Cloning the Unclonable: Physically Cloning an FPGA Ring-Oscillator PUF

Hayden Cook, Jonathan Thompson, Zephram Tripp, Brad Hutchings, and Jeffrey Goeders
Department of Electrical and Computer Engineering
Brigham Young University, Provo, Utah, USA

***Abstract*—This work presents a novel technique to physically clone a ring oscillator physically unclonable function (RO PUF) onto another distinct FPGA die, using precise, targeted aging. The resulting cloned RO PUF provides a response that is identical to its copied FPGA counterpart, i.e., the FPGA and its clone are indistinguishable from each other. Targeted aging is achieved by: 1) heating the FPGA using bitstream-located short circuits, and 2) enabling/disabling ROs in the same FPGA bitstream. During self heating caused by short-circuits contained in the FPGA bitstream, circuit areas containing oscillating ROs (enabled) degrade more slowly than circuit areas containing non-oscillating ROs (disabled), due to BTI affects. This targeted aging technique is used to swap the relative frequencies of two ROs that will, in turn, flip the corresponding bit in the PUF response.**

Two experiments are described. The first experiment uses targeted aging to create an FPGA that exhibits the same PUF response as another FPGA, i.e., a clone of an FPGA PUF onto another FPGA device. The second experiment demonstrates that this aging technique can create an RO PUF with any desired response.

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are increasingly being used to implement digital circuits in security-sensitive industries, such as communications infrastructure, transportation technologies, medical devices, and particularly national defense and aerospace applications. In these types of applications, it is critical that the supply chain is secure, that devices are from a trusted source, and that they remain secure after deployment.

Physical unclonable functions (PUFs) play an important role in FPGA security by providing a unique output derived from the inherent randomness of the manufacturing process. Since a PUF's output is random and unique for every chip, they are commonly used for device authentication, IP protection, and cryptography. As the name suggests, PUFs rely on being physically unclonable, meaning that even if the output of the PUF is known, it cannot be physically replicated on another chip (assuming no change to the challenge circuit). This prevents attacks such as having a counterfeit chip replicate an authentication response of a trusted chip to gain access to a secure system.

This work presents a technique to physically clone a ring oscillator PUF (RO PUF), which is commonly used in FPGAs. Our cloning technique leverages a novel targeted aging technique that allows us to change the relative frequency of individual paths of the FPGA. One can use targeted aging as a rewriting process, changing the PUF response of one FPGA to match the response of another or to any arbitrary value.

To the best of our knowledge, this is the first work to demonstrate the cloning of an RO PUF. We believe this work proves a considerable vulnerability of not only RO PUFs, but other delay-based PUFs implemented on FPGAs.

The main contributions of this work are:

- A novel technique for precise FPGA aging, capable of affecting delays at individual FPGA tiles.
- Demonstrating this aging technique by cloning an RO PUF of one FPGA onto another.
- Demonstrating that for RO PUFs, any arbitrary PUF output can be achieved.
- A potential countermeasure to detect if a chip has undergone targeted aging.

This paper is organized as follows: Section II discusses related work in PUF response emulation and cloning. Section III provides background information on PUFs, BTI, and short circuit induced aging. Section IV presents our targeted aging technique, and Section V discusses the PUF model as well as two potential threat models. We then present the results of two experiments: Section VI presents an experiment that clones the response of an RO PUF from one FPGA onto another, while Section VII presents an experiment that shows that any arbitrary RO PUF response can be achieved with our cloning technique. Section VIII presents a potential countermeasure that allows users to detect if targeted aging has been performed on an FPGA. Finally, Section IX concludes this paper and discusses future work.

II. Related Works

In the literature, the act of cloning a PUF usually refers to simply emulating a PUF response. The most common technique for emulating a PUF response was proposed by Rührmair et al. They use machine learning techniques to discover a PUF's input-output responses and create a create a software model that is identical to the PUF's response [1]. However, this technique requires a threat model that enables injecting the malicious software model into the existing software that handles the inputs and outputs of a PUF, which could be countered with common software authentication techniques.

Another emulation technique is briefly mentioned by Gao et al. [2], who propose that physical emulation can be achieved

H. Cook, J. Thompson, Z. Tripp, B. Hutchings, and J. Goeders are also with the NSF Center for Space, High-performance, and Resilient Computing (SHREC)

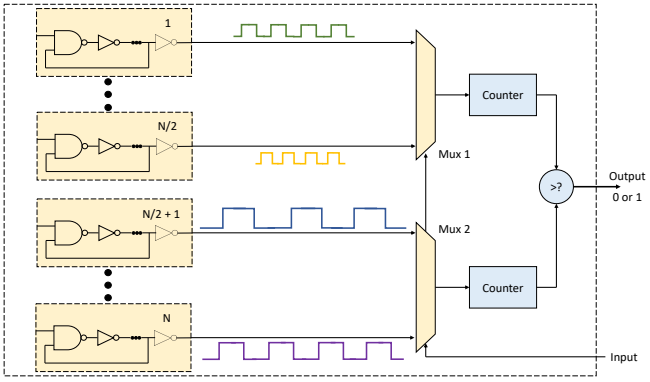


Figure 1: A traditional implementation of an RO PUF.

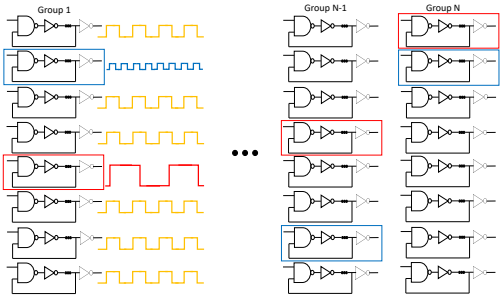


Figure 2: An RO PUF using a 1-out-of-8 masking scheme as a reliability enhancement. This RO PUF contains N groups of ROs, and since each group produces a single bit, has an output of length N.

by creating a biased RO PUF where each RO has a different number of inverters. The larger ROs will have bigger frequencies than the smaller ROs, forcing the comparisons to output a pre-determined value. For FPGAs this would require modification of the bitstream, which once again could be countered by using common authentication techniques, such as hashing and validating the bitstream, or through bitstream encryption, which is supported by most vendors.

These techniques to emulate a PUF response are fundamentally different from creating a physical clone, which requires changing physical characteristics of the device at the transistor level. To the best of our knowledge, only one other work has successfully created a physical clone of a PUF. Helfmeier et al. [3] use a focused ion beam (FIB) to clone a SRAM PUF, which relies on the random values the memory takes on startup. This work is unique in that they use the FIB to etch out part of the transistors that make up the SRAM. This physical alteration of the transistors creates a bias in the SRAM and forces it to the desired value on startup. However, SRAM PUFs are often impossible to implement on FPGAs as the SRAM in most FPGAs are initialized to a predetermined state upon startup. Additionally, this technique is invasive, uses expensive equipment, and requires intimate knowledge of the transistor layout of the PUF (which is more difficult as an FPGA is a much more complex device than an SRAM).

III. Background

A. RO PUFs

PUFs produce a unique output (response) based on an input (challenge) and a source of intrinsic randomness. PUFs are often used for low-cost chip authentication or as a source of true randomness for cryptographic key generation [4]. Silicon PUFs use the intrinsic randomness of the manufacturing process to implement a random function that has a unique output for every chip. While many different types of silicon PUFs exist, such as the SRAM PUF [5], the Butterfly PUF [6] and the Arbiter PUF [7], the RO PUF is often preferred for FPGAs due to their simplicity and overall performance [8]. Additionally, unlike other delay-based PUFs, such as butterfly PUFs and arbiter PUFs, RO PUFs do not require perfectly symmetric routing which is difficult to achieve on an FPGA [9].

Suh and Devadas were the first to propose a ring oscillator based PUF (RO PUF). A Ring Oscillator (RO) is a combinational loop made of an odd number of inverters that oscillates at a frequency determined by the intrinsic speed of the silicon where the RO is placed. Due to manufacturing variations, ROs placed in different locations or on different chips will oscillate at distinctly different frequencies. RO PUFs then compare these random frequencies to output a random string of zeros and ones.

Figure 1 illustrates an RO PUF, which consists of N ROs, each of which are hooked up to one of two MUXes. The output of each of these MUXes are fed into one of two counters, which are in turn, fed into a comparator which outputs either a zero or one for each comparison. To avoid correlation between bits, it is common to use each RO in only one comparison, meaning for N ROs, there would be $N/2$ comparisons resulting in a random string that is $N/2$ bits long.

Maiti et al. proposed a configurable RO (CRO). A CRO uses additional MUXes to change which inverters an RO uses, which will change the frequency of the ROs in the PUF, thus changing the resulting output string. This allows each RO pair in the PUF to produce multiple bits [10], without increasing the size of the RO PUF. While CROs are generally more compact than a traditional RO, they are generally more complicated to implement and suffer from a decrease in uniqueness in PUF outputs among different chips [8].

B. Reliable RO PUFs

Unfortunately, like all delay-based PUFs, RO PUFs are susceptible to changes in temperature or voltage and aging [11]. These phenomena can cause different ROs to change frequencies at different rates, causing some RO pairs to swap relative frequencies which causes the RO PUF response to change. Since the hamming distance only needs to be under some pre-determined threshold for a chip to be authenticated, this is tolerated for PUF-based authentication. However, cryptographic applications require the PUF string to be precisely the same throughout the lifetime of the chip. Using error correcting algorithms or a fuzzy extractor have been proposed as solutions for a RO PUF output's volatility [5]. However,

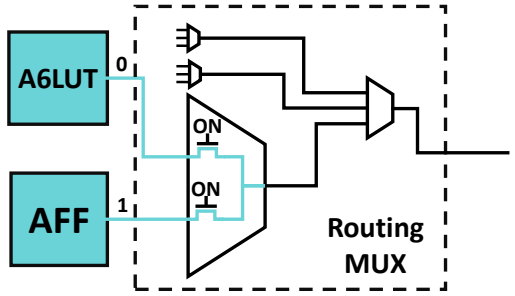


Figure 3: A diagram of a short circuit implemented on an FPGA, from [21].

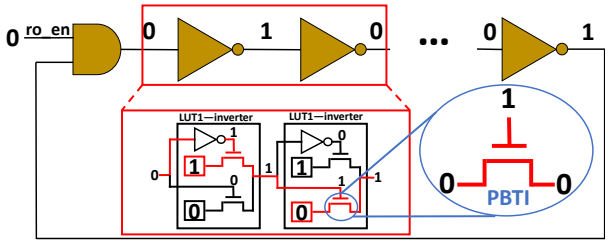


Figure 4: The layout of the ROs that are placed inside the shorted bitstream, including a diagram showing every other LUT being held at a positive bias.

these methods require additional post-processing on the PUFs output, which is undesirable for many applications.

Suh and Devadas [12] proposed a reliability enhancement to their RO PUFs as an alternative to post-processing techniques. Their reliable RO PUF uses a l-out-of-k masking scheme. As illustrated in Figure 2, ROs are grouped into groups of k (where $k = 8$), with each group producing a single bit. Each RO must be characterized to find the ROs that have the minimum and maximum frequencies in each group. These ROs are saved and compared throughout the lifetime of the chip. This greatly increases the difference in frequency between the ROs in a pair, making them less likely to swap relative frequencies during environmental changes or aging. However, this also increases the size of the RO PUF circuit by a factor of k.

C. Bias Temperature Instability (BTI)

Our targeted aging technique used by our cloning method relies on bias temperature instability (BTI). BTI occurs when an electric field is applied across the gate oxide of a MOSFET transistor which can create interface traps that trap carriers, raising the threshold voltage of the transistor and lowering its switching speed [13], [14].

There are two distinct cases of BTI: negative bias temperature instability (NBTI), which mainly affects PMOS transistors, and positive bias temperature instability (PBTI), which mainly affects NMOS transistors. While only NBTI was a concern in older technology nodes, the introduction of high-k dielectrics in sub-45 nm nodes has made PBTI become as much of a concern as NBTI [15], [16].

Historically, the LUTs and routing elements of an FPGA have been made from NMOS pass transistors [17], [18], which are primarily affected by PBTI. The conditions needed for an NMOS transistor to experience PBTI include increased temperature and positive gate bias [19].

Since a major factor of BTI degradation is a constant DC gate bias, an AC gate bias helps protect a circuit from BTI degradation [20]. This can be accomplished by the continuous switching of transistors via a high-frequency signal.

D. Short Circuit Induced Aging on an FPGA

In order to accelerate BTI effects on the FPGA, we configure short circuits on the FPGA. This method was first proposed by Gaskin et al. who used custom bitstreams to configure short circuits onto the FPGA fabric, resulting in high heat and current [21], [22]. By configuring the FPGA to contain over 20,000 short circuits, they show that the temperature rises to at least 177 °C, accelerating BTI and decreasing the frequency of the shorted fabric by 5.1% after 36 days of running short circuits.

The technique we use to induce FPGA aging closely follows their technique. It involves creating short circuits by routing two nets with opposing values (a logic-1 and a logic-0) to drive the same net. In particular, the short circuits use lookup table (LUT) and flip-flop (FF) primitives, as seen in Figure 3. To complete the short, the LUT and the FF must be selected so that they connect to the same routing multiplexer, and the two multiplexer inputs must be activated simultaneously. While it is not normally possible to create designs with short circuits, it is made possible by using RapidWright [23] to create the routed design checkpoint, importing the checkpoint into Vivado, and then disabling DRC checks before generating a bitstream.

This short-circuit technique has many advantages over traditional techniques used to accelerate chip aging via heat, such as using an oven [14], [24]–[26] or a heatgun [27]. One advantage is its ability to be performed remotely. This provides extra convenience for our cloning experiments and expands our attack model for cloning an RO PUF. Additionally, short-circuits allow the chip to get hotter than traditional methods, which in turn speeds up the aging process.

Cook et al. [22] show that short circuits can be placed at different locations on the FPGA to create a non-uniform aging effect. However, in their work the aging is primarily an effect of the self-heating of the chip, so the gradient of aging across the chip is limited due to heat spreading. Their technique does not produce sharp aging gradients that are required to change the frequency of an RO relative to its neighboring tile, and so this aging technique alone is not sufficient to clone PUFs.

IV. Targeted Aging Technique

Cloning PUFs requires precise enough aging that the relative frequencies of two nearby ROs can be flipped. Once flipped, the resulting bit obtained from the comparison will also flip, allowing an attacker to change the PUF's response string to any arbitrary value.

In order to achieve this precise aging effect, we create a design (separate from the PUF challenge-response bitstream) that contains strategically placed ROs surrounded by thousands of short circuits. These ROs are configured simultaneously with the short circuits and change the rate of degradation caused by the short circuits (see *Spoofing Bitstream* in Figure 5).

If the ROs (shown in Figure 4) are enabled by holding the *ro_en* signal high, then the transistors of the ROs switch at high speeds (about $500\,\mathrm{MHz}$). As explained in subsection III-C, these high switching speeds largely mitigate the BTI effects of the surrounding short circuits, provide a protective effect against aging, and cause these RO regions to degrade slower than the surrounding shorted regions.

When these ROs are *disabled*, the chain of inverters present in the ROs speed up the rate of degradation. We believe this phenomenon occurs because disabling the RO holds some of the transistors in a positive bias, causing PBTI aging effects. This results in the RO regions aging at a faster pace than the rest of the fabric. Figure 4 contains a transistor-level diagram of what we believe occurs when we implement an RO in the FPGA fabric, and shows that when disabled (holding the *ro_en* signal low), the transistor gates will be held at a positive bias for every other inverter in the RO. While a disabled RO (chain of inverters) is not the only circuit structure that would hold the transistors at a bias and induce BTI, we choose to reuse the same circuit structure for ease of design.

In order for targeted aging to occur, the ROs must be surrounded by short circuits. Without being exposed to extreme heat from the short circuits, the ROs are ineffective at affecting the speed of the fabric enough to clone an RO PUF. This is expected as BTI aging effects are most pronounced when there exists a combination of high heat and biased transistors.

V. Attack Model

Our targeted aging technique is key in allowing us to clone an RO PUF. Unlike most PUF "cloning" methods, which emulates the PUF challenge-response pairs in software, our cloning method uses our targeted aging technique to physically alter the frequencies of different locations of a chip so that it gives the exact same RO PUF response as another chip.

A. RO PUF Model

To show the versatility of our cloning technique, we chose to target Suh and Devadas's "reliable RO PUF" for our experiments. As explained in Section III-B, reliable PUFs work by grouping ROs into groups of k. In our experiments, we chose $k = 8$. For each group, the ROs with the minimum and maximum frequencies are identified. These ROs are then saved to a configuration stored in non-volatile memory off-chip, such as a separate flash chip. This allows the same ROs to be compared throughout the lifetime of the part with very low probability of the PUF ever naturally changing.

We are specifically targeting a "reliable" PUF because it is designed such that the frequencies of the compared ROs are far apart. While targeting a PUF where ROs have similar frequencies would have been easier, attacking a more resilient PUF model demonstrates the capability of our precise aging technique.

While Suh and Devadas explain the general idea of their reliability enhancement, they leave out the implementation details on how to compare the ROs with the minimum and maximum frequencies. In our implementation we locate the fastest and slowest PUFs indices in each group, and save this configuration data as a list of N pairs: [(A_1, B_1), (A_2, B_2), ..., (A_N, B_N)], $A_i \in [0, 7]$, $B_i \in [0.7]$. Each pair is ordered so that $A_i < B_i$. We then use the comparison $f(A_i) < f(B_j)$ (where $f()$ is the frequency of the RO at the given index) to determine if the PUF bit from that pair is a zero or a one. This is illustrated in Figure 5.

B. Authentication Threat Model

One potential cloning attack is to clone an RO PUF used to authenticate a trusted FPGA. If an attacker is able to physically clone a RO PUF response of a trusted FPGA onto a malicious FPGA (which could contain back doors, hardware trojans, etc.), then they could replace the trusted FPGA with the malicious one without it being detected.

Our threat model for such an attack is illustrated in Figure 5. In our threat model, the user of the trusted FPGA has designed a $k = 8$ reliable PUF architecture and has measured the intrinsic RO speeds to identify the indices of the fastest and slowest ROs of each group. These configuration indices, along with the bitstream, are saved and stored in such a way that they cannot be modified (read-only, non-volatile memory, or validated with a hash).

The attacker in our threat model has knowledge of the trusted design, including the bitstream and configuration indices but cannot make modifications to these files. This could occur if the attacker gained access to the memory where the files are stored or possibly by characterizing the PUF itself (which can be accomplished with a variety of methods [28], [29]). The attacker must know the exact details of the RO PUF (placement and routing of individual wires), as our targeted aging technique targets the individual transistors in the LUTs and routing multiplexers, so access to the bitstream or hardware design files is essential.

With these assumptions met, the attacker can then take a malicious FPGA and use our targeted aging technique to flip the relative frequencies of selected RO pairs. This will flip the corresponding bits and allow the attacker to physically clone the RO PUF response of the trusted FPGA onto their malicious FPGA. With this accomplished, the attacker could swap out trusted components with their counterfeit parts without detection.

For example, in Figure 5, you will notice that the trusted FPGA has been characterized, and for RO group *#3*, the fastest RO is located at $i = 0$, and the slowest RO at $i = 7$. In the attacker's FPGA, the RO at $i = 7$ is actually faster than the RO at $i = 0$, so the spoofing process is required to flip these relative frequencies. An enabled RO is placed at $i = 0$, and a disabled RO at $i = 7$, and they are surrounded by heat-generating short circuits. After this biased aging process, the

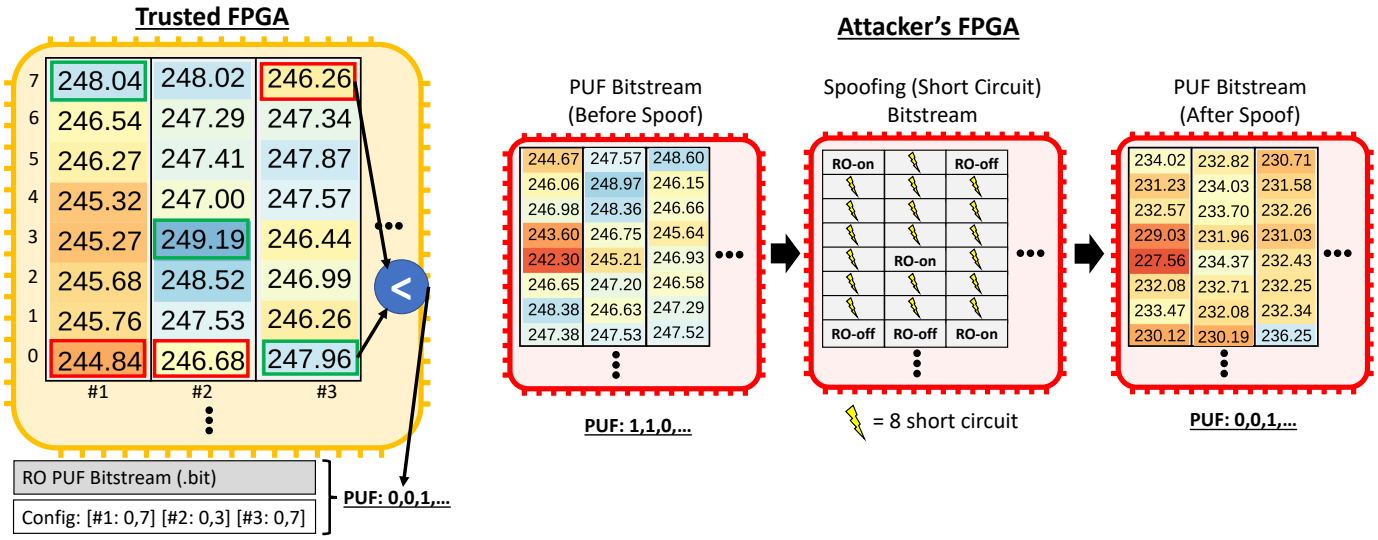


Figure 5: Diagram of the threat model and PUF architecture targeted by our experiment, with heatmaps showing the frequency of the ROs in MHz. The trusted FPGA contains many ring oscillators, grouped into regions, readable using the RO PUF bitstream. Following reliable PUF convention, configuration data indicates which ROs within each region should be compared to generate each bit of the PUF. Our attack model assumes that the trusted bitstream and configuration data is known, but cannot be modified. The attacker's FPGA then undergoes our cloning process such that it now produces the same PUF response, without modifying the trusted bitstream or configuration data.

relative frequencies will have been flipped. This process will be described in detail in our experiment in Section VI.

C. Cryptographic Threat Model

In the previous threat model, one FPGA's PUF response is modified to match another. However, there may be cases where the attacker actually wants to modify the PUF response of a trusted, *already characterized* FPGA (i.e., not clone the response onto another FPGA, but actually modify the response of the original FPGA). For example, this might be applicable for systems that use PUF responses for key generation for cryptographic functions [30], where the attacker wants to modify the trusted FPGA to use a key of their choice.

At first glance, this seems like the same problem as previously discussed: changing the PUF response of an FPGA. However, there are some subtle differences that make this attack a bit harder.

Consider again the example in Figure 5. The cloning attack previously discussed changed the output response of RO group #3 such that the RO at $i = 7$ is made to be slower than the RO at $i = 0$; however, you will notice that these two ROs had similar frequencies to begin with (247.52MHz and 248.60MHz), so the relative frequencies needed to be shifted by only 1.08MHz in order to flip the PUF bit of the attacker's FPGA.

In contrast, if an attacker wants to flip PUF bits by attacking the original, trusted FPGA, they will naturally have to flip ROs that are further apart in frequency. This is because the attacker is now trying to change the relative RO frequencies of ROs that are, by definition, the slowest and fastest ROs in their group. For example, if the attacker wanted to change the PUF response of group #3 in the trusted FPGA from Figure 5, then they would need to flip the relative frequencies of PUFs that run at 246.26MHz and 247.96MHz, a difference of 1.70MHz. This may not seem substantially more difficult than the cloning attack's difference of 1.08MHz, but it does make the attack take longer to perform.

It is also worth noting that for this attack, the current PUF response does not need to be known. However, the RO PUF layout and the locations of minimum- and maximum-frequency ROs (i.e. the RO PUF configuration) must still be known.

This type of attack is demonstrated in our second experiment, discussed in Section VII.

VI. RO PUF Cloning Experiment

A. Experiment Overview

To test that our targeted aging method is able to physically clone an RO PUF, we performed an RO PUF cloning experiment. This cloning experiment is based on the authentication threat model described in Section V-B and targets a 128-bit RO PUF that uses a 1-out-of-8 masking scheme to improve reliability. Each RO in the PUF takes up an entire tile, using seven LUTs as inverters and one LUT as an AND gate. We used RapidWright to ensure all ROs use the exact same placement and routing to avoid bias in our comparisons. Each RO group is one RO (tile) wide and eight ROs (tiles) tall.

We perform this experiment with two Digilent ARTY A7-35T boards containing Xilinx Artix-7 FPGAs. One board contains the "trusted" FPGA while the other the "clone" FPGA. Both boards are modified to bypass the power regulator

Table I: Summary of the RO PUF Cloning Experiment.

Day	Target String	Clone String	Hamming Distance
0	0xf7fdd44402ff615 27b25f91f7e63fe08	0xa3bb27088027f36 2406f228f6ff2bbee	55
1	0xf7fdd44402ff615 27b25f91f7e63fe08	0xf7fdd44002ff615 27b25f91f7e63fe48	2
2	0xf7fdd44402ff615 27b25f91f7e63fe08	0xf7fdd44402ff615 27b25f91f7e63fe08	0

Table II: Summary of the RO PUF Inversion Experiment.

Day	Target String	Clone String	Hamming Distance
0	0x08022bbbfd009ea d84da06e0819c01f7	0xf7fdd44402ff615 27b25f91f7e63fe08	128
1	0x08022bbbfd009ea d84da06e0819c01f7	0x094e2bb7fd009e2 d85da46e2818d01f5	13
2	0x08022bbbfd009ea d84da06e0819c01f7	0x08022bbbfd009ea d84da06e0819c01f7	0

so that VCCINT can be powered directly. The VCCINT was connected to separate channels on a Keysight N6705C power supply, which allows for high currents and precise voltages. Both boards were placed in a thermal chamber set to $35\,^{\circ}\mathrm{C}$ to eliminate any ambient temperature fluctuations.

Before starting the cloning process, the trusted FPGA was characterized by measuring the frequency of the ROs that make up the PUF. The ROs with the minimum and maximum frequencies in each group were chosen and saved to a configuration. This configuration was used to create a targeted aging bitstream. In this bitstream, disabled ROs were placed in the same spot as the minimum-frequency PUF ROs on the trusted FPGA, while enabled ROs were placed in the same spot as the maximum-frequency ROs. 20,544 short circuits were then placed on the FPGA, surrounding the ROs. To ensure no undesired bits flipped during the aging process, targeted aging ROs were placed for each PUF RO pair regardless of whether the corresponding bit needed to be flipped during the cloning process.

We programmed the targeted aging bitstream onto the clone FPGA for 24 hours at a time. After each 24 hour period, we let the device cool and then characterized the frequency of the PUF ROs to determine if the PUF was successfully cloned.

In preliminary experiments, we found that increasing the voltage of VCCINT greatly increases the effects of BTI and further accelerates our targeted aging. In both experiments described in this paper the voltage of VCCINT was increased from the nominal $0.95\,\mathrm{V}$ to $1.05\,\mathrm{V}$. This allows us to reduce the time it takes to clone an RO PUF from months to days. The voltage was returned to it nominal value while characterizing the RO frequencies. The voltage of $1.05\,\mathrm{V}$ is still within the rated operating voltage of the part, which has a maximum voltage of $1.1\,\mathrm{V}$

B. Experiment Results

The results of the RO PUF cloning experiment can be seen in Table I. It took two days to completely clone the PUF response of the trusted FPGA onto the clone FPGA, with all but two bits flipping after a single day.

The heatmaps in Figure 5 show the frequencies for a subset of ROs that make up the PUF for both the trusted FPGA and the attacker's clone FPGA. As explained in Section V-B, the clone FPGA will ultimately use the same RO PUF configuration as the trusted FPGA. This configuration corresponds with the minimum- and maximum-frequency ROs (outlined in red and green respectively) for each RO group of the *trusted* FPGA.

Since the clone FPGA will use the same configuration as the trusted FPGA, our experiment must target the same ROs. Before the cloning process, Figure 5 shows that the PUF substring of the clone FPGA is inverted from the trusted FPGA. By placing the enabled and disabled ROs in the short circuit bitstream, we flip the relative frequencies of the targeted ROs on the clone FPGA, giving it the same PUF output as the trusted FPGA.

While Figure 5 shows that the PUF re-writing process decreases the clone FPGA's *absolute* frequencies across the heatmap, the enabled ROs see an increase in *relative* frequency compared to surrounding RO frequencies. Conversely, the disabled ROs see a decrease in *relative* frequency compared to their neighbors.

VII. PUF Inversion Experiment

In the previous experiment we re-wrote a PUF to match the PUF of another FPGA. As shown in Table I, the 128-bit PUFs of the two FPGAs differed by only 55 bits to begin with, so not all ring oscillator pairs needed to have their relatively frequency flipped. In addition, many of the targeted ROs had similar frequencies to begin with. To further demonstrate the effectiveness of our technique, we next conduct an experiment where we *invert every bit* of a trusted FPGA's PUF.

A. Inversion Experiment Overview

To show that an RO PUF response can be changed to output any arbitrary response, we created an experiment that takes the trusted FPGA from Section VI and inverts its RO PUF output. This experiment addresses the threat model described in Section V-C, and demonstrates that our aging technique can create arbitrary PUFs, even when attacking ROs that are far apart in frequency.

The experiment follows the same procedure as outlined in Section VI-A. We created a targeted aging bitstream similar to the one used in the previous experiment, but swapping the position of the enabled and disabled ROs.

B. Inversion Experiment Results

The results for the inversion experiment are summarized in Table II, and again show that it only took two days to completely invert the RO PUF response of the RO PUF. It should be noted that while this inversion experiment took the same amount of time as the cloning experiment (two days),

it is dependent on the off-the-shelf characteristics of the part, and we have observed it in general taking a bit longer than the cloning process (sometimes several days).

The heatmap in Figure 6 shows the percent change of the frequency for each RO on the entire FPGA, as a result of the targeted aging process. Here, it can be clearly seen which tiles contain enabled and disabled ROs in the targeted aging bitstream. The tiles with a much larger localized slowdown (dark red tiles) are where the disabled ROs were placed and tiles with a much smaller localized slowdown (dark blue tiles) are where the enabled ROs were placed. As a whole the FPGA exhibits non-uniform aging (bottom left ages more than top right), but this does not impact the effectiveness of our attack. We regularly see this non-uniformity and believe it is a product of the heat spreading capabilities of the IC and/or packaging.

Additionally, Figure 6 demonstrates how much overall aging an FPGA must go through in order to clone a PUF. Shorted regions show a slowdown of 5.34% to 6.73%, while the enabled and disabled ROs show slowdowns of 3.18% to 4.84% and 5.00% to 7.92% respectively.

Finally, Figure 7 shows the frequency difference for each of the 128 RO pairs over time. The bit corresponding to each pair flips when the frequency difference dips below zero, which occurs within two days for all RO pairs. After each pair was flipped, we continued the targeted aging for an additional 17 days. The figure shows that the frequency difference over time for each RO pair decreases as a decaying exponential. The largest drop occurs on the first day of targeted aging, with subsequent days seeing a increasingly smaller drop in frequency difference. This is similar to the artificial aging profile demonstrated in [22].

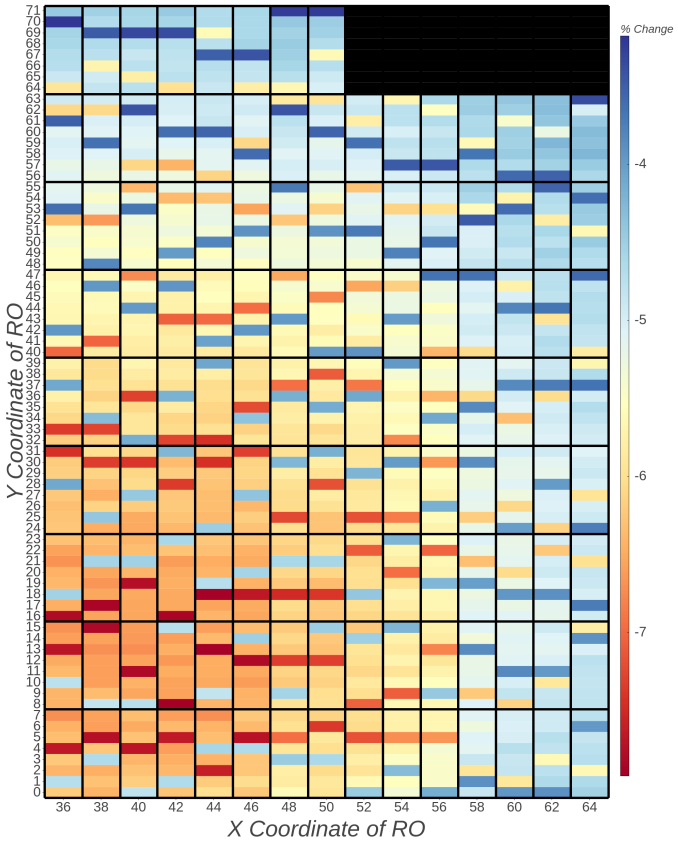


Figure 6: Heatmap showing the percent change during the Inversion Experiment. Each RO group is outlined in black.

C. Recovery

BTI often sees a small amount of recovery after a stress period [31]–[33]. Thus to investigate the effects of recovery on our cloned PUF response, we perform a five day recovery period.

To recover the chip, we cycle through programming the FPGA with five different benchmarks to represent real workloads running on the chip. Three of the designs we use (*bgm*, *blob merge*, and *diffeq2*) are from the VTR 7 benchmark suite [34]. The fourth design is a bitcoin design modified from github.com/progranism/Open-Source-FPGA-Bitcoin-Miner. We run each design on the FPGA for six hours, for a total of 24 hours. We then program our characterization bitstreams to capture the frequencies of the ROs. This process is repeated for the duration of our recovery period.

The recovery period shown in Figure 7 indicates that no significant recovery occurs within the recovery period. This suggests that our cloning technique may be permanent. However, additional recovery experiments should be performed before drawing such a conclusion.

VIII. CounterMeasures

Our targeted aging technique allows the cloning of an RO PUF with nothing more than vendor supported tools and a standard power supply, allowing such an attack to be performed by anybody with sufficient access to the FPGA and design files. While we have demonstrated our cloning method on a reliable RO PUF with a 1-out-of-8 masking scheme, we believe that this same attack can be used on many other delay based PUF, including more traditional RO PUFs. In fact, for standard PUFs the frequency differences would likely be much less, making the attack even easier. Thus it is important to discuss possible countermeasures for our cloning attack.

One possible countermeasure is to detect if a part has been aged. Cloning an RO PUF requires substantial aging of the FPGA fabric, leading to an increase in delay throughout the FPGA (Figure 6). There are many techniques to detect recycled FPGAs by identifying if aging through normal use has occurred [35], [36]. These same techniques can be used to detect if a PUF has been cloned from a relatively new FPGA. However, these techniques may report false positives if the "trusted" FPGA itself has experienced aging from normal usage.

Thus, instead of attempting to identify normal aging, a cloned FPGA could be detected by identifying *targeted* aging. This can be accomplished by calculating the *relative* frequency of ROs that are likely to be targeted during a cloning attack. We define the relative frequency of a region using Equation (1), where N is the number of neighboring RO regions

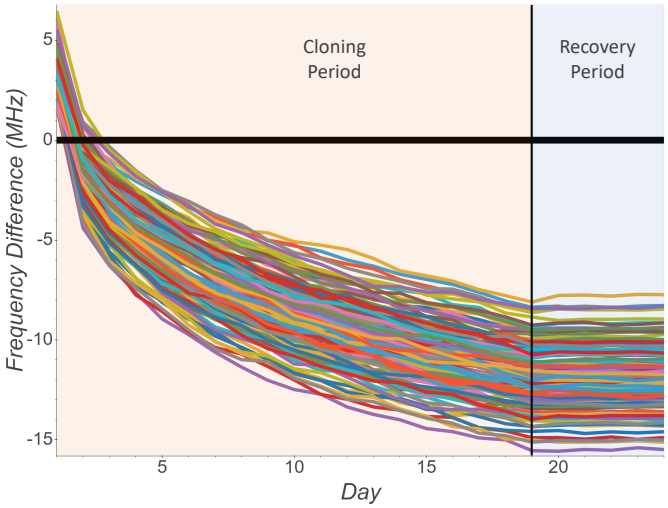


Figure 7: Graph showing the frequency difference for each of the 128 RO pairs. The corresponding bit for each pair is flipped when the frequency difference becomes negative. Targeted aging was performed for the first 19 days, while recovery was performed for the last five days.

and $f_0, f_1, ..., f_N$ are the frequency of those region. For each region, there are usually eight neighboring RO (N, NE, E, SW, S, SW, W, NW), with fewer neighbors at near the perimeter of the RO PUF.

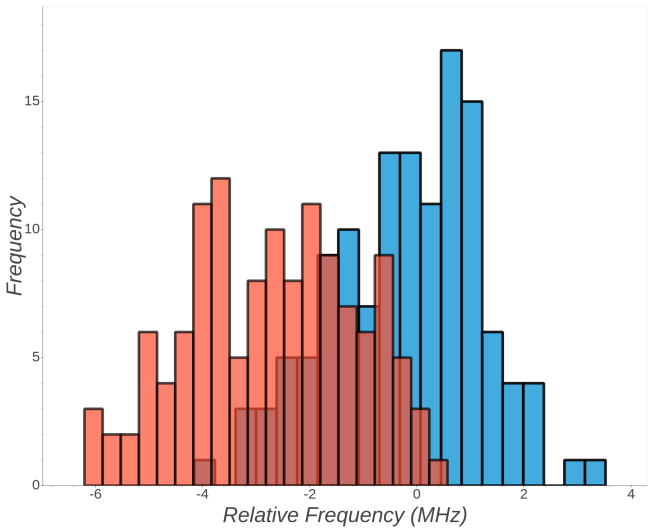
$$f_{Relative} = f_{RO} - \frac{\sum_{i=1}^{N} f_i}{N} \tag{1}$$

During targeted aging, the relative frequency of regions where enabled and disabled ROs are placed greatly increase and decrease respectively. Figure 8a shows the relative frequency distribution of the disabled RO regions from Section VI for both before and after the cloning process. The relative frequencies of the disabled RO regions are initially centered at $-0.209\,\mathrm{MHz}$ but shift to $-2.712\,\mathrm{MHz}$ after the cloning process, an absolute increase of 13.0x. Similarly, the relative frequencies of enabled ROs, shown in Figure 8b, shift the opposite direction, from a mean of $0.169\,\mathrm{MHz}$ to $3.130\,\mathrm{MHz}$, an absolute increase of 18.5x.

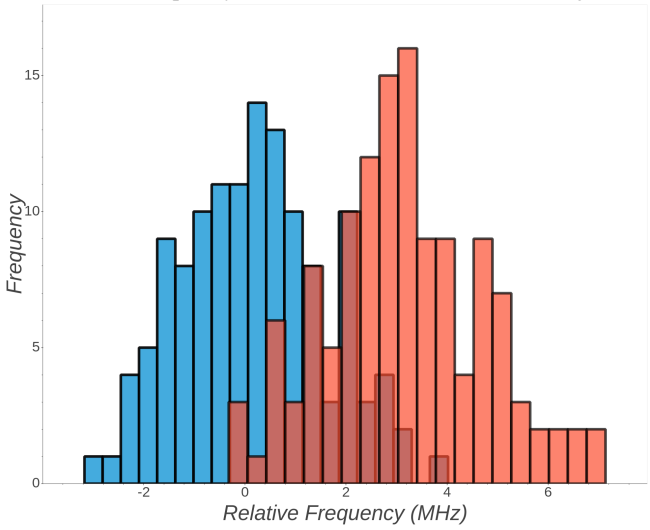
In both cases, the average relative starts near zero but shifts to a much larger value. This shift in relative frequency of target regions could be used to detect if PUF cloning has occurred in an FPGA. If so, appropriate actions could be taken to swap the FPGA out with one that is still trusted. This detection method is simple and effective, but without employing it, users of delay-based FPGA PUFs may be vulnerable to the spoofing attacks we have demonstrated in this paper.

IX. Conclusion

This paper presents a novel targeted aging technique that can be used to physically clone an RO PUF. By exposing enabled and disabled ROs to extreme heat, we respectively increase or decrease the frequency degradation those ROs experience due to BTI. This allows us to perform targeted aging on



(a) Relative frequency distribution of the disabled RO regions.



(b) Relative frequency distribution of the enabled RO regions

Figure 8: Histograms showing the distribution of the relative frequencies of enabled and disabled ROs targeted in Section VI. Both graphs include distributions for both before (blue) and after (red) the cloning process (histograms are overlapping and semi-transparent).

individual paths in an FPGA and can be leveraged to swap relative frequencies of RO pairs in a PUF, flipping the sign of their comparison and the resulting bit in the PUF response.

By flipping the right bits, an attacker can achieve any PUF response on an FPGA, allowing them to clone the RO PUF response of another FPGA. We demonstrate that not only is a cloning attack possible, but also that any arbitrary PUF response can be achieved in as little as two days.

This targeted aging technique is the first method shown to be able to physically clone a delay based PUF. This has significant implications on the security of such PUFs and may allow for new attacks targeting systems previously thought of as secure.

References

[1] U. R ührmair, F. Sehnke, J. S ölter, G. Dror, S. Devadas, and J. ü. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Conference on Computer and communications security (CCS)*, Chicago, Illinois, USA: ACM Press, 2010, p. 237.

[2] M. Gao, K. Lai, J. Zhang, G. Qu, A. Cui, and Q. Zhou, "Reliable and anti-cloning PUFs based on configurable ring oscillators," in *Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, Aug. 2015, pp. 194–201.

[3] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2013, pp. 1–6.

[4] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.

[5] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems (CHES)*, 2007, pp. 63–80.

[6] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Workshop on Hardware-Oriented Security and Trust (HOST)*, Jun. 2008, pp. 67–70.

[7] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[8] M. S. Alkatheiri, Y. Zhuang, M. Korobkov, and A. R. Sangi, "An experimental study of the state-of-the-art PUFs implemented on FPGAs," in *Conference on Dependable and Secure Computing*, Aug. 2017, pp. 174–180.

[9] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications*, 2010, pp. 382–387.

[10] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *Conference on Field Programmable Logic and Applications*, Aug. 2009, pp. 703–707.

[11] M. Mustapa and M. Niamat, "Temperature, voltage, and aging effects in ring oscillator physical unclonable function," in *Conference on High Performance Computing and Communications, Symposium on Cyberspace Safety and Security, and Conference on Embedded Software and Systems*, Aug. 2015, pp. 1699–1702.

[12] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference*, Jun. 2007, pp. 9–14.

[13] N. Weste and D. Harris, *CMOS VLSI design: A circuits and systems perspective*, 4th ed. Pearson, Mar. 4, 2010.

[14] S. Gehrer, "Highly efficient implementation of physical unclonable functions on FPGAs," 2017.

[15] T. T. Kim, P.-F. Lu, and C. H. Kim, "Design of ring oscillator structures for measuring isolated NBTI and PBTI," in *International Symposium on Circuits and Systems (ISCAS)*, May 2012, pp. 1580–1583.

[16] S. Zafar, Y. Kim, V. Narayanan, *et al.*, "A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, re gates," in *Symposium on VLSI Technology*, Jun. 2006, pp. 23–25.

[17] S. Gehrer, S. Leger, and G. Sigl, "Aging effects on ring-oscillator-based physical unclonable functions on FPGAs," in *Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Dec. 2015, pp. 1–6.

[18] V. Betz, "Architecture and CAD for speed and area optimization of FPGAs," Thesis, 1998.

[19] E. A. Stott, J. S. Wong, P. Sedcole, and P. Y. Cheung, "Degradation in FPGAs: Measurement and modelling," in *Symposium on Field Programmable Gate Arrays (FPGA)*, Feb. 21, 2010, pp. 229–238.

[20] H. Reisinger, T. Grasser, K. Ermisch, H. Nielen, W. Gustin, and C. Schlünder, "Understanding and modeling AC BTI," in *International Reliability Physics Symposium*, Apr. 2011, 6A.1.1–6A.1.8.

[21] T. Gaskin, H. Cook, W. Stirk, R. Lucas, J. Goeders, and B. Hutchings, "Using novel configuration techniques for accelerated FPGA aging," in *Conference on Field-Programmable Logic and Applications (FPL)*, Aug. 2020, pp. 169–175.

[22] H. Cook, J. Arscott, B. George, T. Gaskin, J. Goeders, and B. Hutchings, "Inducing non-uniform FPGA aging using configuration-based short circuits," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 4, pp. 1–33, Dec. 31, 2022.

[23] C. Lavin and A. Kaviani, "RapidWright: Enabling custom crafted implementations for FPGAs," in *Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2018, pp. 133–140.

[24] A. Maiti, L. McDougall, and P. Schaumont, "The impact of aging on an FPGA-based physical unclonable function," in *Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2011, pp. 151–156.

[25] F. Kastensmidt, J. Tonfat, T. Both, *et al.*, "Voltage scaling and aging effects on soft error rate in SRAM-based FPGAs," *Microelectronics Reliability*, vol. 54, no. 9, pp. 2344–2348, Sep. 2014.

[26] A. Amouri, F. Bruguier, S. Kiamehr, P. Benoit, L. Torres, and M. Tahoori, "Aging effects in FPGAs: An experimental analysis," in *Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2014, pp. 1–4.

[27] V. Jyothi and J. Rajendran, "Hardware trojan attacks in FPGA and protection approaches," in *The Hardware Trojan War: Attacks, Myths, and Defenses*, S. Bhunia

and M. M. Tehranipoor, Eds., Cham: Springer International Publishing, 2018, pp. 345–368.

[28] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive EM attack on FPGA RO PUFs and countermeasures," in *Workshop on Embedded Systems Security (WESS)*, 2011, pp. 1–9.

[29] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl, "Localized electromagnetic analysis of RO PUFs," in *Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2013, pp. 19–24.

[30] J. Guajardo, B. Škorić, P. Tuyls, *et al.*, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information Systems Frontiers*, vol. 11, no. 1, pp. 19–41, Mar. 1, 2009.

[31] C. Yilmaz, L. Heiß, C. Werner, and D. Schmitt-Landsiedel, "Modeling of NBTI-recovery effects in analog CMOS circuits," in *International Reliability Physics Symposium (IRPS)*, Apr. 2013, 2A.4.1–2A.4.4.

[32] P. Hehenberger, H. Reisinger, and T. Grasser, "Recovery of negative and positive bias temperature stress in pMOSFETs," in *Integrated Reliability Workshop Final Report*, Oct. 2010, pp. 8–11.

[33] M. Slimani, K. Benkalaia, and L. Naviner, "Analysis of ageing effects on ARTIX7 XILINX FPGA," *Microelectronics Reliability*, vol. 76-77, Jul. 1, 2017.

[34] J. Luu, J. Goeders, M. Wainberg, *et al.*, "VTR 7.0: Next generation architecture and CAD system for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 2, pp. 1–30, Jun. 2014.

[35] H. Dogan, D. Forte, and M. M. Tehranipoor, "Aging analysis for recycled FPGA detection," in *Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2014, pp. 171–176.

[36] Y. Isaka, F. Ahmed, M. Shintani, and M. Inoue, "Unsupervised recycled FPGA detection based on direct density ratio estimation," in *International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jun. 2021, pp. 1–6.