FPGA Time-Bounded Unclonable Authentication

Mehrdad Majzoobi, Ahmed Elnably, and Farinaz Koushanfar

Rice University, Electrical and Computer Engineering, Houston TX 77005, USA
{mehrdad.majzoobi,ahmed.elbanby,farinaz}@rice.edu

Abstract. This paper introduces a novel technique for extracting the unique timing signatures of the FPGA configurable logic blocks in a digital form over the space of possible challenges. A new class of physical unclonable functions that enables inputs challenges such as timing, digital, and placement challenges can be built upon the delay signatures. We introduce a suite of new authentication protocols that take into account non-triviality of bitstream reverse-engineering in addition to the FPGA's unprecedented speed in responding to challenges. Our technique is secure against various attacks and robust to fluctuations in operational conditions. Proof of concept implementation of the signature extraction and evaluations of the proposed methods are demonstrated on Xilinx Virtex 5 FPGAs. Experimental results demonstrate practicality of the proposed techniques.

1 Introduction

Any security mechanism is centered on the concept of a secret. Classic cryptography protocols relay on a secret key for reversing trapdoor functions. While such protocols are often secure against algorithmic attacks, it is well-known that the digitally stored secret keys can be attacked and cloned. Furthermore, conventional SRAM-based FPGAs suffer from major limitations in secret key storage since their fabrication technology cannot easily integrate non-volatile memory (NVM). Thus, the keys need to be stored on off-chip memory where communicating the keys to- and from- the off-chip components demands secure channels and additional protocols. On-chip NVM requires the overhead of a constant power source. Such a reliance not only incurs additional overhead, but increases the vulnerability to attacks.

Physical unclonable functions (PUFs) are an efficient enabling mechanism for many security applications [1,2]. PUFs exploit the secret information that inherently exists in the unclonable physical variations of the silicon devices. Moreover, PUFs provide a unique chip-dependent mapping from a set of digital inputs (challenges) to digital outputs (responses) based on the unique properties of the underlying physical device. PUFs can be employed to provide security at multiple levels and for addressing a range of problems in securing processors [3], software protection [4], IP protection [5], and IC authentication [6]. Even though a number of methods for realizing PUFs on FPGAs were proposed and implemented to date [5,7], the scope of the existing FPGA PUF methods is limited. Either they have a limited number of challenge-response pairs, or they are limited by the routing constraints on FPGAs, or they are vulnerable to learning and reverse engineering attacks.

R. Böhme, P.W.L. Fong, and R. Safavi-Naini (Eds.): IH 2010, LNCS 6387, pp. 1–16, 2010.

In this paper, we introduce a novel approach for implementing FPGA PUFs. In this proposed approach, first, the timings of each configurable logic block is accurately characterized for different combinations of inputs. Then a PUF is implemented such that a challenge to the PUF queries relationship of the clock pulses with respect to the delays for a subset of configurable blocks. Reproducing the responses requires the knowledge of the extracted delays as well as the structure and placement of the PUF circuit. We introduce a dynamic authentication protocol, where the placement of the PUF is randomly updated in each round of authentication, forcing the adversary to constantly reverse-engineer the configuration bit stream to discover the PUF structure and placement. The protocol is based on the fact that it is infeasible to reverse-engineer the configuration bit stream and generate the challenge-response signatures through simulation in a short time duration compared to evaluation on hardware. Our contributions are as follows:

- Introduction of structures for efficiently extracting the unclonable analog delay signatures of each of the FPGA configurable logic blocks over the possible inputs (challenges) using only commodity test equipments.
- A suite of new authentication protocols are built upon the extracted timing signatures.A new time-bounded protocol further takes advantage of the gap of between PUF simulation time and evaluation time on the authentic physical hardware to enhance the security of authentication against learning attacks.
- A linear calibration method is developed to compensate for the effects of variations in operating temperature and supply voltage to assure signature consistency.
- Experimental results and proof-of-concept implementation are demonstrated on Xilinx Virtex 5 FPGAs. The results show the applicability of the proposed methods, uniqueness of the signatures, and their robustness against the fluctuations in operational and environmental conditions.

The remainder of the paper is organized as follows. Section 2 summarizes the related literature. In Section 3 we present our method for timing signature extraction from a circuit for each possible challenges. Section 4 introduces time-bounded authentication protocol and its variants. Attacks and countermeasures are discussed in Section 4.3. Calibration for signature and response robustness against fluctuations in operational conditions is discussed in Section 5. Experimental evaluations are demonstrated in Section 6. Finally, we conclude in Section 7.

2 Related Work

The idea of using the complex unclonable features of a physical system as an underlying security mechanism was initially proposed by Pappu et al. [1]. The concept was demonstrated by mesoscopic physics of coherent light transport through a disordered medium. Another group of researchers observed that the manufacturing process variability present in modern silicon technology can be utilized for building a PUF. They proposed the arbiter-based PUF architecture which was based on the variations in CMOS logic delays, and demonstrated its implementation in the ASIC technology, and discussed a number of attacks and countermeasures [2,8,9,6,10]. In particular, they made the observations that the linear arbiter-based PUF is vulnerable to modeling attacks and proposed using nonlinear feed forward arbiters and hashing to safeguard against this attack [2]. Furthermore, they made the important observation that the PUF responses may not all be stable, and to alleviate the issue, proposed utilizing error correcting codes [11]. Further efforts were made to address the PUF vulnerability issues by adding input/output networks, adding nonlinearities to hinder machine learning and enforcing an upper bound on the PUF evaluation time [7,12,13,14,15]. The recent work in [7] demonstrated that even though successful ASIC implementation of arbiter PUF was shown, FPGA implementation of this PUF is not possible in the state-of-the-art technology. This is because of the routing constraints for implementing the similar parallel paths enforced by the regularities in the underlying FPGA fabric. For implementing PUFs on FPGA, Ring oscillator (RO) PUFs were proposed [6]. The major drawback of the RO PUFs is having only a quadratic number of challenges with respect to the number of ROs [12]. Furthermore, the ROs (while in use) consume significant dynamic power due to frequent transitions during oscillations. SRAM PUFs suffer from the same limitation in terms of the number of possible challenge combinations [12].

This paper presents the first practical method and proof of concept FPGA implementation of a PUF with exponential number of possible challenges of different kind including placement challenges. The new proposed PUF uses the unique cell-by-cell characteristics of the FPGA array. Note that time-bounded properties of FPGA PUFs was briefly mentioned by [7]. The authors in [13] exploited the time-boundedness characteristics of generic public key protocol by PUFs but no practical implementation options were discussed.

Besides the ongoing research on PUFs, several other relevant work on delay characterization serve as the enabling thrust for realization of our novel PUF structures. To perform delay characterization, Wong et al. in [16] proposed a built-in self-test mechanism for fast chip level delay characterization. The system utilizes the on chip PLL and DCM modules for clock generation at discrete frequencies.

3 Delay Signature Extraction

To measure the delays of components inside FPGA, we exploit the device reconfigurability to implement a delay signature extraction circuit. A high level view of the delay extraction circuitry is shown in Figure 1. The target circuit/path delay to be extracted is called the *Circuit Under Test (CUT).* Three flip flops (FFs) are used in this delay extraction circuit: *launch FF*, *sample FF*, and *Capture FF*. The clock signal is routed to all three FFs as shown on the Figure. Assume for now that the binary challenge input to the CUT is held constant and thus, the CUT delay is fixed.

Assuming the FFs in Figure 1 were originally initialized to zero, a low-to-high signal is sent through the CUT by the launch flip flop at the rising edge of the clock. The output is sampled T seconds later on the falling edge of the clock (T is half the clock period). Notice that the sampling register is clocked at the falling edge of the clock. If the signal arrives at the sample flip flop before sampling takes place, the correct signal value would be sampled, otherwise the sampled value would be different producing an error. The actual signal value and the sampled value are compared by an XOR logic and the result will be held for one clock cycle by the capture FF.

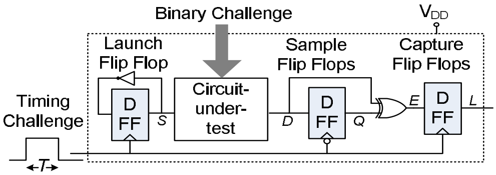


Fig. 1. The timing signature extraction circuit

A more careful timing analysis of the circuit reveals the relationship between the delay of the CUT (t_{CUT}), the clock pulse width (T), the clock-to-Q delay at the launch FF t_{clk2Q}, and the clock skew between the launch and sample FFs t_{skew}. The setup/hold of the sampling register and the setup/hold time of the capture register are denoted by t_{setS}, t_{holdS}, t_{setC}, and t_{holdC} respectively. The propagation delay of the XOR gate is denoted by t_{XOR}. The time it takes for the signal to propagate through CUT and reach the sample flip flop from the moment the launch flip flop is clocked is represented by t_P. Based on the circuit functionality in Figure 1, $t_P = t_{CUT} + t_{clk2Q} - t_{skew}$.

As T approaches t_P, the sample flip flop enters a metastable operation because of the setup and hold time violations and its output becomes nondeterministic. The probability that the metastable state resolves to a 0 or 1 is a function of how close T is to t_P. For instance, if T and t_{CUT} are equal, the signal and the clock simultaneously arrive at the sample flip flop and metastable state resolves to a 1 with a probability of 0.5. If there are no timing errors in the circuit, the following relationships must hold:

$$t_{holdC} < t_P < T - t_{setS} \tag{1}$$

$$t_P < 2T - (t_{setC} + t_{XOR}) \tag{2}$$

The errors start to appear if t_p enters the following interval:

$$T - t_{setS} < t_P < T + t_{holdS} \tag{3}$$

The rate (probability) of observing timing error increases as t_p gets closer to the upper limit of Equation 3. If the following condition holds, then timing error happens every clock cycle:

$$T + t_{holdS} < t_P < 2T - (t_{setC} + t_{XOR}) \tag{4}$$

Notice that in the circuit in Figure 1, high-to-low and low-to-high transitions travel through the CUT every other clock cycles. The propagation delay of the two differ in practice. Suppose that the low-to-high transition propagation delay ($t_p^{l \to h}$) is smaller than high-to-low transition propagation delay ($t_p^{h \to l}$). Then, for example if for low-to-high transitions, $t_p^{l \to h}$ satisfies Inequalities 1, 2 and for high-to-low transitions, $t_p^{h \to l}$ satisfies Inequality 4, timing errors happen only for high-to-low transitions and as a result timing error can only be observed 50% of the times. Figure 2 (a) illustrates this scenario.

Observability of timing errors follow a periodic behavior. In other words, if t_p goes beyond $2T - (t_{setC} + t_{XOR})$ in Inequality 4, the rate of timing errors begin to decrease again. However this time the decrease in the error rate is not a result of proper operation yet it is because the errors can not be observed and captured by the capture flip flip. Inequalities 5 and 6 correspond to the transition from the case where timing error happens every clock cycle Inequality 4) to the case where no errors can be detected Inequality 7.

$$2T - (t_{setC} + t_{XOR}) < t_P < 2T + (t_{holdC} - t_{XOR}) \quad (5)$$
$$t_P < 3T - t_{setS} \quad (6)$$

$$2T + (t_{holdC} - t_{XOR}) < t_P < 3T - t_{setS} \quad (7)$$

Timing errors no longer stay undetected if t_p is greater than $3T - t_{setS}$. Timing errors begin to appear and be captured if t_p falls into the following intervals:

$$3T - t_{setupS} < t_p < 3T + t_{holdS} \quad (8)$$
$$t_p < 4T - (t_{setupC} + t_{XOR}) \quad (9)$$

If the following condition holds, then timing error gets detected every clock cycle.

$$3T + t_{holdS} < t_p < 4T - (t_{setupC} + t_{XOR}) \quad (10)$$

This periodic behavior continues the same way for integer multiples of T, however it is physically upper bounded by the maximum clock frequency of the FPGA device. In general, if T is much larger than the XOR and flip flop delays, the intervals can be simplified to $n \times T < t_p < (n+1) \times T$ and timing errors can only be detected for odd values of n where n=0,1,2,3,...

In the rest of the paper, we refer to the characterization circuit that includes the as a *characterization cell* or simply a *cell*. Each cell in our implementation on FPGA is pushed into one configurable logic block (CLB). The circuit under test consists of four cascaded look-up tables (LUT) each implementing a variable delay inverter. We explain in Section 3.3 how the delay of the inverters can be changed.

3.1 Signature Extraction System

In this subsection, we present the system that efficiently extracts the probability of observing timing failure as a function of clock pulse width for a group of components on FPGA. The circuit shown in Figure 1 only produces a single bit flag of whether errors happen or not. We also need a mechanism to measure the rate or probability at which errors appear at the output of the circuit in Figure 1.

To measure the probability of observing error at a given clock frequency, an error histogram accumulator is implemented by using two counters. The first counter is the error counter whose value increments by unity every time an error takes place. The second counter counts the clock cycles and resets (clears) the error counter every 2^N clock cycles, where N is the size of the counters. The value of the error counter is stored in the memory exactly one clock cycle before it is cleared. The stored number of errors normalized to 2^N yields the error probability value. The clock frequency to the system is swept linearly and continuously in T_{sweep} seconds from $f_i = \frac{1}{2T_i}$ to $f_t = \frac{1}{2T_t}$, where $T_t < t_p < T_i$. A separate counter counts the number of clock pulses in each frequency sweep. This counter acts as an accurate timer that bookmarks when a timing error happened. The value of this counter is retrieved every time the number of counted errors are recorded (i.e. every 2^N cycles). A unique time stamp (T_m) can be calculated which indicates at what point during the sweep time a certain value appears in the clock counter. By knowing T_m, the frequency associated to the m-th error value

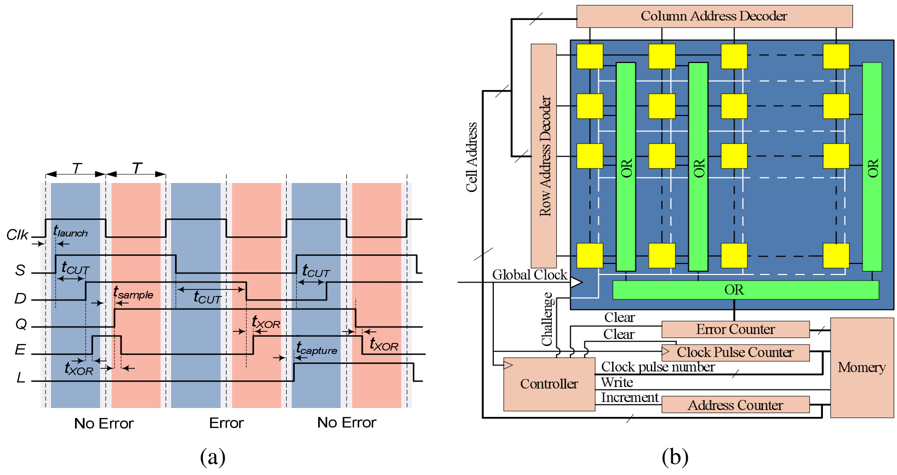


Fig. 2. (a)The timing diagram showing occurrence of timing error. (b) The architecture for chip level delay extraction of logic components.

can be easily calculated using the linear interpolation $f_m = (f_t - f_i) \times \frac{T_m}{T_{sweep}} + f_i$. The system shown in Figure 2 (b) is used for extracting the delays of an array of CUTs on the FPGA. Each square in the array represents the characterization circuit shown in Figure 1 and is referred to as a *cell* in the remainder of the text. Any logic configuration can be utilized within the CUT in the characterization circuit. In particular, the logic inside the CUT can be made a function of binary challenges, such that its delay varies by the given inputs. The system in Figure 2 (b) characterizes each cell by sweeping the clock frequency once. Then, it increments the cell address and moves to the next cell. The cells are characterized in serial. The row and column decoders activate the given cell while the rest of the cells are deactivated. Therefore, the output of the deactivated cells remain zero. As a result, the output of the OR function solely reflect the timing errors captured in the activated cell. Each time the data is written to the memory, three values would be stored: the cell address, the accumulated error value, and the clock pulse number at which the error has occurred. The clock counter is reset at each new sweep. The whole operation iterates over different binary challenges to the cells. Please note that the scanning can also be performed in parallel to save time.

3.2 Parameter Extraction

So far we have described the system that measure the probability of observing timing error for different clock pulse widths. The error probability can be represented compactly by a set of few parameters. These parameters are directly related to the circuit component delays and flip flop setup and hold time. It can be shown that the probability of timing error can be expressed as the sum of shifted Gaussian CDFs [7]. The Gaussian nature of the error probabilities can be explained by the central limit theorem. Equation 11 shows the parameterized error probability function.

$$f_{\mathbf{D},\Sigma}(t) = 1 + 0.5 \sum_{i=1}^{|\Sigma|-1} -1^{\lceil i/2 \rceil} Q(\frac{t-d_i}{\sigma_i}) \tag{11}$$

where $Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^{\infty} \exp\left(-\frac{u^2}{2}\right)$ and $d_{i+1} > d_i$. To estimate the timing parameters, f is fit to the set of measured data points (t_i,e_i), where e_i is the error value recorded when the pulse width equals t_i.

3.3 Challenge Configuration

To enable authentication, we require a mechanism for devising challenge inputs to the device and observing the device invoked responses. Fortunately, the capture flip flop yields a binary response. Assuming that the flip flop characteristics are known and constant, the response is a function of the clock pulse width T, and the t_{CUT}. Thus, one way to challenge the circuit and read its response out is to change the clock pulse width. The use clock pulse width has a number of implications. The response from the PUF will be deterministic if the T are either too high and too low. Predictability of responses makes it easy for the attacker to impersonate the PUF.

Another way to challenge the PUF is to alter the t_{CUT}. So far, we assumed that the delay of CUT is not changing which means the CUT have a specific configuration and specific input vector. Changing the input vector can alter the signal path delay, and hence the response. CUT is implemented by a set of LUT Figure 3 shows the internal circuit structure of an example 3-input LUT. In general, a Q-input LUT consists of 2^Q-1 2-input MUXs which allow selection of 2^Q SRAM cells. The SRAM cell values are configured to implement a pre-specific functionality. In this example, the SRAM cell values are configured to build a negated parity generation function of the three inputs to the LUT. If the number of ones in $A_1A_2A_3$ are even, then the output will be equal to 1. If the inputs A_1A_2 are held constant, the function $O = f(A_1)$ implements an inverter regardless of A_1A_2. However changing the input A_1A_2 can alter the delay of the inverter due to the changes in the signal propagation path inside the LUT and process-dependant variations in delay of paths with the same length. The LUTs in Xilinx Virtex 5 FPGAs, consist of 6 inputs. Five inputs of the LUT can be used to alter the inverter delay yielding 2^5 = 32 distinct delays for each LUTs.

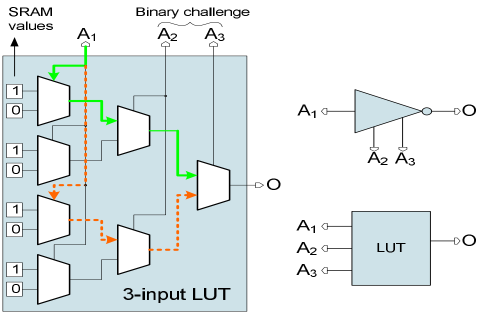


Fig. 3. The internal structure of LUTs. The signal propagation path inside the LUTs change as the inputs change.

4 Authentication

In this section, we show how the extracted cell characteristics in Section 3 can be utilized for FPGA authentication. The following terminology is used in the rest of the paper. The *verifier* (V) authenticates the *prover* (P) who owns the FPGA device. The verifier authenticates the device by verifying the unique timing properties of the device. The challenge vectors are denoted by c_i, $i = 1, \ldots, N$, and the corresponding responses are denoted by r_i, $i = 1, \ldots, N$. The PUF that performs the challenge to response transformation is denoted by $\mathcal{T} : \mathcal{T}(c_i) := r_i, i = 1, \ldots, N$.

4.1 Classic Authentication

The registration and authentication processes for the classic authentication case are demonstrated in the diagram in Figure 4 (a) and (b) (disregard the darker boxes for now). The minimum required assumptions for this case are (i) the verifier is not constrained in power (ii) it is physically impossible to clone the FPGA (iii) the characteristics of the FPGA owned by the prover is a secret only known to the prover and verifier.

As shown in Figure 4(a), during the registration phase, the verifier extracts and securely stores the cell delay parameters by performing characterization as explained in Sections 3. By knowing the FPGA-specific features in addition to the structure and placement of the configured PUF circuit, the verifier is able to predict the responses to any challenges to the PUF circuit. After registrations, the FPGA along with the pertinent PUF configuration bitstream is passed to the end-user.

At the authentication, end-user (prover) is queried by the verifier to make sure he is the true owner of the FPGA. Classic authentication is shown in Figure 4(b). To authenticate the ownership, the verifier utilizes a random seed and generates a set of pseudorandom challenge vectors for querying the prover. The prover responds to the challenges she receives from the verifier by applying them to the configured FPGA hardware. The verifier then compares the received responses from the prover with the predicted ones, and authenticates the chip if the responses are similar.

To ensure robustness against errors in measuring the delays and the change in delay measurement conditions, the registration entity may also compute the error correction information for the responses to the given challenges. To prevent information leakage via the error correction bits, secure sketch techniques can be used. A secure sketch produces public information about its input that does not reveal the input, and still permits exact recovery of the input given another value that is close to it [17].

The device is authenticated if the response after error correction would be mapped to the the verifier-computed hash of responses. Otherwise, the authentication would fail. Alternatively, the verifier can allow for some level of errors in the collected responses and remove the error correction and hashing from the protocol. However, accepting some errors in the responses, the verifier would be more suspectable to emulation/impersonating attacks [2,15].

4.2 Time-Bounded FPGA Authentication Using Reconfigurability

After the FPGA registration, the verifier is able to compute and predict the responses to any set of challenges by knowing (i) the cell-level features of the pertinent FPGA, (ii)

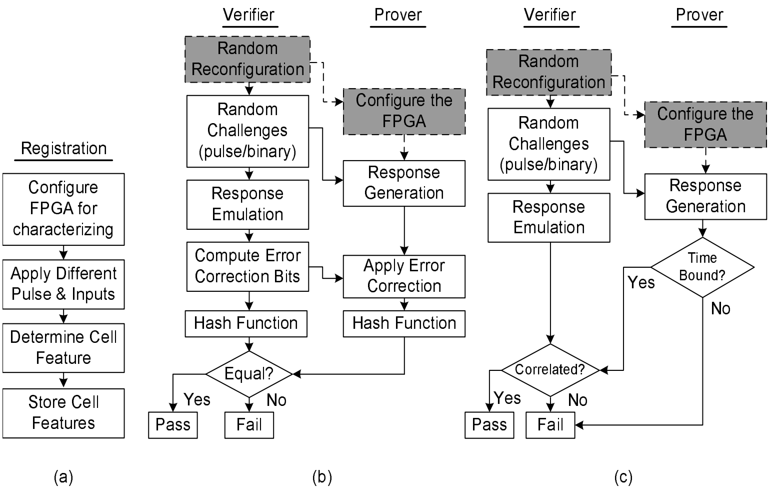


Fig. 4. (a) FPGA registration (b) Classic authentication flow (c) Time-bound authentication flow

the circuit structure and (iii) placement of the PUF circuit. The information on the PUF circuit structure and placement is embedded into the configuration bitstream. In the classic authentic method, the bitstream is never changed. A dishonest prover, off-line and given enough time and resources can (i) extract the cell-level delays of the FPGA and (ii) reverse engineer the bitstream to discover the PUF structure and its placement on the FPGA. During the authentication, he can compute the responses to the given challenges online by simulating the behavior of the PUF on the fly and produce the responses that pass the authentication.

A stronger set of security protocols can be built upon the fact that the prover is the only entity who can compute the correct response to a random challenge within a specific time bound since he has access to the actual hardware. In this protocol, prior to the beginning of the authentication session, the FPGA is blank. The verifier then sends a bitstream to the device in which a random subset of LUTs are configured for authentication. After the device is configured, the verifier starts querying the FPGA with random challenges. The verifier accepts the responses that are returned back only if $\Delta t \leq \Delta t_{\max}$ where Δt is the time lapsed on the prover device to compute the responses after receiving the configuration bitstream, and $\Delta t_{\max}$ is the upper bound delay estimated computation of responses by the authentic FPGA prover device, which is composed of device configuration, response generation, error correction, and hashing time all performed in hardware. The verifier would authenticate the device, only if the time the device takes to generate the response is less than $\Delta t_{\max}$. We denote the minimum emulation time by $t_{\min}^{\mathrm{emu}}$, where $t_{\min}^{\mathrm{emu}} >> \Delta t_{\max}$. Time-bounded authentication protocol can be added to the authentication flow, as demonstrated in Figure 4(c). Compared to the classic authentication flow, a time bound check is added after the hash function. While performing the above authentication, we emphasize on the assumption that the time gap between the hardware response generation and the simulation (or emulations) of the prover must be larger than the variation in channel latency. The time-bound assumption would be enough for providing the authentication proof [7,13,18].

4.3 Attacks and Countermeasures

Perhaps the most dangerous of all attacks is the impersonation attack. Impersonation attack aims at deceiving the verifier into authentication by cloning the same physical device, reverse-engineering and simulation of the authentic device behavior, or storing and replaying the communication, or random guessing. Among these threats only the reverse engineering and simulation attack may stand any chance of success. To break the time-bound protocol, an adversary needs to find the response to a new challenge, he has to reverse engineering the bitstream and simulate (or emulate) the PUF behavior within the given time constraint. Even after many years of research in rapid simulation technologies for hardware design and validation, fast and accurate simulation or emulation of a hardware architecture is extremely slow compared to real device. In addition, even though bitstream reverse-engineering have partially been performed on some FPGAs [19], performing it would require a lot of simulations and pattern matching. Thus, it would take many more cycles than the authentic hardware where the verifying time is dominated by bitstream configuration time (order of $100\mu s$). Simulating bitstream on software models would also take many more cycles than hardware and cannot be done within the limited time-bound. A great advantage of this authentication method is the large degree of freedom in selecting the LUTs that would be queried.

5 Robustness

The extracted delay signatures at characterization phase are subject to changes due to aging of silicon devices, variations in the operating temperature and supply voltage of the FPGA. Such variations can undermine the reliability of the authentication process. In this paper, we take a pragmatic approach to the problem which in conjunction with existing error correction methods [11] can significantly leverage performance and reliability of key generation and authentication. The proposed method performs calibration on clock pulse width according to the operating conditions.

Fortunately, many modern FPGAs are equipped with built-in temperature and core voltage sensors. Before authentication begins, the prover is required to send to the verifier the readings from the temperature and core voltage sensors. The prover then based on the current operating conditions calibrate the clock frequency. The presented calibration method linearly adjusts the pulse width using Equation 12.

$$T_{calib} = \alpha_{temp} \times (temp_{current} - temp_{ref}) + T_{ref} \tag{12}$$

$$T_{calib} = \alpha_{vdd} \times (vdd_{current} - vdd_{ref}) + T_{ref} \tag{13}$$

$temp_{ref}$ and vdd_{ref} are the temperature and FPGA core voltage measured at the characterization time. $temp_{current}$ and $vdd_{current}$ represent the current operating conditions. The responses from the PUF to the clock pulse width T_{calib} are then treated as if T_{ref} were sent to the PUF at reference operating condition. The calibration coefficients

α_{temp} and α_{vdd} are device specific. These coefficients can be determined by testing and characterizing each single FPGA at different temperatures and supply voltages. For example, if $D_i^{temp_1}$ and $D_i^{temp_2}$ are i-th extracted delay parameter under operating temperatures $temp_1$ and $temp_2$, then $\alpha_{t,i} = \frac{D_i^{t1} - D_i^{t2}}{t_1 - t_2}$.

6 Experimental Evaluations

In this section, the implementation details of the signature extraction system are presented. We demonstrate results obtained by measurements performed on Xilinx FPGAs and further use the platform to carry out authentication on available population of FPGAs. For delay signature extraction, the system shown in Figure 2 (b) is implemented on Xilinx Virtex 5 FPGAs. The systems contains a 32× 32 array of signature extraction circuits as shown in Figure 1. The CUT inside the characterization circuit consists of 4 inverters each being implemented using one 6-input LUT. The first LUT input (A_1) is used as the input of the inverter and the rest of the LUT inputs (A_2,...,A_6) serve as the binary challenges to alter the effective delay of the inverter. The characterization circuit is pushed into 2 slices (one CLB) on the FPGA. In fact, this is lower limit on number of slices that can be used to implement each characterization circuit. This is because interconnections inside the FPGA forces all the flip flops inside the same slice use either rising edge or falling edge clocks. Since the launch and sample flip flop must operate on different clock edges, they cannot be placed inside the same slices. In total, 8 LUTs and 4 flip flops are used (within two slices) to implement the characterization circuit. The error counter size (N) is set to 8, and the accumulated error values are stored if they are between 7 and 248.

We use an ordinary desktop function generator to sweep the clock frequency from 8MHz to 20MHz and afterwards shift the frequency 34 times up using the PLL inside the FPGA. The sweeping time is set to 1 milliseconds (due to the limitations of the function generator, the lower sweeping time could not be reached). The measured accumulated error values are stored on an external memory and the data are transferred to computer for further processing. Notice that the storage operation can easily be performed without the logic analyzer by using any off-chip memory.

The system is implemented on twelve Xilinx Virtex 5 XC5VLX110 chips and the measurements are taken under different input challenges and operating conditions. The characterization system in total uses 2048 slices for the characterization circuit array and 100 slices for the control circuit out of 17,280 slices.

The measured samples for each cell and the input challenge is processed and the twelve parameters as defined in Section 3.2 are extracted. Figure 5 shows the measured probability of timing error versus the clock pulse width for a single cell and a fixed challenge. The (red) circles represent original measured sample points and the (green) dots show the reconstructed samples. As explained earlier, to reduce the stored data size, error samples with values of 0 and 1 (after normalization) are not written to the memory and later are reconstructed from the rest of the sample points. The solid line shows the Gaussian fit on the data expressed in equation 11. Parameter extraction procedure is repeated for all cells and challenges. Figure 6 shows the extracted parameters d_1 and σ_1 for all PUF cells on chips $\#9$ while the binary challenge is fixed. The pixels in the

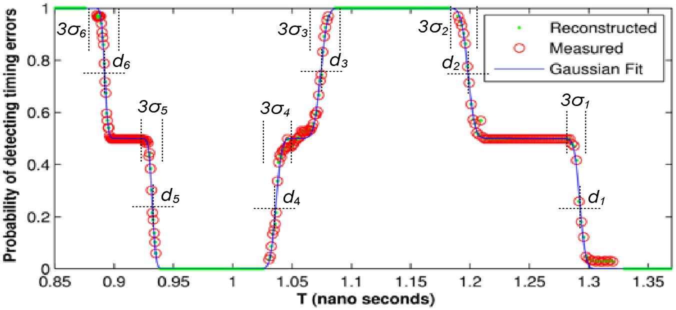


Fig. 5. The probability of detecting timing errors versus the input clock pulse width T. The solid line shows the Gaussian fit to the measurement data.

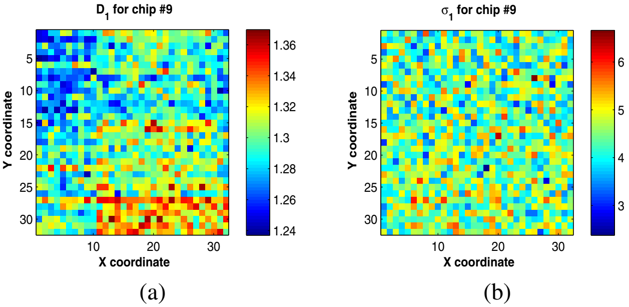


Fig. 6. The extracted delay parameters d_1 (a) and σ_1 (b) for chips 9

images correspond to the cells of the 32×32 array on FPGA, and their value represent the corresponding extracted parameters. Some levels of spatial correlation among d_1 parameters can be observed on the FPGA fabric. The boxplot in Figure 7 (a) shows the distribution of the delay parameters d_i for i=1,2,...,6 over all 12 chips and 1024 cells and 2 challenges. The central mark on the boxplot denotes the median, the edges of the boxes correspond to the 25th and 75th percentiles, the whiskers extent to the most extreme data points and the red plus signs show the outlier points.

Now using the measured data from the twelve chips, we investigate different authentication scenarios. The existing authentication parameters within defined framework substantially increases the degree of freedom under which authentication may take place. These parameters include the number of clock pulses (denoted by N_p), the number of tried challenges (denoted by N_c), the clock pulse width (denoted by T), and the number of PUF cells (N_{cell}) being queried. In other words, in each round of authentication, N_c challenges are tried during which N_p pulses of width T are sent to N_{cell} cells on the chip. The response for each challenge can be regarded as the percentage of ones in the N_p response bits.

In the first experiment, we study the effect of the number of cells and the width of clock pulse on the probability of detection (p_d) and false alarm (p_f). Detection error occur in cases where the test and target chip are the same, but due instability and noise

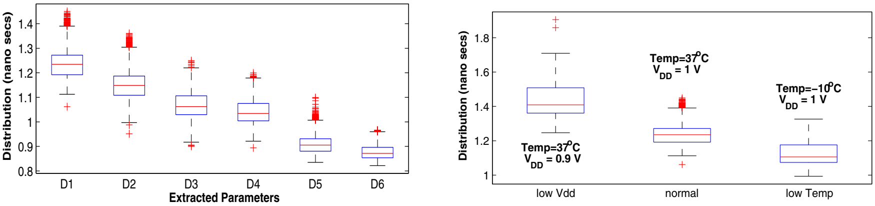
Table 1. Probability of false alarm (a) and probability of detection (b)

(a)

N_{cell}	Challenge Pulse Width					
	1.23	1.15	1.06	1.03	0.9	0.87
64	0.96	0	0	0	0	1.52
128	2.04	0	0	0	0	1.52
256	4.55	0	0	0	0	1.52

(b)

N_{cell}	Challenge Pulse Width					
	1.23	1.15	1.06	1.03	0.9	0.87
64	93.3	96.2	100	100	100	100
128	94.2	98.8	100	100	100	100
256	99.85	100	100	100	100	100



(a)

(b)

Fig. 7. (a) Distribution of delay parameters d_i. (b) The distribution of d_1 for normal, low operating temperature, and low core voltage.

in responses to fail to be authenticated as the same. On the other hand, false alarm corresponds to the cases where the test and target chip are the different, but they are identified at the same chips. During the experiment, the binary challenges to PUF cells are fixed and the number clock pulses is set to $N_p = 8$. Next, we study different cases where the clock width (T) is set to each of the medians of the values shown in Figure 7 (a). Setting the clock pulse width to the median values result in least predictability in responses. The same experiment is repeated for 10 times to obtain 10 response vectors for each chip. After that, the distance between the responses from the same chips (intra-chip) over repeated evaluations are measured using the normalized L_1 distance metric. The same procedure is performed on responses from different chips over difference evaluations in order to find the inter-chip responses.

If the distance between the test chip and the target chip responses is smaller than a pre-specified detection threshold, then the chip is successfully authenticated. In the experiments the detection threshold is set at 0.15.

Tables 1 shows the probability of detection and false alarm for different clock pulse widths and number of queried PUF cells. As it can be observed the information extracted from even the smallest set of cells is sufficient to reliably authenticate the FPGA chip if the pulse width is correctly set. In the next experiments, we study the effect of fluctuations in the operating conditions (temperature and core supply voltage) on the probabilities of detection and false alarm. Moreover, we demonstrate how linear calibration on the challenge clock pulse width can improve the reliability of detection.

To determine the calibration coefficient defined in Equation 12, we repeat the delay extraction process and find the delay parameters for all twelve chips at temperature -10^oC and core voltage 0.9 Volts. The chip operates at the temperature 37^oC and core

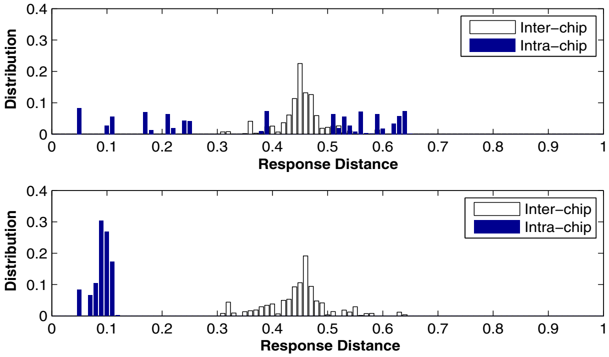


Fig. 8. The inter-chip and intra-chip response distances for $T = 0.95\,ns$ and $N_c = 2$ before (top) and after (bottom) calibration against changes in temperature

Table 2. The probability of detection and false alarm before and after performing calibration on the challenge pulse width in presence of variations in temperature and core voltage

		No Calibration								Calibrated							
		N_C=1				N_C=2				N_C=1				N_C=2			
		v_{low}		t_{low}		v_{low}		t_{low}		v_{low}		t_{low}		v_{low}		t_{low}	
		p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f	p_d	p_f
T	1.23	18.4	0	33.3	16.7	18.4	0	33.3	22.29	100	0	75	0	100	0	75	0
	1.06	18.4	0	18.4	0	18.4	0	18.4	0	50	0	50	0	57.3	0	50	0
	1.01	18.4	0	16.7	0	18.4	0	16.7	0	66.6	0	75	0	68.2	0	75	0
	0.95	18.4	0	16.7	0	18.4	0	16.7	0	66.7	0	100	0	84.9	0	100	0
	0.9	16.7	0	25	0	16.7	0	25	0	83.3	0	91.7	0	83.4	0	100	0
	0.87	25	0	25	0	25	1.5	25	0	100	0	100	0	100	0	100	0

voltage of 1 volts in the normal (reference) condition. We use the built-in sensors and the Xilinx Chip Scope Pro package to monitor the operating temperature and core voltage. To cool down the FPGAs, liquid compressed air is consistently sprayed over the FPGA surface. Figure 7 (b) depicts the changes in the distribution of the first delay parameter d_1 at the three different operating conditions.

The probabilities of detection and false alarm are derived before and after performing calibration on the challenge pulse width for different clock pulse widths and number of binary challenges to the cells. In this experiment, all 1024 PUF cells on the FPGA are queried for the response. The number of pulses sent for each binary challenge is set N_p =8 as before. As it can be seen in Table 2, the detection probabilities are significantly improved after performing linear calibration based on the coefficients extracted for each chip. The variables v_{low} and t_{low} correspond to -10^oC temperature and 0.9 supply voltages respectively. The reported probabilities of Table 2 are all in percentage. Also note that for the challenge pulse width of $T = 0.87\,ns$, the probability of detection reaches 100% and probability of alarm falls to zero after calibration. The same holds true for

$N_c = 2$ and $T = 0.87, 0.9, 0.95$. Thus, increased level of reliability can be achieved during authentication with proper choice of pulse width and number of challenges.

Figure 8 shows how performing calibration decreases the intra-chip response distances in presence of temperature changes. The histogram correspond to $T = 0.95\ ns$ and $N_c = 2$ in Table 2 before and after calibration.

7 Conclusions

We presented a technique for FPGA authentication that takes advantage of the unclonable timings variability present in FPGAs, the FPGA reconfigurability, and its unprecedented speed. Authentication comprises of two phases; namely registration and authentication. During registration, cell level timing features are extracted and stored in a database. Later at the authentication phase, the verifier generates a random configuration bitstream and sends to the prover. A unique aspect of the new method is its high degree of freedom in placing the PUF cells and selection of challenges. The protocol relies on the fact that online reverse-engineering of the bitstream is a non-trivial task. A new calibration method for improving robustness to temperature and voltage fluctuations was demonstrated. Evaluations on Xilinx V5 FPGA show the effectiveness and practicality of the new timing signature extraction and authentication method.

References

1. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297, 2026–2030 (2002)
2. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: CCS, pp. 148–160 (2002)
3. Suh, G., O'Donnell, C., Devadas, S.: AEGIS: A single-chip secure processor. IEEE Design & Test of Computers 24(6), 570–580 (2007)
4. Atallah, M., Bryant, E., Korb, J., Rice, J.: Binding software to specific native hardware in a VM environment: the PUF challenge and opportunity. In: VMSec, pp. 45–48 (2008)
5. Guajardo, J., Kumar, S., Schrijen, G., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
6. Suh, G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC, pp. 9–14 (2007)
7. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Techniques for design and implementation of secure reconfigurable pufs. ACM TRETS 2(1), 1–33 (2009)
8. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: ACSAC, pp. 149–160 (2002)
9. Lee, J., Daihyun, L., Gassend, B., Suh, G., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symp. of VLSI, pp. 176–179 (2004)
10. Holcomb, D., Burleson, W., Fu, K.: Power-up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Trans. Computers (2009)
11. Gassend, B.: Physical random functions. S.m. thesis, MIT (2003)
12. Ruhrmair, U., Solter, J., Sehnke, F.: On the foundations of physical unclonable functions. In: Cryptology ePrint Archive (2009)

13. Ruhrmair, U.: SIMPL system: on a public key variant of physical unclonable function. In: Cryptology ePrint Archive (2009)
14. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure PUFs. In: ICCAD, pp. 670–673 (2008)
15. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: International Test Conference (ITC), pp. 1–10 (2008)
16. Wong, J.S.J., Sedcole, P., Cheung, P.Y.K.: Self-measurement of combinatorial circuit delays in fpgas. ACM TRETS 2(2), 1–22 (2009)
17. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
18. Beckmann, N., Potkonjak, M.: Hardware-based public-key cryptography with public physically unclonable functions. In: Katzenbeisser, S., Sadeghi, A.-R. (eds.) IH 2009. LNCS, vol. 5806, pp. 206–220. Springer, Heidelberg (2009)
19. Note, J., Rannaud, E.: From the bitstream to the netlist. In: FPGA, pp. 264–274 (2008)