Design and Evaluation of a Delay-Based FPGA Physically Unclonable Function

Aaron Mills, Sudhanshu Vyas, Michael Patterson, Christopher Sabotta, Phillip Jones, Joseph Zambreno
Electrical and Computer Engineering, Iowa State University, Ames, Iowa, USA
Email: {ajmills, spvyas, mjpatter, csabotta, phjones, zambreno}@iastate.edu

***Abstract*—A new Physically Unclonable Function (PUF) variant was developed on an FPGA, and its quality evaluated. It is conceptually similar to PUFs developed using standard SRAM cells, except it utilizes general FPGA reconfigurable fabric, which offers several advantages. Comparison between our approach and other PUF designs indicates that our design is competitive in terms of repeatability within a given instance, and uniqueness between instances. The design can also be tuned to achieve desired response characteristics which broadens the potential range of applications.**

I. Introduction

A PUF is a device whose transfer function exploits physical phenomena in a way that cannot be replicated, even if the full design is known. It is also referred to as a physical one-way hash function when implemented in a *challenge-response framework*. PUFs reduce the ability of attackers to circumvent security mechanisms, as these mechanisms are implemented in tamper-resistant hardware [1, 2] rather than software. The devices are unclonable in the sense that, although they may be physically copied, this provides no advantage to an attacker, because each copy will behave differently.

A variety of PUF designs have appeared over the past decade–roughly one each year since 2000 [3]. The most well-known delay-based PUFs are SRAM [4, 5], Butterfly [6], Ring Oscillator (RO) [7, 8], and Arbiter [9, 10] and the FPGA-specific Anderson PUF [11]. A more thorough survey that includes some non-delay-type PUFs appears in [12].

The PUF designed for this paper is a delay-type PUF that inherits the SRAM and Butterfly PUFs' properties. Briefly, compared to the most popular designs, the one proposed:

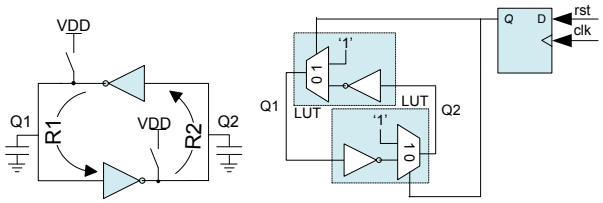
- Uses FPGA reconfigurable fabric, making testing easier than for SRAM PUFs. The circuit can be reset without requiring the whole device to be reset.
- Is simpler than the non-linear Arbiter PUFs. In addition, unlike the arbiter PUF there have been no known model-building attack against PUFs which are based on independent cells.
- Intuitively has a lower power requirement than the RO PUF, since it does not oscillate.

A disadvantage is size: an n-bit response requires at least n cells. Additionally in their raw form this kind of PUF tends to suffer from a slightly higher-than-average level of noisiness.

II. Design and Implementation

A. Principle of Operation

A memory-type PUF uses a small cell whose single-bit value is not known until it has stabilized after reset. For example, in Fig. 1a, if both switches are initially closed, then the capacitors at Q1 and Q2 are both charged. Then at t=0,



(a) Conceptual (b) Logical

Fig. 1: PUF Design

the switches are opened and the circuit resolves to a stable state which depends on the delay through the inverter and interconnects, as well as the switching threshold of the logic. If route R1 has a shorter delay, it will remain at logical 1 and force Q2 to logical 0. If route R2 has a shorter delay, it will remain at logical 1 and force Q1 to logical 0. Thus Q1 and Q2 will always resolve to opposing values, but which has which value depends on the physical properties of the hardware. The PUF design is shown in Fig. 1b. The switches are replaced by combinational logic implemented in LUTs.

The function of the PUF based on the delay of a net d_N can be described as $d_N = d_S + d_R + d_{NOISE}$. d_S represents the static delay estimated by the design tool. d_R is a random variable representing the uncertainty in net delay due to process variation. Finally, d_{NOISE} is a dynamic random variable representing the effects of temperature and voltage variation as well as interaction between circuits. Both d_R and d_{NOISE} may be either negative or positive.

Next, we can characterize the delay of the two nets, Q1 and Q2. Both d_{L1} and d_{L2} are additional random variables representing the delay through the two LUTs employed by a cell to create the necessary logic. They are always positive.

$$d_{q1} = d_{L1} + (d_{Sq1} + d_{Rq1} + d_{NOISEq1}) \tag{1}$$

$$d_{q2} = d_{L2} + (d_{Sq2} + d_{Sq2} + d_{NOISEq2}) \tag{2}$$

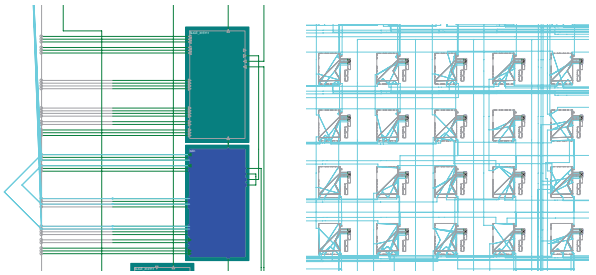
Ideally, the quantity $\Delta d_S = (d_{Sq1} - d_{Sq2})$ should be 0 so that the effect of the random LUT and route delay components dominate. The difference between the delay of the routes, Δd (5), dictates the circuit outcome.

Finally, the PUF can be described by a pair of piecewise functions depending on Δd.

$$Q1 = \begin{cases} 1 & : \Delta d < 0 \\ 0 & : \Delta d > 0 \end{cases} \tag{4a}$$

$$Q2 = \begin{cases} 1 & : \Delta d > 0 \\ 0 & : \Delta d < 0 \end{cases} \tag{4b}$$

$$\Delta d = (d_{L1} - d_{L2}) + (d_{Sq1} - d_{Sq1}) + (d_{Rq1} - d_{Rq2}) + (d_{NOISEq1} - d_{NOISEq2}) = \Delta d_L + \Delta d_S + \Delta d_R + \Delta d_{NOISE} \quad (5)$$



(a) Single-slice PUF cell (b) Portion of 128 cell PUF Array

Fig. 2: PUF layout in FPGA Editor

These functions are not completely defined. If Δd_R is 0, or close to it ("critically matched"), Q1 and Q2 become random variables due to Δd_{NOISE}. This is metastability.

A few properties can be deduced from this analysis:

1) Δd_S should be as close to 0 as possible.
2) Δd_{NOISE} should be as close to 0 as possible.
3) $\Delta d_R \gg \Delta d_S$ and $\Delta d_R \gg \Delta d_{NOISE}$ for reliability.

III. Implementation Details

Ensuring symmetrical routing and identical PUF instances on an FPGA is challenging. The Xilinx toolset includes FPGA Editor which enables creation of *hard macros*. Figure 2a shows a single PUF instance implemented as a hard macro on a Spartan 3 FPGA. Figure 2b shows a grid of such instances. Applying mapping constraints and hard macros maintains pin placement across instances–a potential delay variable [13].

IV. Experimentation and Evaluation

A. PUF Properties

For evaluation the following properties are assumed:

- Reliability: the output of the PUF is consistent.
- Uniformity: there is an equal distribution of 1's and 0's in the output. Also called randomness [10].
- Uniqueness: For two instances of a PUF, the responses to the same challenge are very different.

An additional property introduced in [14], is used:

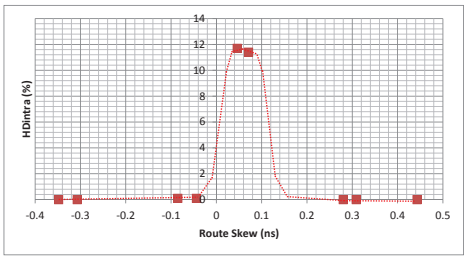
- Autocorrelation: the response bits are uncorrelated.

B. Effect of Routing Skew

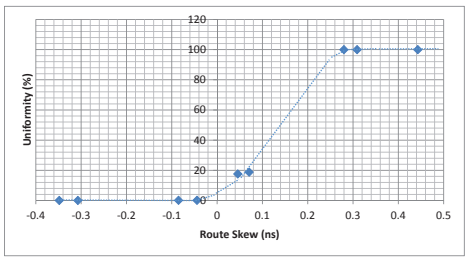
An experiment was performed on a Spartan 3E to verify the delay equations. The lengths of the routes Q1 and Q2 were adjusted by changing the LUT pins that were used, portrayed in Fig. 4. Since the pins of the reset lines were kept fixed, and each LUT has 4 inputs, a total of 9 routing configurations were possible. The configured PUFs were read 100 times each to obtain each data point. The results are shown in Fig. 3.

The skew values shown in Fig. 3 were calculated from the static delay analysis tool. The values for HD_{INTRA}, or the error rate, and Uniformity, or the ratio of 1's to 0's, were gathered empirically. Both of these terms are defined formally in Section IV-D.

The apparent systematic offset from 0 ns in Fig. 3 may exist for two reasons. First, there could be a systematic error in the static timing analysis. Second, as reported by the timing tool, there does exist a small fixed skew on the reset lines for each configuration, which will create a bias in the response.

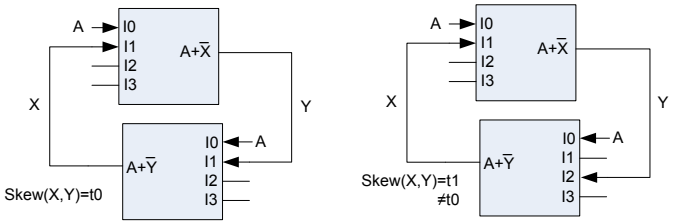


(a) The highest error rate is seen when the routes are nearly matched, since noise tends to dominate under such conditions. Outside this region, the error rate is 0 since the response is fixed.



(b) As the skew increases, the Uniformity saturates at 100% (1's dominate) or 0% (0's dominate).

Fig. 3: Effect of Routing Skew on HDINTRA and Uniformity



(a) LUTs implementing the same function are used to create a combinational loop

(b) Due to FPGA routing constraints, and the delay change inside the LUT itself, the skew (and behavior) may be radically different from (a)

Fig. 4: Effect of LUT Input Selection on Route Skew

The data that was gathered appears to support the delay model. It is evidently not easy to optimize both HD_{INTRA} and Uniformity. For example, one configuration achieves an error rate of 0% with a bias towards generating 1's. Another achieves a 0% error rate with a bias towards generating 0's. The configuration used for analysis was selected to avoid these static behaviors. A more optimal configuration might be available if an exhaustive search through the set of pin permutations were performed, providing greater resolution. This would include the permutation of the reset pins, which were fixed. FPGAs using LUTs with more input pins would provide even more data. Despite these limitations, the experiment performed demonstrates the impact that tiny changes in routing can have on PUF behavior, and the window of routing skew for which the circuit will function as a PUF.

C. Error Correction

Similar to biometrics, PUFs typically require some form of error correction to increase reliability. A simple scheme used here is the majority vote. Two forms are known as temporal majority voting, and spatial majority voting [15].

Temporal majority voting (TMV) is also sometimes referred to as the *repetition code*. It involves making an odd-numbered N_T readings of the PUF, and then determining how many 1's

TABLE I: Design Reliability

Device	HD_{INTRA}	σ	$HD_{INTRA}\prime$	$\sigma\prime$
1	13.5	1.3	5.9	1.9
2	13.0	1.3	6.0	3.0
3	12.1	3.2	8.1	2.6
Average	12.9	1.9	6.8	2.5

were read. If more than $M_T = \frac{N_T-1}{2}$ 1's are read, the output is considered a 1. This acts as a low-pass filter for PUF bits that occasionally toggle. It improves the reliability, but at the expense of the process taking N_T times longer.

Spatial majority voting (SMV) involves logically grouping N_S PUFs for the purpose of generating a single bit. Each PUF in a group produces a bit in parallel, and if the number of 1's exceeds threshold M_S, the overall group is considered to have produced a 1. This form of majority voting helps to move the distribution of 1's and 0's in a string to uniformity. Like TMV, there is a trade off in that N_S times as many cells are required.

The post-processing that is applied in the following sections below first applies TMV with $N_T = 3$ and $M_T = 1$, and then applies SMV with $N_S = 2$ and $M_S = 0$. A number of potential configurations were compared and these values appeared to produce the most improvement in both Repeatability and Uniformity, while limiting overhead.

D. Suitability as a PUF

Each of a collection of three Spartan 3E FPGAs were divided into 8 regions. In each region a PUF array is tested.

1) Reliability: The ideal PUF should exhibit perfectly consistent, or reliable, output for a given instance. The extent to which a PUF deviates from this property can be called its error rate. A simple way to express the error rate, as defined by [14], calculates the average intra-chip Hamming Distance (HD) between a series of samples, for a particular PUF instance i. A baseline n-bit response R_i is extracted from the circuit, and compared to m further samples. The expression to obtain a single value based on a set of intra-chip HD[1] calculations is shown below.

$$HD_{INTRA} = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R'_{i,t})}{n} \times 100\% \quad (5)$$

For this experiment, m, the number of repetitions, is 100. Table I shows the experiment results. In the ideal case this value is 0%. The raw results show a relatively high error rate, typical for memory PUFs [14]. With post-processing ($HD_{INTRA}\prime$) the error rate is improved. The standard deviation with ($\sigma\prime$) and without (σ) postprocessing are also provided.

2) Uniformity: There should be a uniform distribution of '0's and '1's in a given response r for PUF instance i. This metric can be expressed as follows.

$$U_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \quad (6)$$

An ideal PUF shows a uniformity of 50%. Since the PUF error rates are non-zero, the Uniformity is averaged over 100 responses. The results are shown in Table II. For the raw circuit, the Uniformity is low suggesting bias towards producing 0's. After post-processing ($U_i\prime$), the output is closer to uniformity, with a small bias towards producing 1's.

[1]This value is sometimes referred to as μ_{intra}

TABLE II: Design Uniformity

Device	U_i	σ	$U_i\prime$	$\sigma\prime$
1	22.4	3.1	60.1	6.0
2	24.2	2.1	59.0	2.9
3	17.8	5.4	48.7	11.8
Average	21.5	3.5	55.8	6.9

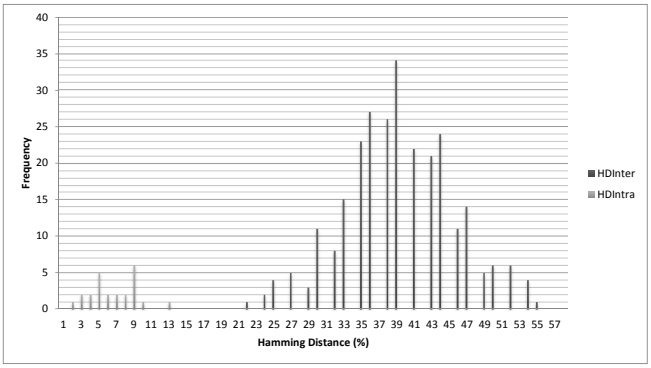


Fig. 5: HDInter vs HDIntra

3) Uniqueness: The uniqueness property is that a PUF copied to another chip should produce a signature with a Hamming Distance of near 50%, which means half the bits are different. The following equation can be applied to determine the uniqueness of a PUF across a population of k chips using pairwise calculations of HD, called HD_{inter}[2].

$$Uniqueness = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{HD(R_i, R_j)}{n} \times 100\% \quad (7)$$

For this experiment, first pairwise comparisons are performed to determine the HD between all circuit instances. With post-processing Uniqueness shifts closer to the ideal 50% average HD–from 34.2 to 47.8.

In Fig. 5, the histograms of HD_{INTER} and HD_{INTRA} are directly compared. The fact that they do not overlap is significant–it indicates that within this population of 24 PUFs it is possible to distinguish between a given PUF's noisy response, and the response of other PUFs. This means it is possible to implement a detection algorithm to identify a given device. We can also estimate the minimum number of unique IDs that could be generated based on the test results. The lowest HD_{INTER} that was observed was 22%, suggesting that in such a case around 28 bits were different between the two signatures. The number of IDs whose HD is 28 from a reference ID is $C(128, 28)$, or around 1.3×10^{28}.

4) Correlation Between Bits: The autocorrelation test [14] can be used to detect correlation between bits. Systematic aspects to process variation may show up as significant correlation at particular intervals. Because signatures are extracted from a common fabric it is possible for spatial correlation to appear due to gradients, seen in [13]. The equation is shown.

$$R_{xx}(j) = \sum_{t=1}^{n} x_t x_{t-j} \quad (8)$$

The test is performed on a signature extracted from each device. The 0's in each signature were replaced by -1, so that

[2]This value is sometimes referred to as μ_{inter}

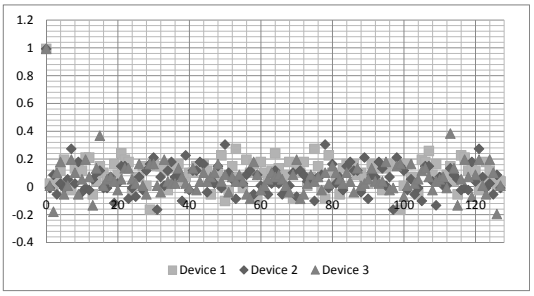


Fig. 6: Autocorrelation

TABLE III: Performance Comparison

PUF	Experiment	Reliability	Uniqueness
Proposed	24 128-bit arrays	6.8%	47.8%
Optical [1]	567 CRPs, 4 tokens	25.25%	49.79%
Coating [2]	31 CRPs, 36 ASICs	$< 5\%$	$\sim$50%
Basic Arbiter [9]	10k CRPs, 37 ASICs	$< 5\%$	50%
Feed-forward Arbiter [9]	10k CRPs, 37 ASICs	9.8%	38%
Basic RO [16, 17]	Many CRPs, 4 FPGAs	$\sim$0.01%	$\sim$1%
RO with comparator [8]	1k loops, 15 FPGAs	0.48%	46.14%
SRAM [18]	65k CRPs on different FPGAs	$<$12%	49.97%
Butterfly [6]	64 CRPs, 36 FPGAs	$<$6%	$\sim$50%
Anderson [11]	36 128-bit arrays	3.6%	$\sim$48%

a correlation value of 0 indicates no correlation, and the values 1 and -1 indicate total correlation. Figure 6 shows the results.

The autocorrelation for interval length 0 is 1 for each device, since each bit is totally correlated to itself. The results show no clear patterns among the three devices, suggesting that the physical configuration of the cells does not have a strong impact on the signature. This appears to contradict the results in [14], but the array used in that experiment is much larger, so the effect of systematic process variation are more pronounced.

E. Performance Comparison

Due to the wide variety of testing procedures it is still difficult to make direct comparisons between designs. Some authors report results in raw PUF form, while others report results with post-processing applied. Also, some works perform experiments based from the perspective of Challenge-Response Pairs (CRP). Despite these limitations Table III demonstrates that the proposed design is competitive. It is also worth mentioning that the design is unoptimized, since a comprehensive examination of the effect of LUT pin choice on Uniqueness and Reliability was not performed.

F. Design Applicability

Applications related to secret-key generation in cryptography [5, 19] require the error rate should be very close to 0. However, applications related to authentication [20, 21] and signature generation [22] are more tolerant to errors. The design would be best relegated to such an application. In particular the low power consumption makes it suitable for embedded applications such as RFID or security tokens.

Another potential application is a random number generator. As discussed in [23] a linear-feedback shift register (LFSR) can be added to a PUF, using it as a seed. The relative noisiness of the raw design could be leveraged in such a way. The LUT pin configurations could be tuned to achieve good uniformity but high error rate.

V. Conclusions and Future Directions

Several qualitative tests have yet be performed on the PUF design presented in this paper, such as measuring behavior in the presence of varying temperature and voltage. It is expected that this design will exhibit behavior similar to existing designs, i.e. a slight increase in error rate. Future work should also see a shift in focus to testing the design in higher-level applications, such as a challenge-response framework. The ability to adjust the relative lengths of the PUF routing is a unique asset in that it can potentially be used to optimize the circuit for different applications.

VI. Acknowledgement

The travel grant to participate in the Embedded Systems Challenge were provided in part by The National Science Foundation (0958510,1059328), Army (W911NF-11-1-0470), Air Force Research Labs and Intel. The FPGA platforms for the contest were donated by Xilinx.

References

[1] R. Pappu, R. Recht, and J. Taylor, "Physical one-way functions," *Science*, pp. 2026–2030, Sep. 2002.
[2] P. Tuyls *et al.*, "Read-proof hardware from protective coatings." in *CHES*, 2006, pp. 369–383.
[3] A. Maiti, V. Gunreddy, and P. Schaumon, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *IACR*, 2011.
[4] M. Hofer and C. Bohm, "An alternative to error correction for SRAM-like PUF," in *CHES*, 2010, p. 335350.
[5] J. Guajardo *et al.*, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *FPL*, Aug. 2007, pp. 189–195.
[6] S. Kumar *et al.*, "The butterfly PUF: protecting IP on every FPGA," in *HOST*, Jun. 2008, pp. 67–70.
[7] S. Morozov, A. Maiti, and P. Schaumon, "An analysis of delay based PUF implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications*, 2010, pp. 382–387.
[8] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC*, Jun. 2007, pp. 9–14.
[9] D. Lim *et al.*, "Extracting secret keys from integrated circuits," *VLSI Systems*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.
[10] Y. Hori *et al.*, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *ReConFig*, Dec. 2010, pp. 298–303.
[11] J. Anderson, "A PUF design for secure FPGA-based embedded systems," in *ASP-DAC*, Jan. 2010, pp. 1–6.
[12] 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, Security and Cryptology*, 2010.
[13] P. Sedcole and P. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *FPT*, 2006.
[14] C. Bohm, M. Hofer, and W. Pribyl, "A microcontroller SRAM-PUF," in *NSS*, Sep. 2011, pp. 269–273.
[15] R. Maes, P. Tuyls, and I. Verbauwhed, "Intrinsic PUFs from flip-flops on reconfigurable devices," in *WISSec*, 2008.
[16] B. Gassend, "Physical Random Functions," Master's thesis, MIT, MA, USA, 2003.
[17] B. Gassend *et al.*, "Silicon physical random functions," in *CCS*, 2002, pp. 148–160.
[18] J. Guajardo *et al.*, "FPGA intrinsic PUFs and their use for IP protection," in *CHES*, 2007, pp. 63–80.
[19] S. Goren *et al.*, "FPGA bitstream protection with PUFs, obfuscation, and multi-boot," in *ReCoSoC*, Jun. 2011, pp. 1–2.
[20] P. F. Cortese *et al.*, "Efficient and practical authentication of PUF-based RFID tags in supply chains," in *RFID-TA*, Jun. 2010, pp. 182–188.
[21] K. Yang *et al.*, "Puf-based node mutual authentication scheme for delay tolerant mobile sensor network," in *WiCOM*, Sep. 2011, pp. 1–4.
[22] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *CHES*, Sep. 2006, pp. 311–323.
[23] L. Kulseng *et al.*, "Lightweight mutual authentication and ownership transfer for RFID systems," in *INFOCOM*, Mar. 2010, pp. 1–5.