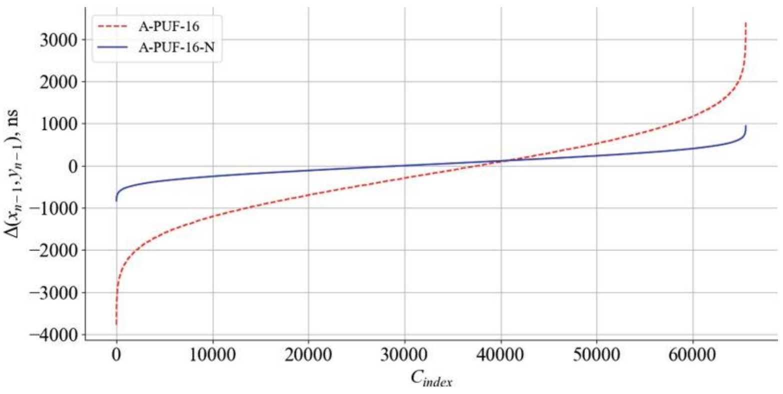


Fig. 5. Graph of $\Delta(x_{n-1}, y_{n-1})$ for two different A-PUF implementations.

A-PUF design and testbenches have been designed using Verilog. Testbenches provide feeding of all possible 2^n challenges to the inputs, generation of a test signal and response values analysis. Furthermore, testbenches have been designed in order to analyse delay difference $\Delta(x_{n-1}, y_{n-1})$ between rising edges of two copies of a test signal, propagated from the outputs x_{n-1} and y_{n-1} to the inputs of AB.

As shown in Fig. 5, the curve of $\Delta(x_{n-1}, y_{n-1})$ values for A-PUF-16-N (proposed design) is much more symmetric in both axes. Furthermore, nonlinear dependency between coordinates makes challenge-response relation much more complicated comparing to the classical A-PUF design. Asymmetry in A-PUF-16 (classical design) graph implies that the set of challenge-response pairs is unbalanced, i.e. probability of zero response bit is much bigger than the probability of one.

Furthermore, the range of delay values $\Delta(x_{n-1}, y_{n-1})$ has shrunken. As a result, the range for classical A-PUF design is [−3772; 3396] ns and for the proposed architecture with the same n, this range has become much more narrow [−832; 943] ns. This fact can be explained by decreasing length for symmetric paths as number of LUT blocks is also decreased. Inter-chip uniqueness has been estimated for both, A-PUF-16 and A-PUF-16-N circuits.

To calculate this metric eight identical components of A-PUF on each chip have been simulated and compared on the same set of challenges. This metric is estimated as 0.489 for A-PUF-16 and 0.473 for A-PUF-16-N, respectively.

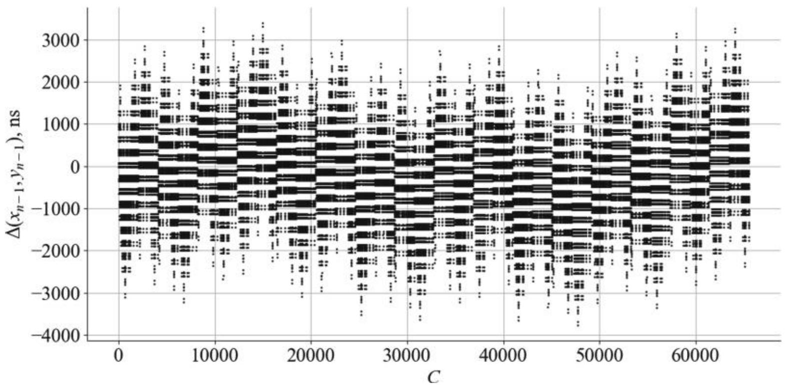


Fig. 6. Graph of $\varDelta(x_{n-1}, y_{n-1})$ depending on challenge values for A-PUF-16 design.

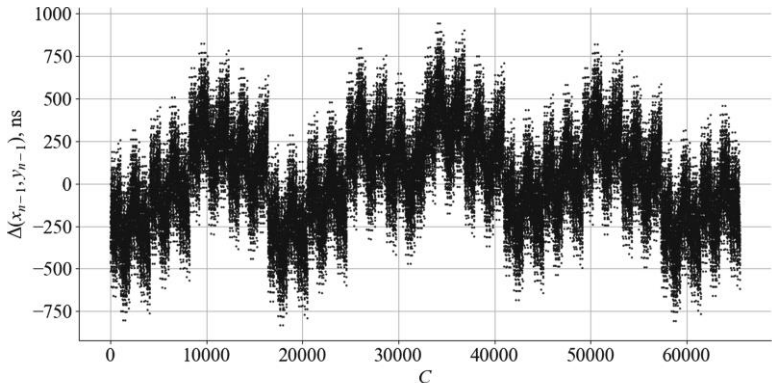


Fig. 7. Graph of $\varDelta(x_{n-1}, y_{n-1})$ depending on challenge values for A-PUF-16-N design.

The ideal value for uniqueness figure of merit is 0.5. Real values of reliability and uniqueness strongly depend on parameter n, AB implementation and can be actually measured only on real hardware [10] by feeding the same challenges multiple times.

Figures 6 and 7 show functional dependency of values $\varDelta(x_{n-1}, y_{n-1})$ on challenge values, sorted in ascending order by the value of C for A-PUF-16 and A-PUF-16-N circuits, respectively.

As shown in figures, circuit A-PUF-16-N has better randomness and values of $\varDelta(x_{n-1}, y_{n-1})$ are less correlated to the challenge values C. This fact can increase the complexity of the machine learning modeling for an attacker [3].

6 Future Works

The proposed design still utilizes two LUT6 for implementation of SW_j block. Alternatively, this block can be synthesized based on two-output LUTs (LUT6_2 or CFGLUT5 in Xilinx 7 Series FPGAs [11]). Both of these LUT blocks utilize one of the inputs to configure the outputs, i.e. if the passed value is '0' then both outputs produce the same value, otherwise the output values are different. Since one of the inputs has to be used in order to provide two distinct outputs, 6 input values for SW_j and SW_{j+1} cannot be fit to 5 available inputs. Therefore, it is more promising to use CFGLUT5 block for a single-LUT SW_j block implementation as it also provides dynamically reconfigurable memory. This feature of the CFGLUT5 makes PUF reconfigurable and increases its uniqueness and security [12].

The circuit implementation of the SW_j block is shown in Fig. 8.

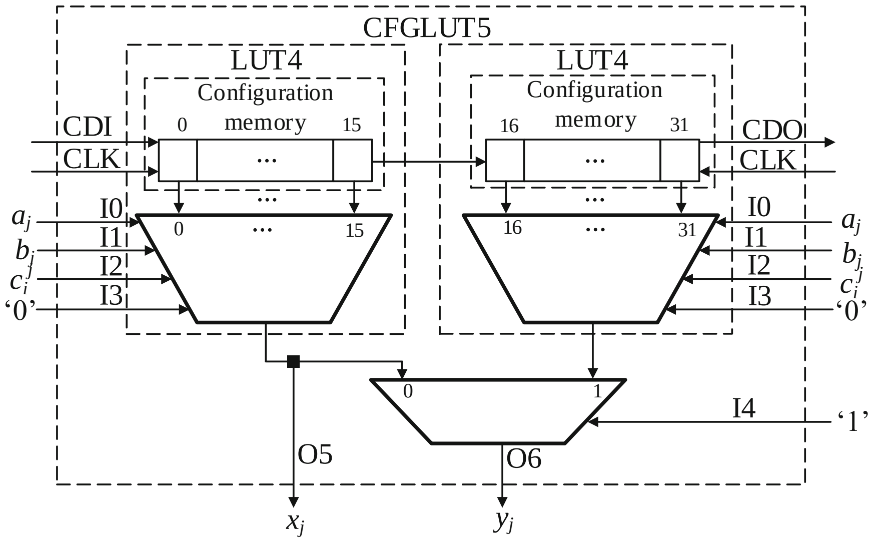


Fig. 8. Circuit implementation of SW block using CFGLUT5.

The input values a_j, b_j and c_i^j are mapped to the inputs I0, I1 and I2, respectively. The input I3 is fed with '0' as it is not utilized for switching block implementation. The input I4 gets the value of '1' as LUT block should have two different outputs O5 and O6 which are mapped to the output values x_j and y_j. The CFGLUT5 can be also dynamically reconfigured using serial port CDI and reconfiguration clock CLK. Thus, switch block can be reconfigured to different paths increasing uniqueness and decreasing the vulnerability to machine learning attacks.

7 Conclusion

This paper presents a new architecture for symmetrical paths block in arbiter based physical unclonable function for efficient implementation in FPGA. Proposed approach utilizes the internal configuration of embedded LUT blocks and saves valuable hardware resources as well as enhances uniqueness and randomness of a classical A-PUF design.

All experimental data described in this paper has been obtained using CAD Xilinx ISE 14.7 [9] and Verilog hardware description language. The results are to be verified on a real hardware implementation to compute true values of inter-chip uniqueness and randomness. The proposed design has to be also verified on vulnerabilities related to machine learning based modeling attacks. The circuit is also highly configurable and can be used to design a logically reconfigurable PUFs.

References

1. Zalivaka, S.S., Zhang, L., Klybik, V.P., Ivaniuk, A.A., Chang, C.-H.: Design and implementation of high-quality physical unclonable functions for hardware-oriented cryptography. In: Chang, C.-H., Potkonjak, M. (eds.) Secure System Design and Trustable Computing, pp. 39–81. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-14971-4_2
2. Ivaniuk, A.A.: The design of embedded digital devices and systems, 337 p. Bestprint, Minsk (2012). (in Russian)
3. Zalivaka, S.S., Ivaniuk, A.A., Chang, C.-H.: Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response. IEEE Trans. Inf. Forensics Secur. **14**(4), 1109–1123 (2018). https://doi.org/10.1109/TIFS.2018.2870835
4. Zalivaka, S.S., Puchkov, A.V., Klybik, V.P., Ivaniuk, A.A., Chang, C.-H.: Multi-valued arbiters for quality enhancement of PUF responses on FPGA implementation. In: Proceedings Asia and South Pacific Design Automation Conference (ASP-DAC 2016), Macau, China, January 2016, pp. 533–538 (2016). (Invited paper). https://doi.org/10.1109/ASPDAC.2016.7428066
5. Hori, Y., Yoshida, T., Katashita, T., Satoh, A.: Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In: Proceedings International Conference "Reconfigurable Computing and FPGAs", Mexico, pp. 298–303 (2010). https://doi.org/10.1109/ReConFig.2010.24
6. Becker, G.T.: On the pitfalls of using Arbiter-PUFs as building blocks. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **34**(8), 1295–1307 (2015). https://doi.org/10.1109/TCAD.2015.2427259
7. Series FPGAs Data Sheet: Overview. https://www.xilinx.com/supp ort/documentation/data _sheets/ds180_7Series_Overview.pdf . Accessed 13 Feb 2019
8. Morozov, S., Maiti, A., Schaumont, P.: An analysis of delay based PUF implementations on FPGA. In: Sirisuk, P., Morgan, F., El-Ghazawi, T., Amano, H. (eds.) ARC 2010. LNCS, vol. 5992, pp. 382–387. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12133-3_37
9. ISE Design Suite. https://www.xilinx.com/pro ducts/design- tools/ise- design- suite.html. Accessed 20 Feb 2019
10. Nexys 4 Artix-7 FPGA Trainer Board: https://store.digilentinc.com/nexys- 4-artix- 7-fpga-trainer-board-limited- time-see-nexys4-ddr. Accessed 20 Feb 2019
11. Xilinx 7 Series FPGA Libraries Guide for Schematic Designs: https://www.xilinx.com/support/documentation/sw _manuals/xilinx13 _2/7series_scm.pdf . Accessed 05 Sept 2019

12. Katzenbeisser, S., et al.: Recyclable PUFs: logically reconfigurable PUFs. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 374-389. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_25