Robust and Flexible FPGA-based Digital PUF

Teng Xu and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{xuteng, miodrag}@cs.ucla.edu

***Abstract*—We have developed the first FPGA-based digital physical unclonable function (PUF) by leveraging the reconfigurability of an FPGA and introducing a new way of using the standard analog delay PUF. The key observation is that for any analog delay PUF, there is a subset of challenge inputs for which the PUF output is stable regardless of operation and environmental conditions. We use only such stable inputs to initialize the look-up tables (LUTs) that are configured in such a way that the digital PUF is formed. We demonstrate the effectiveness of the new security primitive using extensive simulation and experimental results. For example, we show that the new PUF is resistant against a wide spectrum of security attacks and its output stream passes all the NIST randomness tests.**

I. Introduction

The rapid proliferation of mobile systems and devices that operate in potentially hostile environments has elevated security to one of the most important design metrics. For example, security is essential in smart phones, laptops, and wireless sensor networks. Classical software-based public-key cryptography provides a spectrum of elegant and powerful security protocols. However, it is also subject to several important limitations and drawbacks including susceptibility to physical and side channel attacks and high implementation and energy costs.

Hardware-based (especially FPGA-based) security systems and techniques resolve many of these problems. For instance, they are naturally resilient against physical and side channel attacks and highly efficient in terms of area, speed, and energy. On the FPGA platform, a number of lightweight protocols for several cryptographic primitives are presented. A physically unclonable function (PUF) is a deterministic multiple-input multiple-output hardware system that is hard to reverse engineer or to simulate. The essential idea is to utilize process variation to create a hardware-based security primitive which is intrinsically resilient to physical and side channel attacks and, more importantly, is unclonable. Using an FPGA to build a delay-based PUF is proposed in [1].

However, PUFs and their hardware security descendants still suffer from a large number of security, design, and operational problems. For instance, they are susceptible to attacks enabled by pending implementation technologies and they sometimes require very accurate and expensive measurements. Most of all, they lack stability in the presence of variations in environmental and operational conditions, such as supply voltage and localized temperature variations. These problems are closely related to the fact that all of these security hardware platforms operate as analog systems that are notoriously well-known for their susceptibility to these conditions. Therefore, there is a need for conceptually new hardware platforms that are intrinsically digital and, hence, robust.

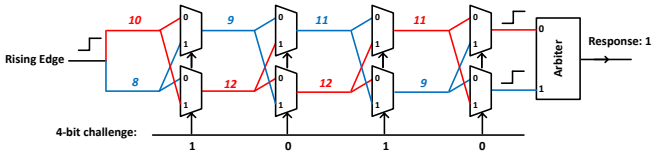


Fig. 1: A 4-bit delay-based PUF and example challenge input. The challenge is intentionally chosen in such a way that the delay difference between the two paths as shown in red and blue are maximized. The paths in red correspond to the gate with larger delay at each stage. The response bit is 1 because the path in blue reaches the arbiter first.

The objective of our paper is to propose a novel type of FPGA-based digital PUF. The starting point is from the key observation that a portion of challenges for a delay-based PUF can bring about more stability than other input challenges. Figure 1 shows an example of a delay-based PUF with a 4-bit challenge as the input. The paths in red correspond to the gates with larger delay at each stage, while the paths in blue correspond to the gates with the smaller delay. If we use 1010 as the 4-bit challenge, the delay difference between the two paths is maximized. Now suppose the temperature increases. Consequently, the delay of the gates will also increase. However, since the challenge 1010 already provides a large difference in path delay, even though the delay increases, there is a very high probability that the red path will still have a larger delay in comparison to the blue path and, hence, is considered stable. We define such challenges which can stand variations in environmental conditions as stable challenges.

We take the next step by leveraging the delay-based PUF with the concept of an FPGA-based digital bimodal function (DBF) which was first proposed in [2]. The essential idea behind the DBF is to have two forms of a function in which one form is fast and compact ($f_{compact}$) and the other is slow and complex ($f_{complex}$). Both forms have exactly the same functionality, (i.e. given the same input, both forms produce the same outputs). Figure 2 demonstrates an example of the FPGA-based implementation of $f_{compact}$. The architecture is composed of a randomly connected FPGA network. The hierarchical structure is constructed by feeding the outputs of the previous stage's LUTs as the input to the next stage. Meanwhile, we use the direct mapping between the primary input and the final output as $f_{complex}$ of the DBF. As a result, both forms have exactly the same functionality. However, as the number of primary inputs and stages in the LUT network

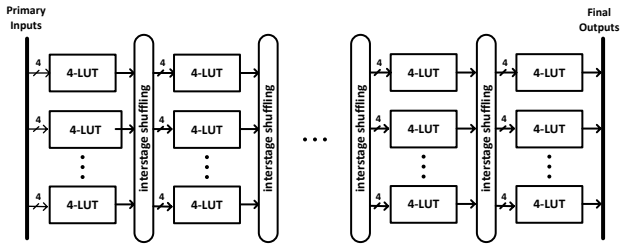


Fig. 2: Example of an FPGA-based DBF LUT network.

grows, the hardware implementation of $f_{compact}$ stays in a relatively compact form while $f_{complex}$ grows exponentially.

A number of security protocols can be designed using DBFs. However, one significant drawback of the DBF is that it can easily be reverse engineered once an attacker learns the configuration of the LUT network. Our FPGA-based digital PUF solves this problem by integrated the delay-based PUF into the design and using its responses of stable challenges to initialize some of the LUT cells in the DBF. Note that we only utilize the stable challenge-response pairs so that our design is resilient against environmental conditions.

The FPGA provides an ideal platform to implement the whole design. The reconfigurability of the FPGA allows the use of the delay-based PUF to initialize the DBF cells. Moreover, the flexible configuration of the DBF enables self trust. For example, the user can customize their device by reconfiguring their FPGA which potentially prevents malicious producers.

Finally, we claim that our design forms a digital PUF in the following sense. In the first place, the basic idea of our proposed design is to utilize the digital functionality of the pseudorandomly connected LUT network to generate multi-input and multi-output mappings. However, traditional PUFs rely on the effect of process variation in the analog properties of the circuit (e.g., delay, frequency, leakage) to generate unique outputs. In our design, although we also utilize the delay-based PUF to ensure unclonability, the responses are only used to initialize the LUT cells of the DBF. Another important argument is that our FPGA-based digital PUF is not subject to operational and environmental conditions. For instance, it is a known fact that many analog systems are highly sensitive to conditions, especially temperature, and supply voltages. Our FPGA-based digital PUF completely removes this problem because of the digital nature of the DBF and the utilization of the stable challenge-response pairs for the part of the delay-based PUF.

II. Related Work

A. Physical Unclonable Functions

A PUF is an unclonable hardware system that has a complex but definite mapping of inputs to outputs. In essence, a PUF is a very complex mathematical function derived from the intrinsic physical behavior of its design. The mapping of inputs to outputs must remain easy to evaluate (for the purposes of authentication) but impossible to predict (for the purposes of security). Pappu et al. demonstrated the first active physical unclonable function using optical mesoscopic systems in 2001[3]. Devadas and members of his research group observed that the intrinsic deep submicron process variation in silicon is an ideal practical and economical starting point to fabricate a large amount of PUFs [4][5].

The delay-based FPGA PUF was first proposed in [1]. They take advantage of the intrinsic delay difference between LUTs to design the delay path. Another type of candidate FPGA PUF is the SRAM PUF which is based on the fact that SRAM cells after powering up are initialized to specific values due to process variation [6]. PUFs used for public key communication are proposed in [7] and some lightweight PUFs are proposed in [8], [9] and [10]. The PUF stability in the presence of variations in environmental and operational conditions is analyzed in [11].

B. Testing PUFs

A number of approaches have been demonstrated to test PUF security properties [12][13]. In addition to testing our PUF using the current criterion, our approach also tests PUF security using standard randomness tests, such as the NIST randomness test suite [14]. As a result, if one succeeds in breaking a PUF that passes the randomness tests it is equivalent to breaking the widely used statistical test.

We briefly compare our FPGA-based digital PUF with other related designs and PUFs. Compared to the DBF design, we have two main differences. First, our FPGA-based digital PUF is unclonable due to the fact that we introduce a delay-based PUF in conjunction with our design while the DBF is clonable. Second, we configure the FPGA using the outputs of the delay-based PUF in which the users can customize the initialization while the DBF is configured by the manufacturer. Compared to the traditional delay-based PUF, our PUF is completely digital and resilient against environmental conditions. Compared to the SRAM PUF, which is a type of weak digital PUF, our proposed PUF has an exponential number of input-output pairs while an SRAM PUF only has a polynomial number of pairs. Furthermore, we prove that the response stream of our FPGA-based digital PUF passes the NIST randomness test suite which has not been directly used as a testing standard for previous PUF designs.

III. Desiderata

Our desiderata of the FPGA-based digital PUF consists of the following requirements. (i) Regarding architecture, we desire a low energy, delay, and area implementation on FPGA. Meanwhile, the digital nature should enable PUF stability against environmental conditions, e.g., temperature and supply voltage. (ii) Regarding security, our design should be resilient against security attacks, both statistical attacks (e.g., coincidence analysis) and physical attacks (e.g., cloning attacks). (iii) Regarding applications, a novel digital PPUF-based random number generator is desired.

IV. PUF Stability

We use the the delay ratio, as defined in Equation 1, to evaluate the relative delay difference between the two PUF paths. In the following test we assume that the gate delays

Delay Ratio (Original T=300K)	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Prob. to stay @ T=250K	0.979	0.987	0.994	0.997	1	1	1
Prob. to stay @ T=350K	0.969	0.975	0.989	0.994	0.997	1	1
Prob. to stay @ T=400K	0.937	0.951	0.959	0.977	0.988	0.996	1

TABLE II: Probability that the output of a 32-bit PUF is stable over varying temperature conditions for different original delay ratios. Assume the original temperature is 300K, we test under 250K, 350K, and 400K respectively.

Delay Ratio (Original T=300K)	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Prob. to stay @ T=250K	0.984	0.986	0.996	0.998	1	1	1
Prob. to stay @ T=350K	0.982	0.986	0.993	0.998	1	1	1
Prob. to stay @ T=400K	0.954	0.974	0.986	0.991	0.997	1	1

TABLE III: Probability that the output of a 64-bit PUF is stable over varying temperature conditions for different original delay ratios.. Assume the original temperature is 300K, we test under 250K, 350K, and 400K respectively.

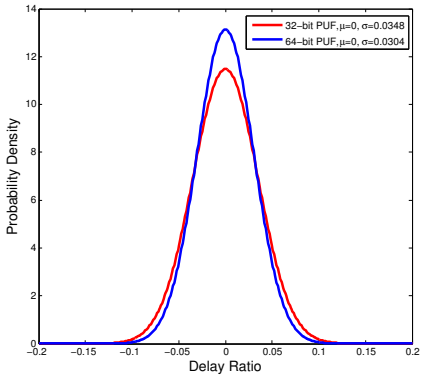


Fig. 3: Distributions of delay ratios for a 32-bit PUF and a 64-bit PUF.

	P(DR≥0.04)	P(DR≥0.06)	P(DR≥0.08)	P(DR≥0.1)
32-bit PUF	12.51%	4.27%	1.07%	0.21%
64-bit PUF	9.34%	2.44%	0.43%	0.05%

TABLE I: Probability that the delay ratio (DR) is larger than some value for a 32-bit PUF and a 64-bit PUF.

of the PUF follow a normal distribution due to the effects of process variation.

$$Delay\ Ratio = \frac{Delay_{p1} - Delay_{p2}}{min(Delay_{p1}, Delay_{p2})} \quad (1)$$

The distribution of the delay ratio for random challenges on a 32-bit PUF and a 64-bit PUF are depicted in Figure 3. In both cases, they follow a normal distribution with their means at 0. The standard deviation of the 64-bit PUF is smaller than the 32-bit PUF. To better visualize the probability that the delay ratio is larger than some value, we use Table I to show the quantified result.

We simulate and test the stability of our PUF under different environmental temperatures. In our test, we assume that the original distribution of the gate delay D_{old} at temperature 300K follows the Gaussian distribution $D_{old} \sim \mathcal{N}(\mu, \sigma^2)$. When temperature changes, the delay changes: $D_{change} \sim \mathcal{N}(\alpha\mu, |\alpha|\sigma^2)$ where α is a ratio which is decided by the new temperature, thus yielding the new delay $D_{new} = D_{old} + D_{change}$. We use the Hotspot tool [15] to compute α under different temperatures. For example, α under 400K is approximately 1. Based on this assumption, under different original delay ratios and after applying temperature changes, we measure the probability that the same challenge produces the same stable outputs. Table II shows these results on a 32-bit delay-based PUF. We draw the conclusion that with a higher original delay ratio, the chance that the PUF output remains stable is higher. For example, when the original delay ratio reaches 0.1, the probability that the output remains stable is 1 no matter how the temperature changes (250K-400K). We repeated the same tests on a 64-bit delay-based PUF and display result in Table III. Compared to the 32-bit test case, the 64-bit test case demonstrates a similar trend and even exhibits better stability under the same conditions. Therefore, we conclude that as long as the original delay ratio reaches some threshold (e.g., 0.1 according to this test), the output will be stable for a wide range of temperature conditions (250K-400K). Challenges that match this threshold are selected as the stable challenges.

V. Architecture

The architecture of the FPGA-based digital PUF shown in Figure 4 is composed of two parts, a delay-based PUF and a DBF. In each clock cycle the delay-based PUF receives a k-bit stable challenge and produces a 1-bit output based on the comparison of the two path delays. The output is then used to initialize an even cell among the LUTs in the DBF while the odd cells keep their original values. We can also make the part to initialize to be flexible from time to time and controlled by the users. In this way we can ensure self trust. For simplicity, the DBF in Figure 4 consists of 2-input LUTs to build an n-bit input m-bit output LUT network, and only the even cells in the LUTs are initialized by the standard delay-based PUFs.

When using the delay-based PUF to initialize the DBF, only the challenges that can produce the stable outputs are used. The use of these stable inputs in conjunction with the inherent stability of the DBF due to its digital nature makes the whole design resilient against environmental conditions. Note that the intrinsic unclonability of the delay-based PUF guarantees the overall architecture to be unclonable.

VI. Operations

In order to use the FPGA-based digital PUF, a set of operations need to be done, we divide them into the following

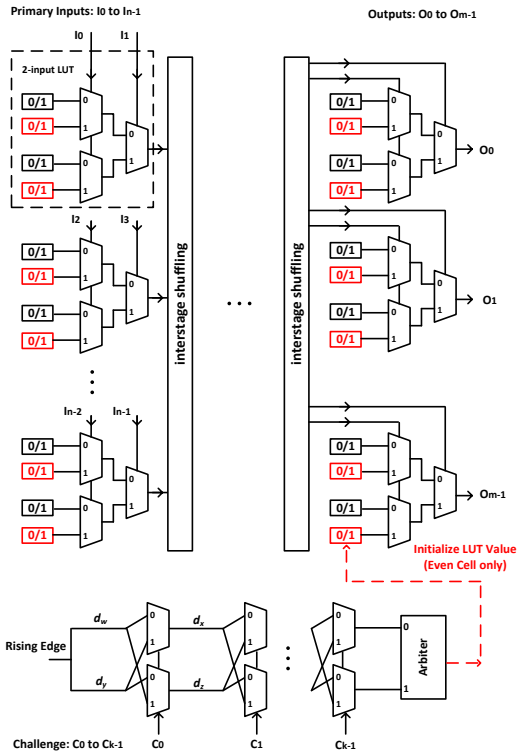


Fig. 4: Overall architecture of the FPGA-based digital PUF. The primary inputs for the DBF are $\{I_0, \ldots, I_{n-1}\}$, and the outputs are $\{O_0, \ldots, O_{m-1}\}$. The corresponding 1-bit output of every stable challenge (e.g., $\{C_0, \ldots, C_{k-1}\}$) from the delay-based PUF is used to initialize an SRAM cell in the LUTs of the DBF. In this example, they are only used to initialize the even cells of the LUTs.

two steps: (i) FPGA configuration and (ii) DBF generation.

A. FPGA Configuration

The essential step before using the DBF is to reconfigure the FPGA for DBF initialization. As we mentioned in the previous section, in every clock cycle the user needs to choose a stable challenge vector $\{C_0, \ldots, C_{k-1}\}$ and feed it to the delay-based PUF. Each challenge is randomly chosen from a pool of stable challenges that is provided by the manufacturer. Note that if we reverse all the challenge bits to $\{\overline{C_0}, \ldots, \overline{C_{k-1}}\}$, the output is reversed too. Hence, there will not be any bias between the number of stable challenges that produce the 0 output and the 1 output. As a result, the output randomness of the delay-based PUF is not reduced because of the use of only stable challenges. Each of the generated output will be used to initialize one cell in a LUT. This procedure is repeated until a random portion of the LUT cells in the DBF are initialized, and the rest are initialized by the user. The reason to choose only a random part of LUTs to initialize is to enable self trust in order to prevent malicious manufacturers, because the manufacturers have no clue how the rest of the LUTs are initialized.

B. DBF Generation

After initialization of the DBF in the FPGA, the user generates the input-output mapping for the DBF which serves as $f_{complex}$. This can be easily done by traversing all the possible inputs and generating the corresponding outputs. The mapping is stored in the form of Boolean functions. Therefore, until now, the initialized DBF embedded in the FPGA serves as $f_{compact}$ and the mapping is $f_{complex}$.

VII. Security Attacks

In this section, we analyze the security properties of the FPGA-based digital PUF. The basic approach is to identify its resiliency against two classes of security attacks: (i) guessing using statistical models, and (ii) technological attacks. In guessing attacks, the attacker observes a polynomial number of challenge-response pairs and statistically analyzes them in order to predict outputs for new challenges. Technological attacks are based on commonly used attack technology, e.g., cloning attacks, side-channel attacks, brute-force simulation, and utilization of special purpose hardware.

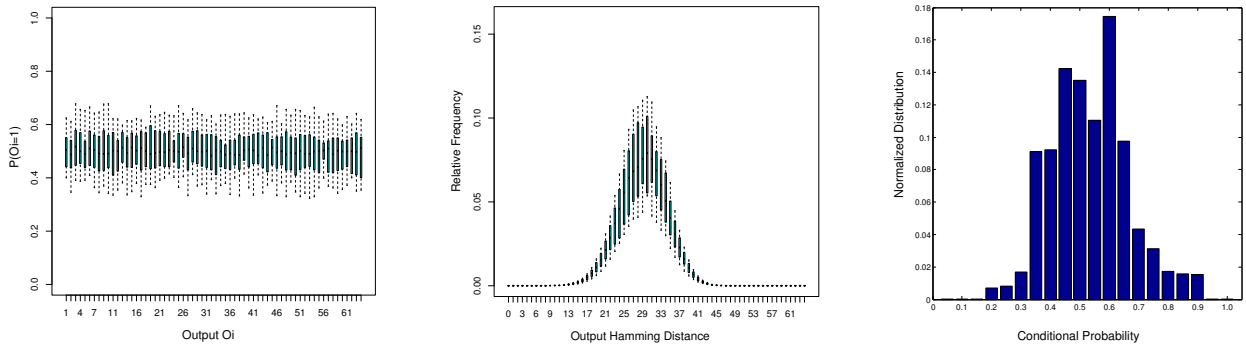
For each type of attack, we conduct comprehensive tests using the FPGA-based digital PUF which is composed of the 64-bit delay-based PUF and the pseudorandomly initialized DBF with 64-input and 64-output. The tests are based on the Xilinx Spartan 6 FPGA platform.

A. Guessing with Statistical Model

1) Frequency Prediction: In this attack, the attacker collects the output data from the DBF and builds a probability distribution for each output. Ultimately, the attacker tries to predict each output O_i based on statistical distributions. The goal of the attacker is to predict $P(O_i = c)$, where $c = 0$ or 1. The ideal situation is that an output is 0 or 1 with a probability of 0.5. Figure 5a shows the mean value of the probability that each output bit is equal to 1. The probability is always close to 0.5, indicating high unpredictability.

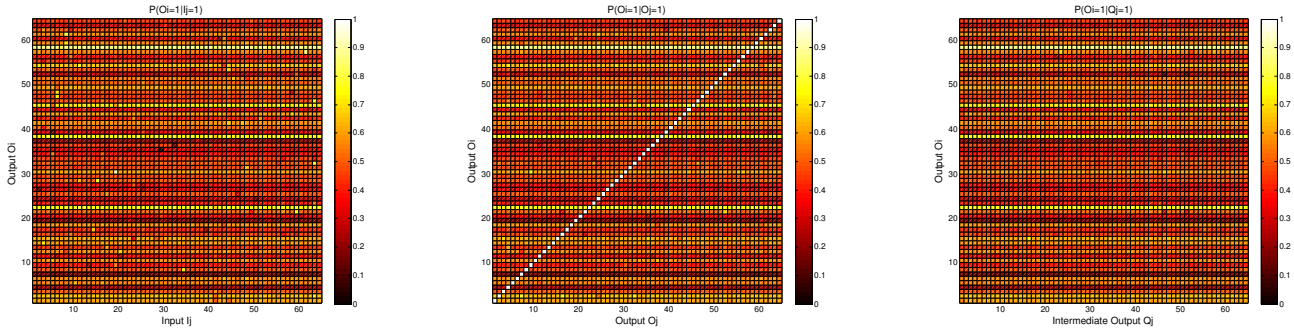
2) Avalanche Effect: In this attack, the attacker tries to predict outputs using knowledge of the outputs for similar inputs. This attack is dangerous when output vectors with similar input vectors are highly correlated with one another. To test this, we analyze the hamming distance between output vectors by changing one bit of the input vectors at each iteration. Ideally, the distribution should be in the form of a binomial distribution with the peak at half of the number of output bits, thus indicating that the avalanche effect is achieved. The result in Figure 5b shows an almost perfect binomial distribution which indicates our matched device is highly resilient against this type of attack.

3) Input-based Correlation: Another type of attack attempts to build correlation mappings between an output bit, O_i, and an input bit, I_j. The goal in this attack is to predict the conditional probability, $P(O_i = c_1 | I_j = c_2)$, where c_1 and c_2 are either 1 or 0. For example, if the attacker observes that output O_i is equal to 1 a large majority of the time when the input I_j is 1, then he can guess with a high probability that output O_i is 1 when I_j happens to be 1. The ideal situation is when the probabilities remain 0.5. Figure 6a depicts a colormap of the conditional probability $P(O_i = 1 | I_j = 1)$. We count and plot the fraction of the conditional probability values in the input-output pair matrix that falls in different range as shown in Figure 5c. Both Figure 6a and Figure 5c indicate low potential for prediction.



(a) (b) (c)

Fig. 5: (a) Probability that an output bit is equal to 1. (b) Distribution of output hamming distances in testing the avalanche effect. The error bars depict the max, 0.75 quantile, mean, 0.25 quantile, and min. (c) Distribution of conditional probabilities $P(O_i = 1|I_j = 1)$ for all pairs of inputs, i, and outputs, j.



(a) (b) (c)

Fig. 6: Conditional probabilities between (a) output bits O_i and input bits I_j, (b) output bits O_i and output bits O_j, and (c) output bits O_i and intermediate output bits Q_j.

4) Output and Intermediate Output-based Correlation: Similar to the previously described attack, this attack attempts to predict an output bit O_i according to the value of a corresponding output bit O_j or even an intermediate output bit Q_j. In the first case, if two output bits have a strong correlation, then the attacker can deduce the output vector by knowing a subset of the output bits. In the second case, we assume the attacker has somehow obtained some intermediate results, possibly through a side-channel attack, then attempts to predict the final outputs based on the mapping. We present a conditional probability map of $P(O_i = 1|O_j = 1)$ in Figure 6b and $P(O_i = 1|Q_j = 1)$ in Figure 6c depicting the low potential for prediction based on output to output correlation and output to intermediate output correlation.

Finally, we briefly compare the statistical test results of our FPGA-based digital PUF with the traditional delay PUF. The results depicted in Table IV are tested on the 64-input 64-output FPGA-based digital PUF and 64-input 64-output traditional delay PUF. We conclude that both our PUF and the traditional delay PUF demonstrate excellent properties regarding output frequency and conditional probability, but only our digital PUF demonstrates an ideal output hamming distance satisfying the avalanche criterion.

	Output Freq.	Hamming Dis.	$P(O_i = 1\|I_j = 1)$
Digital PUF	0.5 ± 0.07	30.9 ± 3.6	0.49 ± 0.01
Delay PUF	0.5 ± 0.09	1.4 ± 0.4	0.52 ± 0.01

TABLE IV: Statistical test results comparison between the FPGA-based digital PUF and the traditional delay PUF. The ideal case for output frequency is 0.5, for hamming distance is 32, and for $P(O_i = 1|I_j = 1)$ is 0.5. The results shown in the table are the average values and corresponding standard deviations.

B. Technological Attacks

1) Cloning Attack: The most intuitive type of technological attack is to clone an identical piece of the FPGA-based digital PUF. Our delay-based PUF mainly takes advantage of process variation in the FPGA. Although many efforts are being taken to minimize the effect of process variation as technology improves, as long as the bias still exists, the delay-based PUF will continue to work and is even more unpredictable. Therefore, due to its unclonability, there is no way for the attacker to figure out the digital PUF configuration. Moreover, since every initialization of our digital PUF is only being used for one time communication, as a result, even if the digital bimodal function on the FPGA after configuration is cloned by an attacker, it will not be a threat because the same configuration is not used for the next round of communication.

2) Side-channel Attack: On one hand, because the mapping between the input-output pairs of the DBF serve as the public key, the power profile of a gate would not reveal any new information. On the other hand, 3D implementation can be used to further hide power information. For example, the digital PUF can be placed in the middle-most layers. Devices with the same architecture but with randomly selected parameters are placed both above and below the actual device. Therefore, the side-channel attack is not a threat.

3) Brute-force Simulation: Suppose the attackers create a LUT to store all the possible challenge-response pairs. This is not feasible due to fact that the number of pairs which need to be enumerated grows exponentially with the number of inputs to the DBF.

4) Special Purpose Hardware: This type of attack requires the attacker to use a very fast processor, e.g., FPGA or ASIC, to simulate the FPGA-based digital PUF. The key thing to note here is that our design can not be reversed engineered and it is easy to create an exponentially larger FPGA-based digital PUF (in terms of simulation effort) by simply adding more inputs and levels of LUTs to the DBF.

VIII. Hardware Random Number Generator

The FPGA-based digital PUF can serve as a hardware random number generator due to its excellent output randomness. We generate the output stream in the following way. Assume that we have 2 independent DBFs: DBF_1 and DBF_2. A random seed is provided as the primary input to DBF_1 and its outputs are shuffled through DBF_2 and fed back as the input to DBF_1 in the next iteration. Meanwhile, the outputs of DBF_1 are collected as the output stream. After this process, we apply Von Neumann correction on the output stream.

We quantify the statistical randomness of the FPGA-based digital PUF outputs by applying the industry-standard statistical test suite of the National Institute of Standards and Technology (NIST). The results in Table V indicate that our system passes all NIST tests.

Statistical Test	Avg. Success Ratio
Frequency	100%
Block Frequency (m=128)	98.9%
Cusum-Forward	99.1%
Cusum-Reverse	99.1%
Runs	97.7%
Longest Runs of Ones	98.4%
Rank	99.6%
Spectral DFT	98.9%
Non-overlapping Templates ($m = 9$)	97.4%
Overlapping Templates ($m = 9$)	97.8%
Universal	100%
Approximate Entropy ($m = 8$)	99.1%
Rand. Excursions ($x = 1$)	97.5%
Rand. Excursions Variant ($x = -1$)	97.3%
Serial ($m = 16$)	98.5%
Linear Complexity ($M = 500$)	97.0%

TABLE V: The average success ratio for the NIST statistical test suite. 1000 bitstreams of 10000 bits are passed to each test. The test passes for p-value$\geq \sigma$, where σ is 0.01.

IX. Conclusion

We have proposed a new type of digital PUF on an FPGA which leverages the traditional delay-based PUF and the DBF by intentionally choosing stable challenge-response pairs from the delay-based PUF and using them for DBF initialization. Our design inherits the security properties from both the delay-based PUF and the DBF regarding small power, delay, and area costs as well as unclonability. Furthermore, the proposed FPGA-based digital PUF enables complete elimination of the standard PUF vulnerabilities, such as susceptibility to operational and environmental variations. We take advantage of the configurability of the FPGA and have proposed an architecture for our design. Our tests indicate that the proposed PUF is resilient to a wide spectrum of security attacks and its output stream passes the NIST randomness test suite.

X. Acknowledgement

This work was supported in part by the NSF under Award CNS-0958369, Award CNS-1059435, and Award CCF-0926127, and in part by the Air Force Award FA8750-12-2-0014.

References

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

[2] T. Xu, J. B. Wendt, M. Potkonjak, "Digital Bimodal Function: An Ultra-Low Energy Security Primitive," *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 292-297, 2013.

[3] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026-2030, 2002.

[4] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," *ACM Conference on Computer and Communications Security*, pp. 148-160, 2002.

[5] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public physically unclonable functions: architecture and applications," *Design Automation Conference*, pp. 242-247, 2011.

[6] J. Guajardo, S. Kumar, G. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection, *CHES*, pp. 6380, 2007.

[7] N. Beckmann, M. Potkonjak, "Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions," Information Hiding: 11th International Workshop 2009, pp. 206-220, Darmstadt, Germany, 2009.

[8] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," *ICCAD*, pp. 670-673, 2008.

[9] T. Xu, M. Potkonjak, "Lightweight digital hardware random number generators," *IEEE SENSORS*, pp. 1-4, 2013.

[10] T. Xu, J. B. Wendt, and M. Potkonjak, "Matched Digital PUFs for Low Power Security in Implantable Medical Devices" to appear in *IEEE International Conference on Healthcare Informatics*, 2014.

[11] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, V. van der Leest, "Experimental Evaluation of Physically Unclonable Functions in 65 nm CMOS, *European Solid-State Circuits Conference - ESSCIRC 2012*, IEEE, 2012.

[12] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," *2008 IEEE International Test Conference*, pp. 1-10, October 2008.

[13] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of Physical Unclonable Functions," *IACR ePrint 657*, 2011.

[14] "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute of Standards and Technology (NIST) Special Publication 800-22, Rev. 1a, Apr. 2010.

[15] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Transations on VLSI Systems*, vol. 14, no. 5, pp. 501-513, 2006.