An Analysis of Delay Based PUF Implementations on FPGA

Sergey Morozov, Abhranil Maiti, and Patrick Schaumont

Virginia Tech, Blacksburg, VA 24061, USA
{morozovs,abhranil,schaum}@vt.edu

Abstract. Physical Unclonable Functions promise cheap, efficient, and secure identification and authentication of devices. In FPGA devices, PUFs may be instantiated directly from FPGA fabric components in order to exploit the propagation delay differences of signals caused by manufacturing process variations. Multiple delay based PUF architectures have been proposed. However, we have observed inconsistent results among them. Ring Oscillator PUF works fine, while other delay based PUFs show a significantly lower quality. Rather than proposing complex system level solutions, we focus on the fundamental building blocks of the PUF. In our effort to compare the various delay based PUF architectures, we have closely examined how each architecture maps into the FPGA fabric. Our conclusions are that arbiter and butterfly PUF architectures are ill suited for FPGAs, because delay skew due to routing asymmetry is over 10 times higher than the random variation due to manufacturing process.

1 Introduction

A Physical Unclonable Function (PUF) has the unique advantage of generating volatile chip-specific signatures at runtime. It not only excludes the need of an expensive non-volatile memory for key storage, but also offers robust security shield against attacks. It is emerging as a promising solution to issues like intellectual property (IP) protection, device authentication, and user data privacy by making device specific signatures possible.

The majority of the PUF designs are based on delay variation of logic and interconnect. The fundamental principle followed in these delay-based PUF is to compare a pair of structurally identical/symmetric circuit elements (composed of logic and interconnect), and measure any delay mismatch that is introduced by the manufacturing process variation, and not by the design.

We will show that Arbiter PUF and Butterfly PUF are inherently difficult to implement on FPGA due to the delay skew present between a pair of circuit elements that are required to be symmetric in these PUFs. This static skew is an order of magnitude higher than the delay variation due to random process variation. Our main contribution in this paper is to present the complexities in implementing two PUFs on a 90nm commodity FPGA platform.A more detailed technical report discussing the root causes of these complexities and details of our implementations is available in [6].

2 Background

A PUF is a function that generates a set of responses while stimulated by a set of challenges. It is a physical function because the challenge-response relation is defined by complex properties of a physical material, such as the manufacturing variability of CMOS devices. Its unclonability is attributed to the fact that these properties cannot be controllably reproduced, making each device effectively unique. Though many PUF architectures have been proposed, we focus on the categories of PUF based on the delay variation of logics and interconnects, specifically the arbiter PUF (APUF) and the Butterfly PUF (BPUF). In the technical report [6], these architectures are also compared with the ring oscillator PUF architecture [5].

Arbiter PUF - An APUF, proposed by Lim et.al [2], is composed of two identically configured delay paths that are stimulated by an activating signal(Fig. 1(a)). The difference in the propagation delay of the signal in the two delay paths is measured by an edge triggered flip-flop known as the arbiter. Several PUF response bits can be generated by configuring the delay paths in multiple ways using the challenge inputs.

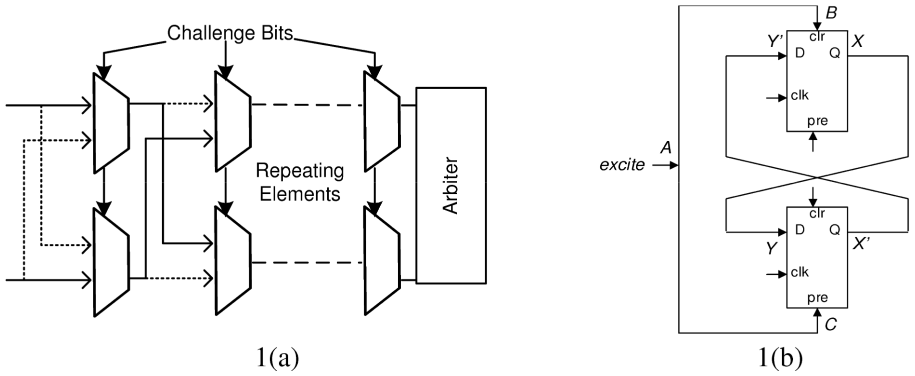


Fig. 1. (a). The Arbiter PUF structure. The symmetric pairs of components are highlighted with matching patterns. (b) A BPUF cell with two cross coupled latches.

Butterfly PUF - The Butterfly PUF, proposed by Kumar et.al [3], is a technique that aims to emulate the behavior of an SRAM PUF [1]. However, the functionality of this PUF is based on the delay variations of interconnects. A BPUF cell employs two cross-coupled latches, and exploits the random assignment of a stable state from an unstable state that is forcefully imposed by holding one latch in preset while the other in clear mode by an excite signal (Figure 1(b)). The final state is determined by the random delay mismatch in the pair of feedback paths and the excite signal paths due to process variation.

The equation for delay d of a net N in a circuit is shown in Equation 1, where d_S is the static delay as determined by the static timing analysis tools, and d_R is the random delay component due to process variation.

$$d_{\mathrm{N}} = d_{\mathrm{S}} + d_{\mathrm{R}} \tag{1}$$

The delay difference between two nets, N_1 and N_2 , in a circuit maybe be expressed as a sum of static delay difference Δd_{S} and random delay difference Δd_{R} [6] as shown in Equation 2.

$$\Delta d = d_{\mathrm{S1}} - d_{\mathrm{S2}} + d_{\mathrm{R1}} - d_{\mathrm{R2}} = \Delta d_{\mathrm{S}} + \Delta d_{\mathrm{R}} \tag{2}$$

A delay-based PUF circuit involves extraction and comparison of the random delay, d_{R} while minimizing Δd_{S}. In the ideal case for a delay based PUF, $\Delta d_{\mathrm{S}} \rightarrow 0$ and the delay skew is purely a function of the random delay component. However, typically the output of a given PUF structure will be at least partially dependent on Δd_{S}, causing the output to be biased. Further, if $\Delta d_{\mathrm{S}} > \Delta d_{\mathrm{R}}$, the effect of random variation becomes insignificant, and the output of the PUF structure becomes static regardless of d_{R}. The effectiveness of the PUF depends on how much symmetry we can achieve between a particular pair of elements in order to minimize the effect of Δd_{S}. This symmetry requirement determines the implementation complexity of a PUF on FPGA.

In Figure 1(a), the basic structure of APUF, is shown. The pairs of nets connected to the multiplexers (pairs shown with different patterns) need to be symmetric in order to minimize Δd_{S}. In figure 1(b), a BPUF cell is presented. For a functional BPUF, the pair of nets AB/AC as well as XY/X'Y' need to be symmetric along with the latches.

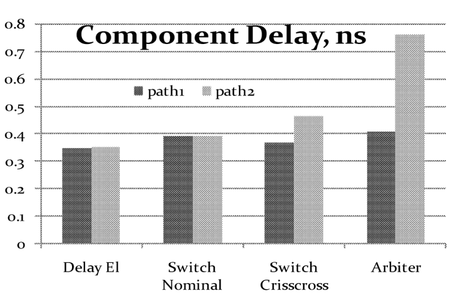
3 PUF Architecture in FPGA

The Xilinx Spartan3E FPGA device that we used as a platform in our experiments is made up of Configurable Logic Blocks (CLBs) surrounded by a sea of interconnect. Inside the CLB, there are 4 slices which contain configurable look-up tables and flipflops. In this section, we examine the mapping of PUF elements into this structure.

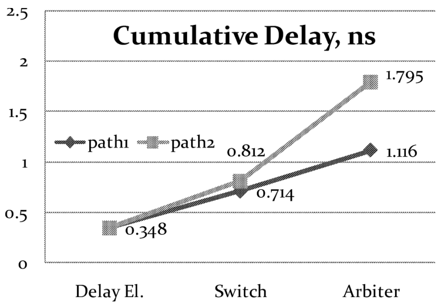
3.1 Arbiter PUF

The two primary components required for the Arbiter PUF architecture are the switches and the arbiter itself. A useful secondary component is a delay element: trivial logic that inserts additional delay into the paths. We used identical 2-input MUXes in two different slices for the switches. The arbiter was instantiated as a positive clock-edge triggered flipflop in a slice. A look-up table in a slice served as the delay element. We examined several different mapping schemes for these components [6].

Using timing analysis tools, we observed the routing delay caused by each component. This value does not account for the manufacturing variability. Therefore, we hoped to find identical static delays for symmetric routes. Figure 2(a) shows the delay caused by each component for a possible route. There are two values for the switch component: Switch Nominal is the delay of the signals when the



2(a)



2(b)

Fig. 2. (a) Individual delays of each component. (b). Cumulative delay of the path through after each component.

paths are straight, Switch Crisscross is the delay of the signal when the paths are crossed. Figure 2(b) shows the cumulative delay of the signal propagated along the route when the switches are crossed.

These results indicate that this PUF structure will not function. The 3σ value of delay variation due to process variability in 90 nm technology has been estimated to be approximately 3.5% [4]. On the other hand, we observe that in this case, variation due to routing is much higher than expected variation due to process variability: $d_{\mathrm{S}}/d_{\mathrm{R}}$ is 25.6 times.

There are two causes of routing variation in this design: the asymmetric routes for the switch crisscrossing paths and the much more dramatic asymmetric routes to the arbiter. The larger difference in arbiter routes is due to the fact that routing to a CLK input of a flipflop requires sending the signal through multiple additional segments to reach the CLK port, whereas the route to the D input of the flipflop is comparatively simple.

All attempted mapping schemes produced similar results. While some directions significantly reduced the routing delay, a difference of at least 100 ps remained. Figure 3 shows the delays we observed for the arbiter component in each direction, depending on the location of the arbiter, and the mapping scheme. Under the best possible conditions of 2 CLB mapping and West to East placement, we observe $\Delta d_{\mathrm{S}}/\Delta d_{\mathrm{R}}$ to be 11.6 times.These results indicate that additional delay due to the complexity of routing a signal to the CLK input of a flipflop cannot be avoided using current routing schemes and architectures. Asymmetry in routing of crisscrossing switch routes is also present, but that delay difference is dwarfed by the arbiter.

3.2 Butterfly PUF

Though the pair of latches can be safely assumed to be identical in a Butterfly PUF, the real design challenge comes when a designer has to ensure that the

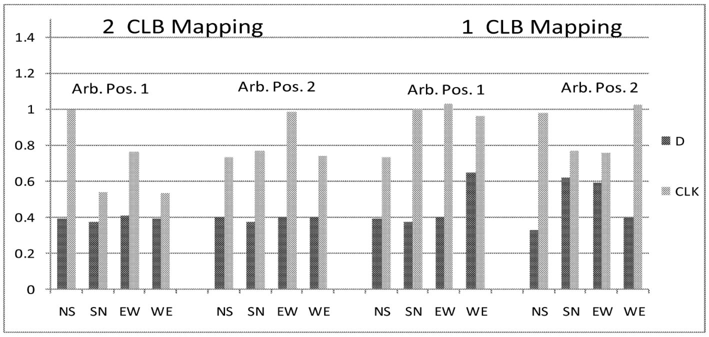
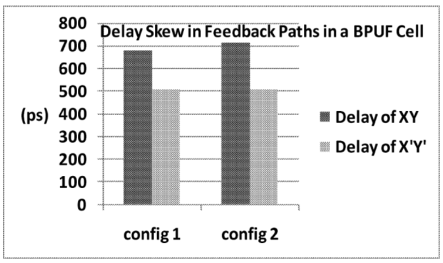
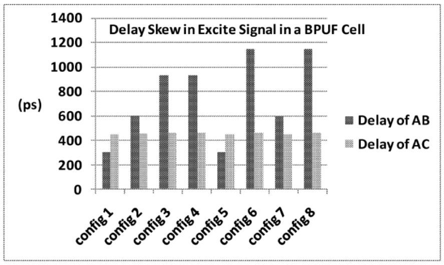


Fig. 3. Delay difference in routing to the D input and the CLK input of a slice under various conditions. NS - North to South layout; SN - South to North; EW - East to Wes; WE - West to East.



(a)



(b)

Fig. 4. (a) The delay values of nets XY and X'Y'. Two configurations correspond to two possible placements of the pair of latches inside a CLB. (b) The delay values of nets AB and AC. Eight configurations correspond to four possible placements of the excite signal buffer in a CLB for each of the two possible placements of the latches in the CLB.

interconnection pairs (XX'/YY' and AB/AC) are symmetric. Since no layout level information inside the switch box is available, we depend on the static delay values provided by the design tool. In Figure 4, we present a set of data showing the delay skew in the pair of nets AB/AC and XY/X'Y' (refer to figure 2(b)) as a result of automatic routing with timing constraint.

For the delay skew in XY/X'Y' net pair, the minimum value of the ratio $\Delta d_S/\Delta d_R$ is estimated to be 17 whereas the same quantity for the delay skew in AB/AC is 16. Since Δd_S is an order of magnitude higher than Δd_R, it is obvious that this PUF implementation will produce highly biased outputs. Even with the manual routing, it has been observed that the delay of a pair of interconnects do not match based on the static delay value. From the results it is evident that BPUF cell suffers from the asymmetric nature of the FPGA routing resources.

Thus, our observations contradict the results presented in [3]. However, we note that our experiments have been done on a Spartan-3E FPGA, and not on a Virtex-5 as used in [3].

4 Conclusion

In this work, we have analyzed how the peculiarities of FPGA routing affect the implementations of delay based PUFs. Our results show that symmetry requirements for Arbiter and Butterfly PUF architectures cannot be satisfied using available FPGA routing schemes, despite the apparent routing flexibility of FPGA devices. Using the best possible routing, the delay difference due to static variation routes is an order of magnitude higher than expected delay variation due to manufacturing variability. Yet an architecture without the mirror symmetry requirement, such a Ring Oscillator based PUF, can produce a working PUF. Ultimately, understanding how a particular PUF architecture maps into FPGA fabric allows us to select a promising architecture for further investigation and characterization of PUF circuits in FPGAs.

Acknowledgement

This work was supported in part by NSF grant no 0855095.

References

1. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Cryptographic Hardware and Embedded Systems (2007)
2. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., Van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2005)
3. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: IEEE International Workshop on Hardware-Oriented Security and Trust, HOST (2008)
4. Sedcole, P., Cheung, P.Y.K.: Within-die delay variability in 90nm FPGAs and beyond. In: Proceedings of IEEE International Conference on Field Programmable Technology (2006)
5. Suh, G.E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: Proceedings of Design Automation Conference (2007)
6. Morozov, S., Maiti, A., Schaumont, P.: A Comparative Analysis of Delay Based PUF Implementations on FPGA. IACR ePrint tbd/2009 (submitted December 19, 2009)