ESE 498 Capstone Design Final Report
Power Delivery Network as a Physically Unclonable Function

Patrick Naughton (patricknaughton@wustl.edu) - BS Electrical Engineering
Robert Esswein (resswein@wustl.edu) - BS Electrical Engineering, BS Computer Engineering
Advisor: Dr. Xuan ’Silvia’ Zhang (xuan.zhang@wustl.edu), ESE Department

Submitted to Professors Trobaugh and Wang and the Department of Electrical and Systems Engineering
Abstract—Physically unclonable functions leverage process variation in the manufacture of silicon chips and circuit boards to map inputs to outputs in an irreversible and unpredictable but consistent manner. They have many applications as security primitives: they can serve as truly random number generators, create secret keys, and fingerprint specific chips. These primitive functions can then be used to secure confidential information and regulate access to private resources. Current approaches tend to utilize variation in the production of silicon dies as the source of variability in their function’s output. We present a PUF that leverages variation in the entire circuit board’s power delivery network to produce an output to a given input. This allows us to uniquely identify entire boards rather than just individual chips. Because the PUF’s output also depends on the board’s PDN this approach allows us to detect hardware Trojans, malicious chips placed onto circuit boards that carry out hardware-level attacks. These chips’ presence subtly changes the impedance of the PDN allowing us to detect them. The PUF itself as well as its potential to identify hardware Trojans are evaluated here using several different copies of an FPGA development board.

I. INTRODUCTION

Security of electronic systems at the hardware level plays a critical role in maintaining the authenticity and integrity of digital communications and data processing. Nearly every electronic system is built on a printed circuit board (PCB) that provides routing between the different integrated circuits (ICs) that comprise the system. Engineers designing these circuits are responsible for protecting confidential data processed by the board and adequately managing permissions to access these data. Physically unclonable functions (PUFs) are mappings from inputs, called challenges, to outputs, called responses, that can enhance the security of a system by providing certain security primitives. For example, PUFs can provide services such as truly random number generation, secret keys, and chip identification, that serve as the back-bone for different security protocols. The mapping of the PUF from inputs to outputs is unpredictable and unique to a given chip but is reproducible. Traditional PUFs leverage process variation in the manufacture of silicon chips to implement this function.

As computer systems have grown more complex, many PCBs have begun integrating different kinds of computational resources into their designs. Field Programmable Gate Arrays (FPGAs) are reconfigurable logic units that allow computer engineers to implement custom logic circuits. Recently, designs across many different domains have begun using FPGAs because of their capacity to accelerate specific computations [7], [11], [17]. Due to their extremely flexible nature, FPGAs open up many security opportunities as well as vulnerabilities. We specifically target them because their ubiquity means that tools developed for them could relatively quickly and easily be deployed to many systems in production.

In this work, we develop a PUF that derives its randomness from process variation across the entire power delivery network (PDN) of a PCB. PUFs have many security applications such as identification and authorization, and digitally signing computation results [3]. Originally designed to authenticate specific chips [3], this work presents a PUF capable of authenticating an entire printed circuit board. We leverage the change in logic gate delays as a function of supply voltage as well as process variation in the manufacturing of printed circuit boards and their components to precisely characterize a given board’s PDN. In addition, we show how this PUF can be applied to hardware Trojan detection. Hardware Trojans are malicious circuits covertly placed onto PCBs that can disrupt a board’s normal operation [2]. Since they have direct access to hardware resources, the threat they pose cannot be mitigated with software patches alone. Detecting the presence of these circuits automatically allows compromised boards to be discarded or fixed before they are deployed.

II. BACKGROUND

Modern PCBs typically have extremely complicated PDNs that can supply different voltages to the various chips on the board. This PDN is typically composed of switching and linear regulators as well as several RLC networks that connect each IC to the appropriate supply voltages. Because all the circuits draw current from this PDN, changes in
the power consumption of one chip will generally create voltage fluctuations that affect all parts of the PDN. With an adequate sensor, these voltage fluctuations can be detected. This work is primarily concerned with the RLC network that connects different chips to the PDN. Process variation across different boards of the same type can result in different impedances observed between the power rails of a given chip.

FPGAs have recently seen increased use as a method to allow for dynamically reconfigurable hardware accelerators in many applications [18]. They are often placed onto computer PCBs or even integrated into System-on-Chips (SoCs) to provide expanded computational capabilities. By virtue of being placed on the same board (or even within the same chip), the FPGA is able to observe voltage fluctuations on the PCB’s PDN. However, FPGAs typically have no way to directly measure their supply voltage. Previous work has shown that Time-to-Digital converters (TDCs) can be implemented using standard FPGA logic primitives (buffers and latches) to measure changes in delay through the gates of the FPGA [4]. [4] also showed that these changes in delay very strongly correlate with changes in the supply voltage of the FPGA. Other work has shown that these (or similar) sensors can detect voltage fluctuations caused by other circuits on the FPGA [9], circuits on the same SoC [18], and even from other chips on the same PCB [10]. Each of these examples implemented a TDC that was sensitive enough to perform power side-channel attacks remotely.

Physically unclonable functions (PUFs) were developed to verify a particular piece of hardware, such that an attacker could not spoof another chip. A PUF must satisfy the following constraints: it must be easy to compute on the given system, it must output what looks like random noise, it must be unique from system to system including different copies of the same design, and it must be difficult for an attacker to reproduce even when the attacker has physical access [15]. PUFs are designed with challenge response pairs, typically very large challenges and responses so that the PUF is not reproducible using a large lookup table.

III. RELATED WORK

Previous work on PUFs has primarily focused on identifying individual chips. Maiti et al. demonstrated a delay-based PUF that uses ring-oscillators to create a small, portable implementation on an FPGA [8]. This PUF utilizes very few of the available FPGA resources and allows a user to uniquely identify a given FPGA (or IC if implemented as an ASIC). More recently, Spence et al. implemented an arbiter PUF whose stages can be reconfigured remotely [12]. Their design is similarly instantiated in an FPGA in order to uniquely identify it. Their design makes the PUF more resistant to attacks, namely, learning-based attacks. Outside of FPGAs themselves, Tehranipoor et al. showed that a PUF could be created using just DRAM cells already present on many computer circuit boards [14]. The PUF makes use of the fact that the initial charge on capacitors in the DRAM cells is random when the device powers up. Their results show that this approach can produce large (128 bit) PUFs that are quite stable and unique across chips, making them good candidates for use as security primitives.

While these PUFs have been shown to effectively authenticate individual chips, existing literature has not substantially addressed PUFs that fingerprint entire PCBs. Previous methods have focused on exploiting process variation within ICs rather than across entire systems. By their very nature, these PUFs cannot detect board-level modifications because the electrical properties they rely on to form their PUF all reside within one chip. This work departs from this paradigm by utilizing the PDN of the board as the source of variation for our PUF. Electrical changes to the board will then be reflected in the characterization of the PDN and can be detected with an appropriate sensing method.

Considering hardware Trojan detection, several different approaches have been employed for identifying these malicious circuits. One method attempts to analyze the bitstream or netlist of a circuit to determine if it contains malicious logic [5], [16]. However, these approaches have the obvious disadvantage that they cannot detect any hardware Trojans outside of the file that they analyze. If a part is added to the board by an attacker, it is
impossible for this analysis to reveal the threat. Another approach attempts to detect the hardware Trojan at runtime by examining the actions of the circuitry and detecting anomalous behavior [6]. This partially remedies the previous problem by looking at the overall behavior of the circuit. However, this method has a few main disadvantages. First, detecting the hardware Trojan after or while it is performing its attack could be too late. For example, if the attack launched by the Trojan crashes the system, its detection does not offer any benefit. Additionally, this approach relies on being able to somehow differentiate normal behavior from that induced by the Trojan. If the Trojan performs some unexpected action that the detection system is not calibrated to find, the Trojan may be missed entirely. Our approach departs from both of these previous lines of work by attempting to detect the very presence of the Trojan by its impact on the impedance of the PCB’s PDN. This effect cannot be eliminated or hidden because even if the Trojan is not active its presence modifies the impedance of the PDN.

IV. APPROACH

We use the impedance of the PCB’s PDN as a repeatable, unique signature to identify a given board. Because of the ubiquity of FPGAs on modern circuit boards, we perform this characterization from within an FPGA. This allows our method to potentially be employed on many existing circuit boards without any changes to the hardware on the board.

The restrictions of using an FPGA mean that it is difficult to treat this as a simple system identification problem: namely, we have only very coarse control of the input and nonlinear, uncalibrated measurement of the output. For these reasons, we do not attempt to derive an analytic model of the PDN or input; instead we simply examine several different candidate characterizations and choose the one that generates the most well-behaved outputs. For the purposes of a PUF, well-behaved responses will be stable and unique: multiple runs on the same board of the same test will generate the same output, but running the same test on two different boards should generate different outputs. In addition to these characteristics, we would also like small modifications of the board’s PDN to cause the output of these tests to change in order to enable hardware Trojan detection.

A. Voltage Measurement

We use Time-to-Digital Converters (TDCs) to measure voltage fluctuations of the PDN. TDCs are circuits which convert a delay time to a digital signal that can be directly manipulated. For this work, we use the specific TDC presented in [4] to maximize our time resolution. This TDC utilizes a chain of buffers to delay a clock signal and determine how far it propagates within half of a clock period. Figure 1 shows a block diagram of this device.

![Fig. 1: Block diagram of a TDC [4]. This design uses a latch and series of buffers to detect how far a clock edge propagates within half of the clock period (propagation depth). This value can then be saved in a flip-flop. In this case, the propagation depth is 2.](image-url)

Measuring this delay gives us an indication of the supply voltage on the board because gate delays tend to increase as supply voltage drops, causing the clock signal to propagate through fewer buffers. By measuring the transient voltage fluctuations when we stress the PDN by suddenly drawing power at different frequencies, we can characterize the overall impedance of the PDN as seen from the power port of the FPGA. Although Figure 1 shows the TDC delay line as a series of buffers, our design uses two different primitives to actually implement this. The “buffers” in our initial delay line are composed of two NOT gates. For the observable delay line, each “buffer” is one stage of a CARRY4 primitive. This was done because the delay time...
through a CARRY4 stage is generally lower than the delay through a buffer or two NOT gates, which makes the TDC more sensitive [4]. We do not use CARRY4s for the initial delay line since sensitivity there does not actually affect our measurements: we opt for NOT gates (with a longer delay time) so that we can use fewer resources and still achieve the desired delay. The TDC has 32 buffer stages in the initial delay line and 63 stages in the observable delay line.

After the propagation depth is captured in the TDC register, some basic post-processing is performed to remove noise and clear glitches, as well as to compress the data into a smaller format. First, for each bit in the TDC register except the least-significant bit (LSB), we take the AND of it with the bit immediately less significant than it. This has the effect of removing isolated 1s in the bit vector that may have been caused by glitches in the latch. Now, starting from the LSB, this cleaned vector consists of a continuous string of 1s followed by a continuous string of 0s. Since there are only 64 possible values this vector can take on (all zeros, one 1 followed by zeros, two 1s followed by zeros, etc.), we simply convert this vector into a 6 bit number by adding up the values of all the bits. This is equivalent to taking one plus the largest index that is a 1 in the vector.

B. Frequency Response

To stimulate the PDN, we instantiate a large number of ring-oscillators on the FPGA that can be enabled or disabled. We refer to this group of oscillators as a power virus. Each oscillator is simply a NAND gate with its output tied to one of its inputs and the other input serving as the enable signal. When the enable signal is asserted, the ring-oscillators begin switching which causes the virus to alternate state very quickly, suddenly drawing a large amount of power. We can then use the TDC to measure the response of the PDN as it attempts to counteract the change.

Figure 2 shows the overall block diagram of our design. The TDC and power virus are implemented in an FPGA embedded inside the Zynq-9000 SoC. We also make use of the processor on board to facilitate communication with an outside computer. The processor is able to directly access memory elements inside our TDC so that it can configure different measurement settings and read out the TDC’s output after we make a measurement. We use this processor to set the measurement window, size of the power virus, and to specify the challenge to be tested by the board. Additionally, it can communicate with an external laptop via a standard UART interface allowing us to collect and process data on a separate computer.

In order to characterize the impedance of the PDN, we wish to find its response to a varying frequency of stimulation. To find this frequency response, we vary the frequency with which we turn the virus on and off. At each clock tick, we record the propagation depth into the TDC in block RAM. We then can measure statistics of the output waveform measured by the TDC at different frequencies to generate the frequency response of the PDN.

At the same time that we collect a trace of the propagation depth of the TDC, we compute the energy contained in the waveform and subtract out the energy due to its “DC-bias.” Specifically, if we take measurements for $K$ time steps and each measurement has a propagation depth of $d_k$, we
compute \( E = \sum_{k=0}^{K-1} d_k^2 \) as the total energy of the waveform. We also compute \( S = \sum_{k=0}^{K-1} d_k \) so that we can subtract out the energy contributed by the offset of the wave rather than the oscillations themselves.

After the entire trace has been collected and these values have been accumulated in our hardware design, we use the Zynq processor to read out \( E \) and \( S \) and compute \( E' = E - S^2/K \), the original energy minus the energy contributed by the DC-offset. Subtracting off the DC-offset makes the final measurement more stable even when the overall voltage supplied to the board changes (for example, if the board is plugged into a different outlet). This final value as well as the frequency at which it was recorded is then sent over a serial bus to be recorded on a computer. By performing this measurement many times while oscillating the power virus at different frequencies, we can approximate the value of \( E' \) as a function of frequency. We use this frequency response as a characterization of a given board's PDN.

Of course, the magnitude of this energy changes if more or fewer measurements are taken. We decided to fix our recording window at 10,000 clock ticks (100 \( \mu s \)) rather than average over the recorded time in order to preserve as much information as possible (division on the board could result in information lost due to rounding). Additionally, note that the units of this energy are \([\text{depth}]^2[\text{clockticks}]\), which have no physical meaning. We do not attempt to convert this measure to a physically significant metric because it is extremely difficult to express the actual source voltage as a function of propagation depth, and because this actually makes the measure less informative. This is because propagation depth is also a function of process variation in manufacturing the FPGA as well as the PCB itself, meaning that even if we could directly find the voltage corresponding to a given propagation depth, doing so would make the effects of process variation more difficult to observe.

C. Use as a PUF

For a challenge, our PUF uses a random binary signal as the input to the enable pin of the power virus. This input was chosen because it has a low crest factor (it delivers a lot of power without having too high of an amplitude) and still stimulates many frequencies [13]. The response is simply the overall response of the PDN, measured as though we were measuring the frequency response. We use a 128-bit long signal and set the enable bit of the power virus to each bit in sequence, wrapping around to the beginning after we hit the end. We advance to the next bit of the sequence each clock tick and record the response to this challenge for 10,000 clock ticks. The final energy response (with the DC term subtracted out) constitutes the response to a given challenge. Figure 3 summarizes this process.

D. Detection of Hardware Trojans

Hardware Trojans can be very small chips that remain inactive until a specific “trigger” causes them to perform some action [2]. Because the chips can remain dormant, it is not sufficient to simply measure power consumption of the board to detect them. However, since these circuits will eventually have to draw some power to perform their malicious computations, they will likely be attached to the PDN of the PCB somehow. Since the chip will necessarily have some impedance, its addition will subtly change the frequency response of the PDN. Our approach allows us to detect this change even if the chip remains dormant.
Since our PUF is designed to derive some of its variation from the PDN of the circuit board itself, the addition of a hardware Trojan should impact the response of the board to a given challenge. Thus, we simply utilize the same types of challenges to try to detect a Trojan. Since the change to the PDN is somewhat subtle, the effect on the response to a given challenge can be small depending on the particular frequencies interrogated by a given challenge. Thus, in order to reduce the error from this variability, we examine the response to many more challenges when explicitly checking for a hardware Trojan rather than just examining the difference between boards.

V. EXPERIMENTS

We used the ZedBoard [1] as a testbench for our experiments. The ZedBoard has a Zynq-9000 SoC that contains an ARM Cortex-A9 based processor as well as programmable logic in the form of look-up tables (LUTs) and flip-flops. The processor allows us to easily interface between our hardware design and external computer in order to record measurements. We implemented a TDC using 32 double NOT gate buffers for the initial delay line and 63 CARRY4 stages for the observable delay line. We instantiated 128 groups of ring oscillators that each contain 128 oscillators. This allows us to finely control how much power the virus draws by modulating the number of ring oscillators switching. By activating more ring oscillators, we draw more current. We can specify which groups get activated from the Zynq by supplying a 128-bit bitmask. Once triggered by the Zynq, the hardware on the FPGA will activate the power virus (whichever groups are specified by the bitmask) in some pattern and simultaneously record the values output by the TDC for 10,000 clock ticks. Our clock runs at 100MHz so we record for a 100μs window. In order to verify the operation of our TDC, we began by detecting a simple step function in which we record some baseline measurements of the TDC before activating the power virus and observe the change in TDC values. In addition, we oscillated the power virus in a square wave and examined the resulting readings from the TDC. Figure 4 shows plots of the traces gathered from these two experiments.

A. Frequency Responses

To characterize the frequency response of the board, we oscillate the power virus in a square wave at different frequencies and measure the energy of the output, subtracting off the energy contributed by the DC-offset. We activate 52 groups of the power virus so that a total of 6656 ring-oscillators are enabled when the virus is on. We record the board’s response for 10,000 clock ticks at 100MHz and compute the energy response (without the DC component) for many frequencies to find the overall frequency response. Starting with a period of 2 clock ticks, we multiply this period by 1.01 between measurements to obtain (logarithmically) equally spaced measurements in the frequency domain. Because the period of oscillation has to be an even integer number of clock ticks, we must round the new computed period up to the closest such value. This is not a problem at higher periods, but when the period is small, this limitation prevents us from measuring with a small frequency resolution. For this reason, our data at high frequencies are more spaced out. In total, we measure the frequency response at 395 frequencies from 10kHz up to 50MHz.

We collect each frequency response 100 times to capture noise in our measurements. Figure 5 shows a plot of these responses for one ZedBoard. The response is remarkably consistent between runs except for a few outliers at higher frequencies.

Using the unmodified board’s frequency response as a baseline, we then tested to see if we could detect changes to the board’s PDN by observing how the changes affect the frequency response. We model a hardware Trojan as a simple capacitor. Although a real Trojan would undoubtedly contain more components than this, this experiment makes it more difficult to detect the presence of a Trojan because the addition of more components would likely result in larger changes to the response. Additionally, many Trojans remain dormant for long periods of time, effectively electrically insulating most of their components from the board’s PDN.
(a) Response of the PDN measured by the TDC when we turn all of the ring oscillators on after 5,000 clock ticks (50 µs). The voltage fluctuations are clearly visible in the transient and the steady-state measurement is lower while the power virus is on.

(b) Measurements from the TDC when the power virus oscillates in a square wave at 24,801 Hz. The measurements reflect the response we expect where the propagation depth drops whenever the virus is on and recovers when we turn it off.

Fig. 4: Plots of the initial measurements taken by the TDC to verify its operation.

Fig. 5: Plot of 100 energy responses taken from the same board. The response is remarkably consistent except for a few outliers at high frequencies.

Fig. 6: The location of C206 on the Zedboard. [1]

By testing whether or not we can detect just a capacitor, we provide evidence that we could detect these Trojans even while they remain dormant. We examine the impact of connecting a 1 µF, 100 nF, and 10 nF capacitor in parallel with a 10 nF decoupling capacitor for the internal voltage of the FPGA (specifically C206 [1]). Figure 6 shows the location of this capacitor on the PCB. We then measured the frequency response of the board 100 times in the same way that we collected the baseline response.

Figure 7 shows how the energy response changes as a result of attaching different capacitors to the board in parallel with C206. The solid lines
Fig. 7: The impact of adding capacitors to the PDN on the frequency response of the board. The average across 100 runs is shown as a solid line with the shaded regions showing \( \mu \pm 3\sigma \). This figure focuses on the portion of the frequency response with the largest divergence between the traces.

show the average response over 100 runs while the shaded areas show \( \pm 3\sigma \) bounds (where \( \sigma \) is the standard deviation of the response across runs of a given frequency). This figure shows the area of the responses that demonstrated the greatest divergence which occurs at relatively high frequencies around 10MHz. Additionally, the standard deviation is quite small for lower frequencies around 1MHz compared to its value at about 20MHz. This corresponds to the outliers observed in the raw energy responses. Despite this increased variance, the differences between the responses of the modified board and the unmodified board are much larger than the variations observed between runs using the same configuration. This indicates that there is detectable variation introduced by the addition of a capacitor to the PDN that could be exposed as a change to a challenge’s response.

To test our challenge framework we randomly generated 1,000 128-bit challenges and collected the energy responses that the board output. We used the same challenges to distinguish between an unmodified board and a board with a simulated hardware Trojan attached as well as to differentiate between different boards. Figure 8 shows a histogram that displays the frequency of different distances between the unmodified board’s average response to a challenge and a modified board’s average response to the same challenge. This distance is normalized by computing the t-test statistic for the difference between means which we calculate using Equation 1:

\[
t = \frac{|\bar{x}_1 - \bar{x}_2|}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}
\]

Here, \( \bar{x} \) is the average response of the board to a given challenge over 100 runs and \( s \) is the standard deviation of this response. \( n_1 = n_2 = 100 \) in this case since we test every challenge 100 times.

In addition to testing modifications to the board we also collected samples from the unmodified board again to estimate how much the response changes without modifications to the board between runs and to compare this variation to that observed when we actually modify the board. As the histograms show, the addition of extra components causes the responses of more challenges to be farther from the baseline response.

To test the uniqueness of our PUF between boards we ran the same test as for detecting hardware Trojans but tested on multiple different PCBs. Again, we performed the same test twice on the same board to compare how much our measurements change between runs with the variation observed from changing boards entirely (shown as “Board 0” in Figure 9). Figure 9 shows the results of this experiment. For a given challenge, there tends to be a much larger difference between boards than results from simply adding a capacitor to the PDN. In addition, the two boards tested show a substantial difference between their responses as well as between their response and the baseline board, which is desirable because we would like to distinguish between any two boards, not just determine whether or not a board is different from this particular baseline board.

VI. RESULTS

These results indicate that a TDC implemented in an FPGA is sufficiently sensitive to act as a sensor
Fig. 8: Histograms of the t-statistic normalized distance between the average response of the baseline and the average response of the test board to a given challenge. Modifying the PDN of the board results in a higher probability that the response to a given challenge will be farther from the response of the unmodified board. Here, we modified the PDN by attaching different valued capacitors to a decoupling capacitor (C206) near the FPGA.

Fig. 9: Histograms of the t-statistic normalized distance between the average response of the baseline and the average response of the test board to a given challenge. Different boards generate responses very far from the original board’s response.

For a PUF that secures an entire PCB rather than a single chip. In order to identify a given board or determine if it has been modified, we examine the t-test statistic of the average response of the board to a given challenge over 100 runs. To identify a board it is generally sufficient to examine the response to just one challenge. If we set a threshold at \( t = 40 \) the probability of incorrectly rejecting a board is just 0.9% and the probability of incorrectly accepting a different board is just 2.3% in the worst case.

In terms of detecting a hardware Trojan, using this same \( t = 40 \) threshold, the probability of incorrectly accepting a board which has been modified is substantially higher: we observed a 79%, 72%, and 71% false acceptance rate when testing one challenge for a board with 1\( \mu \)F, 100nF, and 10nF capacitors attached respectively. This is because the difference in impedance of the board’s PDN induced by a single capacitor modification is of course much less than that caused by swapping the entire board. Additionally, when switching boards, we also leverage process variation in the manufacture of the FPGAs themselves, resulting in larger differences in the boards’ responses to a given challenge. For this reason, it is necessary to examine a board’s response to multiple different challenges to accurately detect a hardware Trojan.

If we say trials where the response’s t-statistic is above our threshold of 40 are “successes,” the random variable number of successes \( Q \) has a binomial distribution. If we test \( n \) independently randomly selected challenges and we assume that the true proportion of successes is \( s = 0.009 \) (i.e. that the board is unmodified) then we can compute the probability of observing at least \( Q \) successes using Equation 2:

\[
\sum_{i=Q}^{n} \binom{n}{i} (s)^i (1 - s)^{n-i}
\]  

(2)

Performing a standard \( p \) test, we can make the determination that a given board has a hardware Trojan if this resulting probability is less than some \( \alpha \) threshold such as \( \alpha = 0.001 \). Let \( Q^* \) be the first number of successes for which this probability
is less than or equal to $\alpha$. We can then analyze the probability that we falsely accept a board by noting that if a board has been modified, the true $s$ value is (at least) $s = 0.2$. A false acceptance occurs when $s = 0.2$ but $Q < Q^*$. For example, if we test 100 challenges, $Q^* = 5$ if $\alpha = 0.001$. The probability of a false acceptance in this case is $1.9 \cdot 10^{-5}$. Thus, by testing multiple challenges, we can reduce the probability of false acceptance and false rejection by more accurately estimating the underlying probability distribution. This same analysis can be applied to distinguishing between boards as well. Testing 100 challenges 100 times each takes just $10^7 ns = 10 ns$ which is completely feasible so long as this PUF is tested somewhat rarely.

VII. DISCUSSION AND FUTURE WORK

Our final design uses about 25% of the FPGA’s LUTs. This is primarily due to the power virus which we made as large as possible to provide the largest signal we could and also to make the virus reconfigurable via a C program (so that it did not require resynthesizing our design to change the number of oscillators switching during our measurement period). While this does leave most of the FPGA still available, this PUF does require a large amount of resources considering how rarely it would be used in practice. Additionally, in our experiments we used fewer than 7,000 ring oscillators, meaning that we could reduce this footprint substantially by simply eliminating unused oscillators. This improvement is expected to reduce our utilization to about 13% of the FPGA: still substantial, but markedly less.

The main limitation of this work was the strength of the signal required to effectively differentiate between boards and especially to detect hardware Trojans. This problem manifests as a requirement for many ring oscillators to be able to distinguish between boards. This causes the PUF to take up many resources in the FPGA and makes it less suitable for actual use. As mentioned previously, we could substantially reduce the required resources by simply eliminating unused ring oscillators. However, this improvement alone would likely be insufficient to bring the resource utilization down to a reasonable level. An interesting future direction for this work would be to determine if a smaller power fluctuation can be induced to still achieve detection of hardware Trojans and board identification. In addition, we would like to investigate a more efficient way to produce the same voltage fluctuations. The combination of these two improvements could substantially reduce the footprint of our PUF and make it more suitable for production use alongside other logic.

Another interesting future direction would be to test this PUF’s ability to detect hardware Trojans co-resident on the FPGA. Many hardware Trojans are introduced in firmware updates rather than via physical chips, so detecting these threats would make the PUF more effective at protecting PCBs. This provides further motivation for reducing the size of the PUF so that it can always sit on the FPGA without disrupting an engineer’s ability to place other logic on the chip.

VIII. CONCLUSION

We present a novel PUF that serves to authenticate an entire PCB rather than a single integrated circuit. By leveraging process variation in the manufacture of the components of the PDN, our PUF can uniquely identify a given board based on its responses to a set of challenges. In addition, because the response to a given challenge is influenced by the entire PDN, this PUF can detect the presence of hardware Trojans using the same tests used to identify the board. Because any hardware Trojan will necessarily have some impedance, its presence changes the response of the PUF. This detection can be performed using a reasonable number of challenges.
In this section we describe various experiments and techniques that were tried that were not included in the main report because they were not as effective or were less relevant than the sections we did include. We discuss the sensitivity of our detection scheme to the placement of the hardware Trojan, different frequency responses we analyzed, variations on the challenge-response pair we present, and conclude with some remarks about the TDC itself.

A. Sensitivity to Placement of the Capacitor

An experiment we performed that had a negative result was an attempt to detect capacitors that were placed at the peripheral module (PMOD) ports of the ZedBoard. We collected the power response of a board with and without the capacitor to compare and see if there was a qualitative difference between the responses. Figure 10 shows the results: the responses are essentially indistinguishable.

![Fig. 10: Power responses of the board with and without a capacitor attached. This capacitor was connected between the 3.3V and ground rails of a PMOD port (JD). The responses are essentially indistinguishable.](image)

This result was somewhat expected. The PMOD ports are very far from the FPGA that runs the voltage monitor, so the capacitor should only very minutely change the response of the PDN. Additionally, the capacitor was attached to the 3.3 V power supply while the FPGA’s core voltage is attached to a 1 V network. These two networks are powered by different regulators, and so we expect changes on one to be difficult to detect on the other. While it would have been encouraging if there was substantial difference between the two responses, this is a limitation of our implementation. Fortunately, Trojans located far from the FPGA itself are less suited to stealing information because it is also more difficult for them to covertly gather information for the same reason that it is difficult to detect them in the first place. Thus, it is unlikely that a real hardware Trojan would be placed here and if it were, it would be very difficult for it to affect the operation of the board.

B. Other Frequency Responses

Besides using the energy response of the PCB’s PDN we also experimented with other types of responses to the power virus oscillation, namely, the peak-to-peak response as well as different components of the Fourier transform of the signal.

1) Peak-to-Peak Response: We computed the peak-to-peak of the TDC signal at different frequencies to find the peak-to-peak frequency response of the trace. We experimented with this response because it is extremely easy to compute on the FPGA itself which allows us to quickly test many frequency points and re-sample each frequency to gain insight into the noise of the response. Figure 11 shows plots of many frequency responses collected across different boards and SoCs.

As is clear from the figure, the response is neither stable nor very unique between different boards. For this reason, although it is an easy statistic to calculate, we decided to pursue responses that maintain more information from the original time-domain trace.

2) Fourier Transform Response: We tested using different components of the Fourier transform as the frequency response. Figure 12 shows frequency responses obtained by taking just the power at the fundamental frequency (12a) as well as from taking
the power in the entire bandwidth of the response (excluding the DC component) (12b).

The power response across the entire bandwidth is both more stable and more unique than examining just the power at the fundamental frequency. However, while the power response in the whole bandwidth does look promising, we decided against pursuing Fourier analysis as a method of producing a response from the board. This decision was made because of the difficulty of computing the Fourier transform. If we compute it off-board on a dedicated computer, there is a very tight information bottleneck in that we have to read the propagation depth of the TDC at all 10,000 time steps. Because the UART connection to the board is relatively slow, this approach would severely increase the time it took to compute each response. We then experimented with computing the Fourier coefficients on the FPGA. However, this proved too imprecise to actually be useful. Finally, since the power in the whole bandwidth is more informative anyway, we don’t actually need all of the information provided by the full Fourier transform, because we can compute the same thing using the RMS value of the TDC propagation depth (along with its average value to subtract out the DC-component). Thus, this analysis did not really offer any advantages over computing RMS values, so we did not pursue it further.

3) Including the DC-offset in the Response: We initially tried gathering the energy response of the board without subtracting out the DC component. If we do not subtract out this energy, our characterization of the PDN is not robust to small changes in the supply voltage to the overall board. Figure 13 shows how changing which outlet the board is plugged into causes the response to shift, making true differences in the PDN more difficult to detect. Note that these two outlets were located in different buildings, one in a residential area, and one in a
more commercial area, but the same board was tested in both cases.

Fig. 13: Energy response of the PDN with the DC term included. The only thing that changed between these two runs was which wall outlet was plugged into.

As a result, we modified the statistics we collected so that the DC energy contribution could be subtracted from the response value. This resulted in a more stable characterization of a given board as discussed in the report.

C. Variants of the Challenge

1) Virus Placement Challenge: Our initial attempt at using this as a PUF was to create 128 groups of 128 ring oscillators and make a challenge a 128-bit bitmask specifying which virus groups should actually oscillate during our TDC measurement. This gave us a total of 16,384 ring oscillators. We were limited to about 128 oscillators in each group because of the limited number of LUTs on the FPGA. This idea did not work as a PUF for two main reasons. First, changing the configuration of oscillators (while keeping the total number the same) did not substantially change the board’s energy response. Additionally, even adding one or two more groups of oscillators had very little effect. Second, many potential bitmasks would have to be ruled out because, based on previous experiments, the board’s PDN could only support oscillation of about 7,000 ring oscillators at a time without crashing due to the large magnitude of the voltage fluctuations. Thus, this idea did not end up producing a usable PUF.

2) Challenge Scaling: One attempt at using this sensor to develop a PUF involved using a 128-bit long string as an oscillation pattern for the power virus. That is, the string encoded when the virus should be enabled and when it should be disabled. In order to have the virus oscillating during a majority of our 10,000 clock tick measurement window (to make the signal sufficiently strong), we set the virus’ enable pin to a given value of the bit string for 64 clock ticks before moving on to the next bit of the challenge. This meant that the virus would be controlled for 8192 clock ticks. After this point, the enable pin simply remained in whatever state it was last set to. This idea did not produce substantial differences in responses when we added different capacitors to the board. This is because, since each bit of the challenge is repeated for so long, this format does not allow for the interrogation of high frequencies, where different responses are actually visible. Thus, we opted to simply repeat the challenge bit string so that high frequencies would also be interrogated.

3) Faster Challenge Clock: We also experimented with clocking the enable pin to the power virus at a higher frequency so that the board’s response at higher frequencies could be collected. We selected a 250 MHz clock for the power virus for a couple of reasons. First, we found from several experiments that higher frequencies would yield better variation across boards than lower frequencies. The smallest we can make the period of the power virus activation is 2 clock ticks, so we are able to clock the power virus up to 125 MHz. Since we still only record measurements at 100 MHz, it is possible that this causes most of the energy to be contained in frequency components above the Nyquist frequency (50 MHz). We do not believe that this is a problem because we do not attempt to reconstruct the original signal. We only examine the overall energy response, so frequency aliasing does not affect this response. This is related to the second reason that we chose 250 MHz for the clock fre-
quency: if we had chosen 200 MHz and switched our virus at the maximum frequency of 100 MHz, nearly all of its energy (in the fundamental as well as at harmonics) would alias to 0 Hz which we subtract out. Thus, we use 250 MHz so that our maximum frequency does not alias to DC.

D. Analysis of the TDC

1) TDC Using Only NOT Gates: Our initial design of the TDC used two NOT gates as a buffer for both the initial and observable delay lines. This produced reasonable results, but we eventually moved to the CARRY4 primitive for the observable delay line because [4] suggests that the lower delay through this primitive results in a more sensitive sensor (the same change in supply voltage will generate a larger change in propagation depth). We did observe this effect to some extent, but it was not extremely substantial; the typical baseline propagation depth increased from about 28 to about 31.

2) Time-Domain Response: The time-domain behavior of the TDC measurements exhibited some unintuitive behavior that we were never able to explain. When examining the step response of the propagation depth, we see that it actually initially increases before dropping below its baseline level. Depending on the number of ring-oscillators switching, the steady state propagation depth with the power virus enabled was sometimes higher than the steady state depth before enabling the virus. Since the propagation depth should be related to the supply voltage in the same direction, this result seems to indicate that the supply voltage actually rises when we draw more current. We attempted to verify this result using an oscilloscope and measuring the voltage across the current sense resistor on the ZedBoard, but this measurement was so noisy that no conclusion could be drawn from it. Measurements closer to the FPGA itself were not made because of the difficulty of safely accessing these pins.

We hypothesize that the higher steady-state voltage sometimes observed with the power virus enabled could be the result of the power delivery network overcompensating for the current draw from the FPGA. However, the initial increase in propagation depth observed in the transient response is much more difficult to explain. Examining the BOM of the ZedBoard [1], the linear regulators closest to the FPGA have transient load responses on the order of microseconds or tens of microseconds, whereas the time resolution achieved with the TDC is 10 ns. Thus, we believe it unlikely that we are simply missing the downward spike and the regulators are compensating before we can take our next measurement. It’s possible that this spike is actually introduced by our sensor itself, although we are unsure of the exact mechanism by which this would occur.
REFERENCES