2018 IEEE Symposium on Security and Privacy

FPGA-Based Remote Power Side-Channel Attacks

Mark Zhao and G. Edward Suh
Computer Systems Laboratory
Cornell University
Ithaca, New York
yz424@cornell.edu, suh@ece.cornell.edu

***Abstract*—The rapid adoption of heterogeneous computing has driven the integration of Field Programmable Gate Arrays (FPGAs) into cloud datacenters and flexible System-on-Chips (SoCs). This paper shows that the integrated FPGA introduces a new security vulnerability by enabling software-based power side-channel attacks without physical proximity to a target system. We first demonstrate that an on-chip power monitor can be built on a modern FPGA using ring oscillators (ROs), and characterize its ability to observe the power consumption of other modules on the FPGA or the SoC. Then, we show that the RO-based FPGA power monitor can be used for a successful power analysis attack on an RSA cryptomodule on the same FPGA. Additionally, we show that the FPGA-based power monitor can observe the power consumption of a CPU on the same SoC, and demonstrate that the FPGA-to-CPU power side-channel attack can break timing-channel protection for an RSA program running on a CPU. This work introduces and demonstrates remote power side-channel attacks using an FPGA, showing that the common assumption that power side-channel attacks require specialized equipment and physical access to the victim hardware is not true for systems with an integrated FPGA.**

I. Introduction

As we increasingly rely on hardware acceleration to improve the performance and energy efficiency of computing systems, Field Programmable Gate Arrays (FPGAs) have recently been widely adopted in large-scale datacenters. For example, Amazon offers FPGA instances in its EC2 service, allowing customers to rent FPGAs in its cloud computing environment [1]. Microsoft heavily utilizes FPGAs in its datacenters for various tasks ranging from web searches to network crypto and machine learning [2]. Similarly, Baidu also accelerates deep neural networks in its datacenters with FPGAs [3]. Furthermore, hardware vendors such as Intel and Xilinx have introduced heterogeneous System-on-Chip (SoC) designs, which integrate both processing cores and FPGA fabric in one silicon die. These FPGA-based SoCs will likely to be utilized in mobile and embedded applications.

In this paper, we show that these integrated FPGAs introduce a new security vulnerability that can be exploited to perform power side-channel attacks in software, without requiring physical access or proximity to the target system. Power side-channel attacks infer confidential information based on the data-dependent variations in a target system's power consumption [4]. In order to obtain power consumption traces, traditional power analysis attacks require physical access to the system; attackers insert a low-impedance resistor in series with the power supply and use an oscilloscope to measure the power consumption as the voltage drop across the resistor. In this paper, we demonstrate that an on-chip power monitor can be constructed using the programmable logic of an FPGA, allowing us to measure dynamic power consumption with sufficient resolution to enable power analysis attacks. In essence, the integrated FPGA opens the door for remote power analysis attacks.

This FPGA-based power side channel may be exploited in a variety of system architectures that allows an untrusted user to program a part of an FPGA. In cloud computing infrastructures, many studies from both academia and industry have proposed mechanisms to virtualize and share FPGAs among multiple users so that multiple accelerators co-reside on one physical FPGA [5]–[10]. Even in cloud platforms where each FPGA is allocated to a single user, untrusted user logic is co-resident with privileged control logic called the 'shell' [11]. Similarly, in personal computing platforms, an FPGA fabric can be shared among multiple programs, including a potentially malicious application. In such a shared FPGA platform, we show that an FPGA-to-FPGA attack, where an attack circuit in one part of the FPGA steals a secret used by another circuit on the same FPGA, is viable.

As a concrete example, we demonstrate a simple power analysis (SPA) attack on an RSA accelerator on an FPGA. In this example, we implement an RSA decryption engine and a power monitor on one FPGA, but isolate both logically and physically; there is no connection between the two modules and they are implemented at physically different locations on the FPGA. This is analogous to either two users or one user and privileged control logic co-resident on one FPGA. Our experiments show that the FPGA-based SPA can reliably recover RSA private keys only with a small number of power traces and without manual efforts; the power analysis can be fully automated.

In addition to the attacks within an FPGA, we also study attacks on an FPGA-CPU SoC. Surprisingly, we found that an FPGA power monitor can not only detect the power consumption of an FPGA but also the power consumption of other components on an SoC, in particular a CPU. This implies that various FPGA-to-CPU attacks are possible where the malicious logic on an FPGA can monitor the power consumption of software programs running on a CPU core. Our experiments show that the FPGA power monitor can indeed detect software operations on an ARM processing core. The CPU power trace can also be used to enable timing

DOI 10.1109/SP.2018.00049

attacks on software programs because the power consumption reveals when internal program operations start and end even when outputs are delayed. To illustrate the timing attack based on the power monitor, we show how an RSA private key can be recovered even when a secure modular exponentiation function, which inserts delays to yield a constant execution time, is used.

The following summarizes the main contributions of this paper.

- This paper points out the power side channel vulnerability that an integrated FPGA introduces. Unlike traditional power analysis attacks, this new vulnerability is exploitable purely in software, opening the door for remote attacks over the network.
- This paper provides a detailed study of the capabilities of an FPGA-based power monitor circuit using ring oscillators (ROs).
- This paper experimentally demonstrates multiple FPGA-based power side-channel attacks, both FPGA-to-FPGA and FPGA-to-CPU.
- This paper demonstrates that a power analysis attack can be performed using another power monitor circuit based delay lines, and discusses tradeoffs between ROs and delay lines in the context of power side-channel attacks.
- This paper discusses potential countermeasures to prevent FPGA-based power analysis attacks.

The rest of the paper is organized as follows. Section II presents our threat model and the overview of the proposed attack. Section III shows how an on-chip power monitor can be built on an FPGA and experimentally evaluates the RO-based power monitor. Section IV demonstrates a simple power analysis (SPA) attack on an RSA accelerator on an FPGA. Section V demonstrates attacks on software programs running on a CPU. Section VI discusses other types of FPGA-based power monitors, limitations of the proposed FPGA-based power side channel, and potential countermeasures. Section VII discusses related work, and Section VIII concludes the paper.

II. Overview

In this section, we describe the threat model that we assume for systems with an integrated FPGA and discuss how the FPGA can be used to enable power side-channel attacks even without physical access. Then, we further discuss potential attacks that may be performed using the FPGA power side channel.

A. Threat Model

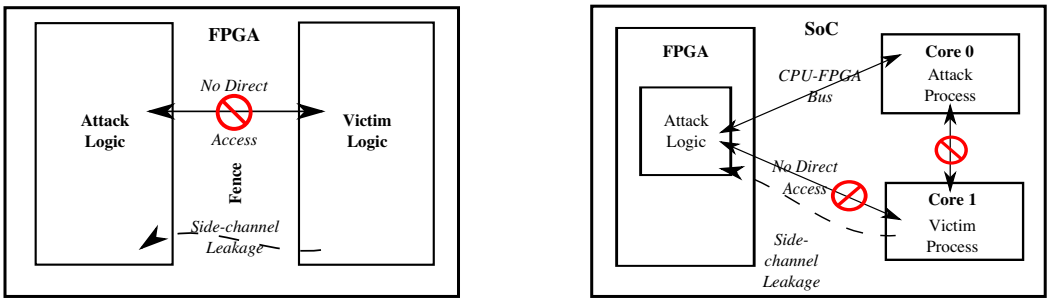
There are two approaches to integrate FPGAs into modern computing systems. In today's datacenters, FPGAs and CPUs are usually integrated as discrete components and are thus implemented as separate chips on a board. The processor and the FPGA communicate through an off-chip bus and may share main memory (DRAM). Alternatively, FPGA fabric may be integrated into a single SoC that includes other components such as general-purpose cores and GPUs, as in Xilinx's Zynq Series or Intel's SoC-FPGAs. These SoC architectures are likely to be deployed for mobile and embedded systems thanks to their smaller footprint and lower power consumption.

In this study, we consider the threats from an adversary who can program a part of an integrated FPGA to implement a circuit of his or her choice. The goal of the adversary is to steal secret information from other parts of the FPGA or the host system. We do not consider attacks on the availability or the integrity of a system. We also assume that hardware designs on an FPGA and software programs on the host system themselves are not secret and may already be known to an attacker. In that sense, we do not consider reverse-engineering attacks on the hardware IP (Intellectual Property) on an FPGA. We assume that the adversary does not have physical access to the system and therefore cannot directly measure physical properties such as power consumption.

In a typical FPGA design flow, a user specifies the logic that he or she wants to implement on an FPGA using a hardware description language (HDL) such as Verilog or VHDL. Then, an FPGA synthesis tool such as Xilinx Vivado or Intel Quartus translates this register-transfer level (RTL) code into a bitstream that describes a concrete hardware implementation on an FPGA. The FPGA tool allows users to specify design constraints to control low-level implementation details such as I/O pins, frequency, placement, routing, and others. We assume that an adversary provides HDL code that is used to generate a bitstream and control the programmable logic for a part of an FPGA. We study both cases where the adversary is allowed to use design constraints to specify low-level details for placement and routing and where the adversary cannot use the placement and routing constraints.

There are two types of attacks that we consider in our threat model. In the first case, we consider an FPGA that is shared among multiple users or processes for efficiency. For example, in proposed datacenter architectures, a malicious user can rent a virtual FPGA instance and configure programmable logic on one part of an FPGA while another user rents another virtual FPGA mapped to another part of the same physical FPGA [5]–[10]. In today's datacenter, the user logic and the operator's privileged 'shell' logic are both implemented on the FPGA even when only one user is assigned per FPGA [11]. In a shared FPGA, the malicious user may implement an attack circuit on one part of the FPGA with a goal to steal a secret from a victim's circuit on the same FPGA. We call this an FPGA-to-FPGA attack and show the scenario in Figure 1a. We assume that the shared FPGA has common security mechanisms to ensure isolation between different users; the attack and victim circuits are both logically partitioned (i.e. there is no illicit connection between the two modules) [12] and physically partitioned (i.e. with a 'fence' of unused configurable logic blocks) [13].

In the second case, we consider FPGA-to-CPU attacks where an attack circuit on the FPGA is used to extract a secret on a CPU (or a GPU) on the same SoC. Here, we assume an SoC architecture that contains both FPGA fabric and traditional processing elements such as CPUs and GPUs



(a) FPGA-to-FPGA attacks on a shared FPGA.

(b) FPGA-to-CPU attacks on a heterogeneous SoC.

Fig. 1. Threat models.

as shown in Figure 1b. The attack may also work for a discrete FPGA that shares the same power supply with a CPU on the system. Yet, we do not study the discrete FPGA case in this paper. We assume that the system has proper protection mechanisms to prevent direct accesses from an FPGA fabric to the rest of a system that is used by another user or process.

B. FPGA-Based Power Side-Channel Attacks

Traditional power side-channel attacks require physical access to the target system in order to measure the power consumption. Under our threat model, such power side-channel attacks through direct power measurements are not allowed. Similarly, given access control and partitioning mechanisms, an attacker cannot directly access confidential data that belong to another user or process.

In this work, we propose to use an FPGA as a power monitor to enable power side-channel attacks in software without physical proximity to the target system. If programmable logic on an FPGA can be used to implement a power monitor, it allows an adversary who can program the FPGA to measure the power consumption of other logic on an FPGA or programs running on a CPU/GPU of the same SoC. Because an FPGA is programmed by loading a bitstream in software, an adversary who has permission to program at least a part of an FPGA can perform power side-channel attacks remotely.

The following section discusses how a power monitor can be realized on an FPGA using its programmability. Then, we will demonstrate both FPGA-to-FPGA and FPGA-to-CPU attacks in the following sections.

C. Attack Scenarios and Use Cases

The capability to monitor the power consumption using an FPGA can be used in various attack scenarios or in potential countermeasures. Here, we briefly discuss different attacks and security mechanisms that can utilize the FPGA-based power side channels.

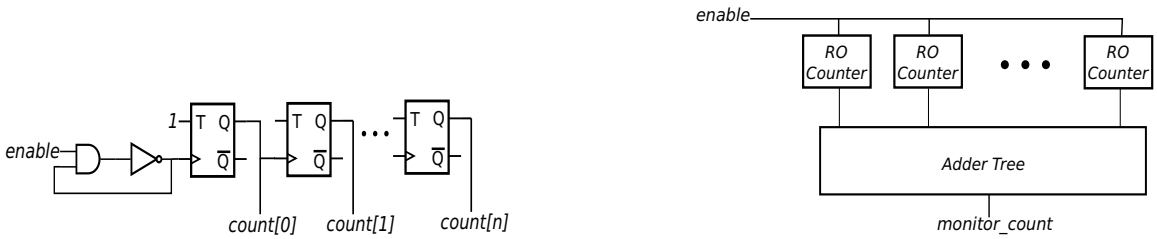
Side-channel attack. The power consumption of a cryptographic operation often depends on a secret key value. The FPGA-based power monitor, with sufficient time and power resolutions, can be used for traditional power side-channel attacks to learn secrets used by other parts of the FPGA and programs on a processing core on an SoC.

Covert-channel attack. It is relatively easy to create a circuit on an FPGA or a program on a CPU that is designed to intentionally control the level of switching activities, which determine the power consumption. Having capabilities to both control and monitor power consumption, an adversary can create covert channels either between different modules on an FPGA or from a CPU to an FPGA. The covert channel can be used to bypass traditional access control mechanisms to intentionally leak secrets.

Timing attack using the power side channel. The execution time of a sensitive computation often depends on the value of a secret due to data-dependent control flows or microarchitectural behaviors such as cache hits or misses. Under the threat model where an adversary can only observe the timing of input/output operations, a natural countermeasure against timing attacks is to delay outputs to hide their timing variations. However, simply delaying an output consumes less power compared to performing actual computations, and the internal timing of the computation may still be reflected in its power consumption trace. Therefore, the FPGA-based power monitor may be used as a way to measure hidden timing information and perform a timing side-channel attack.

Program identification. Different operations in a program lead to different levels of power consumption. For example, a floating-point operation may consume more power compared to an integer operation. Cycles when a processor is stalled due to a cache miss will consume less power compared to cycles with active computations. As a result, the power consumption trace may be used as a way to identify which program or hardware accelerator is running on a system and lead to a privacy concern. As an example, a recent study by Delimitrou and Kozyrakis [14] shows that a program in the cloud computing environment can be identified using microarchitecture behaviors. A similar attack may be performed using the power consumption trace.

Attack detection. The FPGA-based power monitor may also be used to *defend* against attacks. For example, a recent study [15] showed that a malicious user could perform a denial-of-service attack on an FPGA by programming a malicious power virus to trigger transient voltage emergencies not detectable by traditional thermal sensors or slower integrated



(a) A single RO counter.

(b) Adder tree to combine multiple RO counters.

Fig. 2. A Ring Oscillator (RO) based on-chip power monitor design.

power monitors. An on-chip power monitor capable of sensing nanosecond-scale transients can be used to detect and prevent such a power-based DoS attack. Kim et al. [16] showed that embedded power monitors can be used to detect malware in mobile systems. In a similar manner, the FPGA-based power monitor may be used to detect malicious or compromised software based on the power signature or anomaly in power consumption.

III. FPGA-Based Power Monitor

In this section, we discuss how an on-chip power consumption monitor can be built as software-programmed logic on a modern FPGA. We first discuss the operating principles behind on-chip power monitors, and then describe our implementation of one such monitor and characterizes its performance.

A. Operating Principle

The power demanded by a CMOS circuit can be modeled as the sum of the static and dynamic components of power consumption. As power side-channel attacks often leverage data-dependent changes in the power consumption, we only focus on monitoring the dynamic power consumption, P_{dyn}. For one CMOS cell, the average dynamic power consumption can be modeled as the sum of charging and short-circuit power consumption, $P_{dyn} = P_{chrg} + P_{sc}$, where $P_{chrg} = \alpha * f * C_L * V_{DD}^2$ and $P_{sc} = \alpha * f * V_{DD} * I_{peak} * t_{sc}$. α is the activity factor, f is the clock frequency, V_{DD} is the supply voltage, C_L is the load capacitance, I_{peak} is the current peak caused by the switching event, and t_{sc} is the short-circuit time. The equations clearly show that the dynamic power increases proportionally with the activity of the circuit, α.

A power distribution network (PDN) converts and distributes power from the power supply to individual circuit components. The goal of the PDN is to provide a clean voltage supply resistant to varying current demands [17]. To maintain a constant voltage, a PDN uses a voltage regulator to adjust the amount of supplied current and uses decoupling capacitors as a buffer to handle current variations. However, the voltage regulator and the decoupling capacitors cannot completely hide current variations, and high switching activities often lead to transient voltage drops in the PDN of an FPGA [18]. In other words, the voltage drop on the PDN reflects the power consumption.

The PDN can be modeled as an equivalent RLC matrix. Thus, the transient voltage drop seen by a circuit can be approximated by the following equation.

$$V_{drop} = IR + L\frac{di}{dt} \tag{1}$$

The voltage drop depends on both steady-state current consumption (IR drop) as well as short transients ($\frac{di}{dt}$ drop) caused by switching logic on the FPGA. In typical CMOS circuits, combinational logic delays can be modeled to be inversely proportional to the voltage supplied to each gate [18]. Therefore, a change in the combinational logic delay reflects the voltage drop, which again reflects the power consumption and the switching activity of the circuits.

This correlation between combinational logic delay and the power consumption can be leveraged to build an on-chip power monitor on an FPGA. In essence, we can program an FPGA with a circuit that allows us to measure a combinational path delay, and use the delay to estimate the power consumption of other modules that share the PDN. In this work, we propose to use ring oscillators (ROs) to build a voltmeter on an FPGA. Previous studies have used ROs to detect voltage changes due to physical attacks [19] and also showed their oscillation frequency changes when there is a large amount of switching activity on an FPGA [20]–[22]. Here, we show that the ROs can also be used as a general-purpose voltmeter that can distinguish different levels of switching activities.

As an alternative, a previous work has also proposed another voltmeter design on an FPGA based on a delay line coupled with a time-to-digital converter (TDC) [23]. Another paper then demonstrated utilizing the delay line-based monitor to characterize transient voltage fluctuations caused by switching logic in FPGAs [24]. The delay-line based power monitor can also be used for the power side-channel attacks on an FPGA, as we later discuss in Section VI. In this study, we primarily use the RO-based power monitor as it is more portable and easier to implement across a wide range of FPGAs without careful customizations.

B. RO-Based Power Monitor Design

A ring oscillator consists of an odd number of inverters connected in series along with an AND gate such that the output of the last inverter is combinationally fed back into the input of the AND gate. The other input of the AND gate is connected to an enable signal. The oscillation frequency

of the RO is thus inversely proportional to the time that a signal takes to propagate twice around the circuit. In addition to voltage variations, the propagation delay is dependent on process and temperature variations [20]. However, by reducing the number of stages in the RO, one can both reduce the temperature variation dependency [25], as well as increase the resolution of the ring oscillator [21]. The oscillation frequency can then be approximated by Equation 2, where k and f_0 are constants, and $V(x, y, t)$ is the transient supply voltage at the RO's location.

$$f_{RO} \approx k * V(x, y, t) + f_0 \tag{2}$$

The above equation suggests that the RO can be used as a voltmeter if we can measure its frequency. To measure the frequency of the RO, we construct the RO counter circuit shown in 2a. Only one inverter stage is used in order to reduce temperature dependency and improve resolution. The output of the RO circuit is used to clock a counter that increments every oscillation period. As the ring oscillator is oscillating much faster than the system clock, the counter is constructed as a chain of T-flip-flops (TFF) to eliminate slow carry chains. Alongside the TFF counter, a second counter is clocked by a reference clock running with the frequency f_{Ref}. In our implementation, the FPGA system clock is used as the reference clock. Both the RO circuit and the reference counter are enabled at the same time. The RO is allowed to run until the reference counter reaches a pre-determined sampling cycle count C_{Ref}. Then, the RO is disabled and the TFF counter C_{RO} is read. The RO frequency can then be calculated using Equation 3, where $\varepsilon \in [0, \frac{f_{Ref}}{C_{Ref}}]$ is the quantization error introduced by the phase difference between the two clock pulses.

$$f_{RO} = C_{RO} * \frac{f_{Ref}}{C_{Ref}} + \varepsilon \tag{3}$$

There exists an inherent trade-off between time and power resolution of the RO-based power monitor; distinguishing a small difference in power consumption requires running ROs long enough for that difference to show up as a sufficient change in the oscillation count, but at the cost of reduced time resolution. Furthermore, the sensitivity of the monitor may also depend on the spatial proximity of the monitor to switching logic [24].

C. RO-Based Power Monitor Characterization

Previous studies have characterized the performance of ROs on FPGA fabric for non-security purposes [21], [25]–[27] and demonstrated that the RO frequency can detect activities in circuits around the oscillators [20], [21]. Studies have also proposed using ROs as power monitors to detect physical tampering and the presence of embedded hardware Trojans [19], [22]. However, no work has explicitly studied the performance of ROs in the context of power analysis attacks. For example, it is not studied if ROs can be used as a way to detect a varying level of circuit activities due to data dependencies, and if so

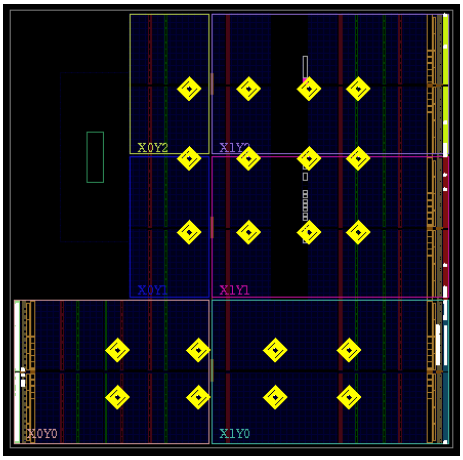


Fig. 3. RO placements on the FPGA fabric indicated by diamond marks.

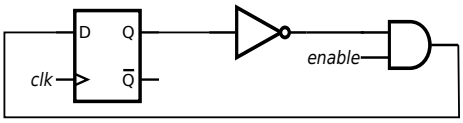


Fig. 4. One instance of the power virus. Nets in the circuit switch with an activity factor of either 1 or 0 depending on *enable*.

what the power and time resolutions can be. For such attacks, it is also important to understand how the RO-based monitor's accuracy changes as a function of spatial proximity to the target logic as today's FPGA partitioning scheme will likely put attack and victim circuits in different parts of an FPGA. Here, we provide detailed experimental results to understand the capabilities and limitations of the RO-based power monitor on a modern FPGA.

Our experimental setup runs on a 28nm Zynq-7020 SoC on a Zedboard, which integrates a hardened dual-core ARM Cortex-A9 with an Artix-7 equivalent FPGA with 53,200 LUTs [28]. The power monitor circuit is instantiated on the FPGA fabric. In order to obtain a higher power resolution with a given sampling period, we create network of 20 RO circuits and take the sum of all RO counters as the final power monitor value, as shown in Figure 2b. In this experiment, the 20 ROs are distributed throughout the FPGA fabric using placement constraints shown in Figure 3 in order to average dependence on spatial proximity to switching logic. We later show that the placement and routing constraints are not necessary for building an RO-based power monitor that is accurate enough for successful power analysis attacks on RSA. The reference clock runs at 100 MHz. Finally, a control program running on the CPU receives the summed power monitor counter value via pipe-like device files provided through Xillybus, which can be accessed via FIFOs on the FPGA [29].

1) Experiment 1: Power Estimate: Our first experiment was to characterize the frequency change of the ring oscillators to a known change in switching activity. To do so, we created a 'power virus' that consists of one flip flop whose output is fed

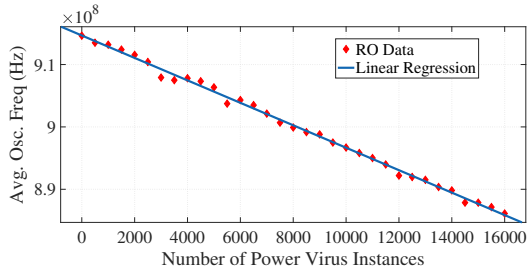


Fig. 5. The average RO frequency versus the power consumption level. A linear regression is shown to fit the data.

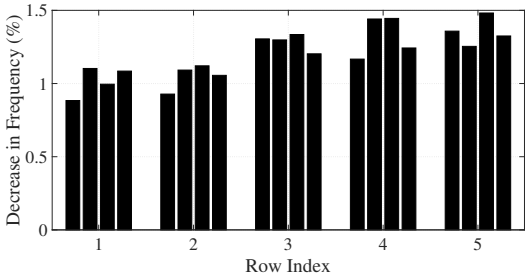


Fig. 6. The RO frequency drop with varying RO locations. Row indices correspond to the rows in Figure 3, increasing from bottom to top (i.e. higher row indices are closer to power viruses). In each row, bars correspond to RO instances from left to right in both the plot and the FPGA.

through an inverter followed by an AND gate and back into its own input port, ensuring that the inverter and the AND gate are instantiated as two separate LUTs as shown in Figure 4. In this manner, the activity factor of each net in the circuit can either be set to 1 or 0 via the enable signal, resulting in the maximal and minimal dynamic power consumption for the given circuit.

We then instantiated 16,000 instances of the power virus on the FPGA fabric so as to cover the majority of the FPGA, minimizing the response's dependency on spatial proximity. We then grouped 500 evenly-distributed instances of the power virus together and tested 33 different power consumption levels by turning on a varying number of the 32 power virus groups. For each power level, we record 1,000,000 RO power monitor samples using the sampling period of 1,001 cycles at 100 MHz. We then take the average of the 1,000,000 samples as our steady-state RO frequency, which minimizes influence due to $\frac{di}{dt}$ transients.

The results from the experiment are shown in Figure 5. The baseline average frequency across all 20 ROs, with no additional switching activity, is 914.7MHz. A linear correlation which very closely models the correlation between switching activity and oscillation frequency can be constructed as $f(x) = -1800x + 9.147 * 10^8$ with an R-squared value of 0.9966, where x is the number of power virus instances actively switching. Using a post-implementation simulation of the power virus circuit via Xilinx Power Estimator and setting the activity factor of all nets to 1 provides a dynamic power consumption of 0.721W for all 16,000 power virus instances. Assuming that each power virus instance consumes the same amount of power, we can compute a linear relationship between the power consumption and the RO frequency.

2) Experiment 2: Spatial Dependence: One common security mechanism to isolate sensitive logic is to use unused CLBs as a 'fence' to isolate the sensitive logic from untrusted logic [12], [13]. However, the fence of unused logic does not prevent power side-channel attacks as the power distribution network is still shared across an FPGA. In this experiment, we characterize the frequency change of individual ROs with respect to their spatial proximity to switching logic that consumes power.

As in Experiment 1, we instantiate a network of 20 ROs placed throughout the FPGA fabric. In this case, we instantiate 6,400 instances of the power virus, and constrain the placement of the power virus to the upper third region (corresponding to clock regions X0Y2 and X1Y2 in Figure 3) of the FPGA via a `pblock` constraint.

For each of the 20 ROs, we read the counter value for 1,000,000 samples using the sampling period of 1,001 cycles at 100MHz with all 6,400 instances of the power viruses off. We then measured the average RO counter value again with all 6,400 power viruses on. To remove process variations among RO instances, we calculated the percent change in the oscillation frequency of each RO with and without the power viruses running. The results are shown in Figure 6.

We observe that for ROs that were placed directly inside the power virus's `pblock` region (i.e. row 5), the oscillation frequency changed by approximately 1.3 to 1.5 percent. ROs placed at the other end of the FPGA fabric (i.e. row 1) showed a frequency change between 0.9 and 1.1 percent, which is smaller compared to closer ring oscillators but still significant. Thus, while the spatial proximity to switching logic does increase the RO sensitivity, physically separating logic by unused CLBs does not eliminate information leakage through the power side channel.

3) Experiment 3: Time Resolution: We now study the time resolution, in terms of sampling frequency, achievable by RO-based power monitors. It is clear that the maximum sampling frequency must be smaller than the RO oscillation frequency. However, the maximum sampling frequency for the RO-based power monitor in practice depends on multiple factors including the necessary precision for power measurements, quantization errors, environmental noise, and others. There is an inherent trade-off between the sampling frequency and the accuracy of the RO-based power monitor; a lower sampling frequency needs to be used for a higher accuracy.

The first source of inaccuracy comes from the inherent quantization error as shown in Equation 3. The phase difference between the reference and RO clocks may affect the RO count by 1. Also, the RO delay change over a sampling period must be larger than one RO oscillation period in order to be detected by an RO count. In that sense, the quantization error is proportional to the inverse of the RO counter value C_{RO}.

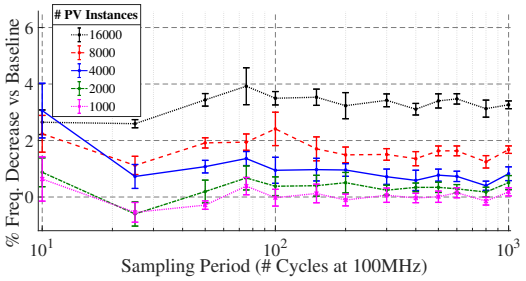


Fig. 7. The percent decrease in the RO frequency and the coefficient of variance as functions of a sampling period. Each line represents a different number of active power virus instances.

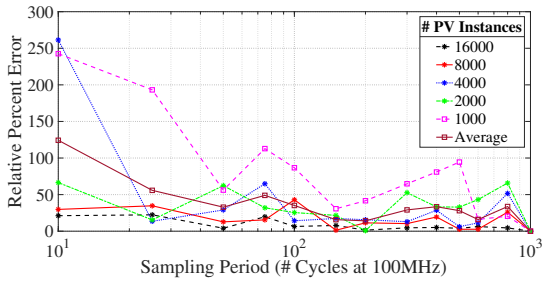


Fig. 8. The percent error in the RO frequency drop compared to the reference case with a 1,000-cycle sampling period.

The quantization error linearly increases with the sampling frequency as C_{RO} increases linearly with the sampling period.

For a given sampling period, C_{RO} increases as the oscillation frequency f_{RO} increases. Therefore, to minimize the quantization error, we should use the fastest RO design. This explains why our power monitor uses ROs with a single inverter loop.

The second source of inaccuracy comes from environmental noise. For example, the oscillation count is affected by thermal noise. A system may also execute other programs, which affects the power consumption. These environmental variations will be better averaged out with a longer sampling period.

Given the trade-off, the sampling frequency that can be used in practice depends on the voltage drop that the power monitor needs to observe, which is again dependent on the power consumed by the target logic as well as the spatial proximity between the RO and the target. The sampling frequency needs to be set so that the RO-based power monitor's accuracy is high enough to be able to detect the target voltage drop.

To experimentally evaluate the effect of sampling frequency on the observed RO oscillation count, we instantiate the same 16,000 power viruses as described in Experiment 1. We then record a 100 millisecond power trace across 15 different sampling periods ranging from 10 cycles (10 MHz) to 1,000 cycles (100 kHz) with the power viruses on and the power viruses off. For each sampling period, we average 10,000 temporally evenly-distributed samples and calculate the percent change in oscillation frequency caused by the power viruses. We perform this analysis with power levels of 16,000, 8,000, 4,000, 2,000, and 1,000 instances of the power virus.

Figure 7 shows the average percent change in the oscillation frequency caused by the power virus for each sampling period compared to the nominal measurement across different levels of switching activity. Additionally, the coefficient of variation ($CV = \frac{\sigma}{\mu}$) is shown for each data point as an error bar. Figure 8 shows the difference in percent frequency drop for each sampling period compared to the 1,000 cycle sampling period for the corresponding switching activity level, which we take as our 'correct' result. We believe that this is a reasonable comparison as with a 1,000 cycle sampling period, the quantization error is negligible and environmental noise is better averaged out.

For long sampling periods, the RO frequency decreases roughly linearly with the number of power virus instances increases, as expected from the results in Experiment 1. However, for short sampling periods, this linear relationship no longer holds. For example, with the sampling period of 10 cycles, the RO frequency decrease is greater when 4,000 power viruses are on compared to when 8,000 power viruses are on. We also observe that relative error tends to increase as the sampling period decreases when comparing the RO frequency change for each sampling period with the reference result with the long period of 1,000 cycles. Similarly, the coefficient of variation (CV) also tend to increase as the sampling period decreases. These results suggest that shorter sampling periods lead to less accurate and consistent results.

The experimental results also show that there is a trade-off between the time resolution and the power consumption that can be reliably detected by the RO-based power monitor. In Figure 7, even relatively long sampling periods cannot consistently detect the increased switching activity compared to the baseline with no activity when only a small number (1,000) of power viruses are used; for some sampling periods, the results even show an *increase* of the oscillation frequency when the power viruses are turned on. However, as the number of power virus instances increases to 2,000, long sampling periods show a consistent decrease in the oscillation frequency compared to the baseline. For high levels of switching activity (e.g. 16,000 power virus instances), even short sampling periods are sufficient in detecting a large decrease in oscillation frequency compared to the baseline. In the context of power analysis attacks, our results show that the optimal sampling period heavily depends on the amount of power consumed by the target device. Reliably detecting a small amount of power consumption requires the attacker to use a long sampling period, while shorter sampling periods may be used when targeting larger or more active circuits.

IV. FPGA-to-FPGA Power Analysis Attack

In this section, we demonstrate one instance of the FPGA-to-FPGA power side-channel attack using the RO-based power monitor. More specifically, we demonstrate a simple power analysis attack on an RSA cryptomodule that is implemented on an FPGA.

```
mod_exp(M, d, N)
{
  R = 1
  S = M
  for (i = 0 to n-1)
  {
    if (d mod 2 == 1)
      R = R*S mod N
    S = S*S mod N
    d = d >> 1
  }
  return R
}
```

Listing 1. Pseudo-code for a square-and-multiply modular exponentiation.

A. Background: RSA Accelerators

The RSA algorithm, first publicly described by Rivest, Shamir, and Adleman in 1978, still remains as one of the most popular public-key cryptographic algorithms in use today [30]. RSA decryption rests on the computation of a large modular exponentiation $M = C^d \ mod \ N$, where C is the ciphertext message, d is the private key exponent, N is the RSA modulus, and M is the decrypted plaintext. The security of RSA is largely based on its key size, which is typically 1,024 to 4,096 bits.

However, computing such a large modular exponentiation is computationally expensive due to the large operands involved. Because the operands are too large for basic data types, performing an RSA decryption on processors is often inefficient, and thus many modern systems use dedicated cryptographic accelerators to speed up the modular exponentiation.

One simple algorithm that many RSA accelerator designs use is *square-and-multiply*, which decomposes the exponent d into the sum of successive powers of 2 (i.e. binary notation). Thus, the decryption can be computed as Equation 4, where n is the length of the decryption exponent d and d_i is bit i of d. This equation can then easily be implemented in both hardware and software by following the algorithm presented in Listing 1.

$$M = \prod_{i=0}^{n-1} \left(C^{2^i}\right)^{d_i} mod \ N \tag{4}$$

B. Hardware Implementation

To study if the RO-based power monitor is accurate and fast enough to perform a power analysis attack in practice, we implemented a 1,024-bit square-and-multiply circuit on the FPGA fabric of a Zedboard. The RSA circuit implements the algorithm as described in Listing 1. The circuit consists of two dedicated modular multiplication modules, as well as a state machine that iterates through each bit of the 1,024-bit input exponent, starting at the least-significant bit. One multiply module is dedicated to compute the square term $S = (S*S) \ mod \ N$, while the other module is used to compute the multiplication $R = (R*S) \ mod \ N$. However, if the current least-significant bit of the iteratively-shifted exponent is 0, the second multiplier instead computes $R = (R*1) \ mod \ N$. Both multipliers are synchronized to start computation at the same cycle for each iteration of the loop.

The multipliers themselves are implemented following a shift-and-add algorithm. Thus, the multiplicand is progressively shifted one bit to the left for each bit of the multiplier. If the corresponding bit of the multiplier is 1, the shifted multiplicand is added to the product P. Furthermore, to ensure that the product is always reduced to $mod \ N$, P, $P - N$, and $P - 2N$ are computed in each step, and the correctly reduced result is used in the next step. The module returns the correct modular multiplication result after all 1,024 bits of the input multiplier value have been iterated through.

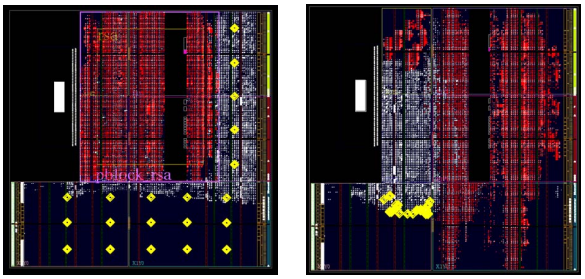
Thus, for a typical decryption, the cryptomodule will require $1,024^2 = 1,048,576$ cycles to complete. Our implementation runs at 20MHz on the relatively low-end and low-speed FPGA fabric of the Zedboard. Thus, the module is capable of computing one modular exponentiation every 52.4ms, or at a throughput of 19.1 ops/sec, with each iteration of the square-and-multiply loop taking 51.2 microseconds. The throughput is comparable to modular exponentiation cores available on the market [31]. In terms of resource utilization, the module requires 15,561 LUTs and 12,843 FFs.

C. Simple Power Analysis (SPA) Attack

In the square-and-multiply circuit, if the current bit of the exponent is 1, then both multipliers will be performing sequences of additions, resulting in high switching activity in the FFs and LUTs that store and compute the multiply's intermediate results. However, if the exponent bit is 0, then only the squaring multiplier's logic will switch, while the other multiplier's logic will be idle. Thus, we expect that the power consumption will be different between an iteration with an exponent bit of 1 and an iteration with an exponent bit of 0.

As a result, the RSA cryptomodule is vulnerable to a Simple Power Analysis (SPA) attack [4]. While SPA has been demonstrated on RSA numerous times since the technique's introduction by Kocher et al. in 1999, recording the requisite power traces has always relied on both a form of physical access to the victim device as well as additional monitoring equipment (e.g. an oscilloscope). In fact, multiple countermeasures against SPA aim to detect the physical modifications made to a device when recording power traces [19], [22], [32]. On the other hand, the RO-based power monitor on an FPGA can be programmed in software and requires neither physical access to the victim device nor additional hardware.

We instantiate the RSA cryptomodule on the Zedboard alongside the RO-based power monitor in various configurations. We consider three separate cases, ranging from least to most restrictive from the attacker's perspective: attacker-defined place and route of the RO circuits (PR), physical isolation between the power monitor and the RSA cryptomodule (ISO), and no user-defined constraints (NoPR). These configurations are shown in Figures 3, 9a, and 9b, respectively. Ten 1,024-bit private keys are generated using OpenSSL. All



(a) Physical Isolation (ISO) (b) No Place and Route (NoPR)

Fig. 9. Spatial placement of RSA module (in red) and RO power monitors (yellow diamonds) in two placement scenarios.

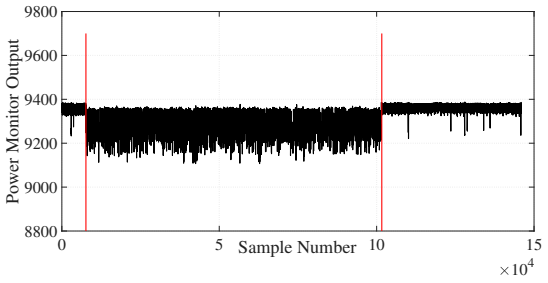


Fig. 10. RSA power trace recorded with the RO-based power monitor. **NOTE:** lower power monitor outputs (i.e. lower oscillation frequencies) correspond to *higher* power consumption.

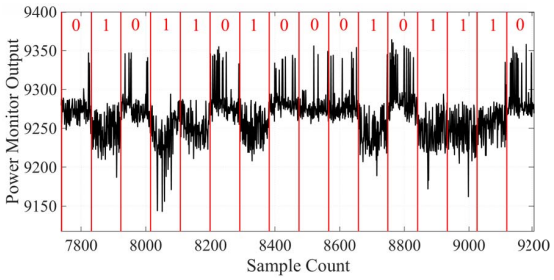


Fig. 11. A zoomed-in view of the RSA power trace showing 16 key bits.

three configurations use the same ten private keys in the RSA cryptomodule. A control program running on the ARM core can then instruct the RSA module to decrypt a message using one of the ten keys by sending a command to the FPGA via Xillybus.

The RO-based power monitor is enabled continuously and its oscillation counts are read out by another process on the ARM core at approximately 2 MHz. The RSA cryptomodule is instructed to perform one modular exponentiation with one of the private keys at a randomly-chosen time. The power monitor trace is exported to a desktop computer and analyzed.

Figure 10 shows a raw power trace corresponding to one full modular exponentiation in the PR configuration. Figure 11 presents a zoomed-in view on a portion of the same power trace. The vertical bars delineate the boundary between successive modular multiplications. From one raw power trace, it is possible to visually identify the difference between one and two switching modular multipliers, which correspond to a key bit of 0 and 1, respectively. It is thus possible to perform a SPA attack to recover the full 1,024-bit decryption key.

We automate the attack with a MATLAB script. As shown in Equation 5, we observe that the recorded power consumption is the sum of the dynamic power consumption of an RSA module, the dynamic power consumption of other activities, the constant static power consumption, and electronic noise, which we consider to be all independent. To identify the section of a power trace that corresponds to the RSA computation (i.e. P_{RSA}), we first record a power trace that only captures background noise (i.e. $P_{other} + P_{static} + P_{noise}$) when the RSA module is idle. We then filter the trace through a low-pass filter to remove high-frequency electronic noise. We can then create a threshold that corresponds to the maximum and minimum background power consumption.

$$P_{total} = P_{RSA} + P_{other} + P_{static} + P_{noise} \tag{5}$$

Next, we perform the same low-pass filtering on a power trace that includes an RSA computation. We can easily identify the portion of the trace corresponding to the RSA computation by observing the first and last samples that go beyond our baseline power consumption threshold. P_{static} is subtracted away from the power trace, leaving us with low-frequency noise and P_{RSA}. Then, we can divide the section of the power trace corresponding to the RSA computation into 1,204 segments, one for each key bit, and simply classify each segment as a 0 or 1 based on the average of the power trace values over the segment. We found that this algorithm is sufficient to recover most of the key bits accurately. The accuracy can be further improved by performing the attack on multiple traces for the same key and taking the majority vote for each bit.

We could successfully recover all ten private keys across the three configurations using the automated algorithm. Table I summarizes the results, showing the average number of incorrectly-guessed bits when only one trace is used and the number of traces to fully recover each key correctly.

In the PR configuration, the attack could successfully recover the vast majority of bits in one trace. The incorrect guesses were often due to short spikes of low-frequency switching noise, likely coming from background activities on a CPU. To remove this noise, we used multiple traces for each key and took the majority vote for each bit. In the PR configuration, only 5 or fewer traces were required to fully recover a key.

[1]Due to the additional noise added at the start and the end of computation caused by large amounts of logic switching on and off, our attack missed the very last bit (MSB) of these keys, but was able to fully recover the remaining bits of the key.

[2]The median, as opposed to the mean, was used as the classification metric in this case. Due to variations in automatic place and route, different metrics yield higher bimodality (and thus easier classification) in power traces.

[3]The standard deviation, as opposed to the mean, was used as the classification metric for the same reasons above.

TABLE I
THE AVERAGE NUMBER OF BIT ERRORS WHEN ONE TRACE IS USED AND THE TOTAL NUMBER OF TRACES TO RECOVER A FULL SECRET KEY UNDER THREE RO CONFIGURATIONS: CONTROLLED PLACE AND ROUTE (PR), PHYSICAL ISOLATION (ISO), AND NO CONSTRAINTS (NOPR).

Key ID	PR		ISO		NoPR	
	Error	# Traces	Error	# Traces	Error	# Traces
1	3	3	81	27	30	6
2	3	3	178	27	33	6[1]
3	49	3	23	5	30	5
4	7	3	37	8	6	3
5	16	3[1]	44	3	67	18
6	40	4	32	5	76	22
7	49	5	5	3	43	8
8	2	5	30	5	148	22[1,2]
9	2	3	1	3	91	17[3]
10	18	5	1	3	49	7
Average	18.9	3.7	43.2	8.9	57.3	11.4

The ISO configuration represents the case when sensitive logic is separated from potentially malicious logic using a 'fence' of CLBs as proposed by Huffmire et al. [13]. In the same manner, we physically separated the power monitor from the RSA cryptomodule using the `pblock` constraints. Table I shows that the physical separation increases the average number of bit errors compared to the PR configuration where the ring oscillators are distributed across the FPGA. This increase is expected as the sensitivity of the RO-based power monitor decreases with spatial separation, as demonstrated in Section III. However, all ten keys could still be recovered correctly only using 8.9 traces on average, showing that physical separation is not sufficient as a countermeasure.

In the NoPR configuration, we implement our attack logic including the power monitor solely in RTL code without using any design constraints. In this case, the ring oscillators are arbitrarily placed by a design tool. The frequencies of individual ring oscillators can also vary significantly as there is no placement or routing constraint. Surprisingly, we found that our attack can still recover all ten keys successfully only with relatively minor increases in the number of traces compared to the ISO configuration. This result shows that restricting user interfaces to disallow low-level design constraints cannot fully prevent the FPGA-based power analysis attack.

We believe that the RO-based power monitor still works well even without controlled placement and routing for two reasons: (1) power analysis attacks rely on *relative* changes in the power consumption and RO counts, which makes variations in individual ring oscillators unimportant, and (2) the FPGA tools tend to place the LUTs belonging to one RO closely, often in the same CLB, with consistent routing - so the variations among individual ROs are often not significant.

D. Effects of Noise on SPA Attack

Our attacks demonstrate that an entire RSA key can be recovered correctly with a small number of power traces when the victim system is mainly running a single RSA computation at a time. The power traces in our previous experiments include the noise from other components on an FPGA such as the Xillybus IP or attack logic as well as the Linux operating system running on the CPU. The noise showed up as periodic spikes in the power consumption. However, the magnitude and the duration of these spikes were low enough to be removed relatively simply using multiple power traces.

Here, we study the impact of more significant noise from co-resident circuits on an FPGA or software on a CPU on the effectiveness of the FPGA-based power analysis attacks. For this purpose, we consider the case where a co-resident FPGA circuit consumes a large amount of dynamic power, using the power virus in Figure 4 to add significant dynamic power consumption while an RSA computation is running. In this experiment, we instantiate 8,192 instances of the power virus next to the RSA cryptomodule on an FPGA. The power viruses use 8,192 FFs and 16,384 LUTs. We also implement the power monitor circuit with 20 ROs without any place and route constraints (NoPR) on the FPGA. Compared to the baseline case with no FPGA activity, enabling the power viruses results in a 1.27% drop in the RO frequency observed by the power monitor. Enabling the RSA module, with both multipliers switching, alongside the power viruses results in a 2.54% drop in the RO frequency. Because the power virus has a high activity factor, the combined power consumption of the 8,192 power virus instances is comparable to the power consumption of both multipliers in the RSA module even though the power viruses use less FPGA resources.

We first evaluate the impact of a large, but constant noise on the effectiveness of our attack. Equation 5 suggests that the additional power consumption will simply be superimposed onto the RSA power signal. In other words, the oscillation frequency of the ROs will simply be shifted down by a constant factor, and our automated attack should not be impeded when a constant noise is added to the power trace. We experimentally tested this case by constantly enabling all 8,192 power virus instances and recording the power traces of the RSA cryptomodule running with Key 1. The automated SPA attack resulted in 32 bit errors on average with a single trace and could fully recover the entire RSA key correctly using 5 traces. The result is comparable to the NoPR case without the power virus (see Table I), showing that the constant noise does not noticeably affect the power analysis attack.

In practical systems, the power consumption of an FPGA or CPU task that adds noise is likely to change dynamically. We model this case by dividing the 8,192 power virus instances into 32 groups and dynamically changing the number of power virus groups enabled at run-time. There are 32 power levels, where each level corresponds to the power consumption from additional 256 power virus instances. In our experiment, we had a control process on the CPU randomly set the power level to a value between 0 and 15 every 30 RO sampling periods. The RO sampling frequency is 2MHz, so the number of active power virus instances changes roughly at 67KHz with variations from non-determinism in CPU load and CPU-FPGA communication latency. We use 16 power levels, which corresponds to a maximum of 4,096 enabled power virus instances, because the power consumption of 4,096 power viruses is

comparable to the power consumption of one multiplier unit in the RSA module, which our power analysis attack aims to detect in the power trace. The power trace is recorded using the RO-based power meter in the NoPR configuration when both the RSA module and the power viruses are active.

To deal with the significant random noise that changes constantly, we modify our automated attack script to first compute the average RO frequency over multiple power traces, and then use the average to guess if each bit of an RSA key is 0 or 1. The modified attack script could fully recover the correct RSA key using 31 traces, compared to 5 traces under constant noise. While the randomly changing noise increased the number of traces required for a successful attack by approximately six times, the experiment shows that random noise can also be filtered out by using more traces.

When the magnitude of the random noise is increased even more to use all 8,192 power viruses, we found that the added noise introduced difficulty in automatically identifying the start of the RSA decryption. Our script identifies a section in the trace with a large increase in power consumption as an RSA computation. Having a large number of power viruses randomly enabled makes this identification process difficult because many large spikes in power consumption appear in the observed power trace. However, we believe that this scenario represents an unlikely worst case in practice as applications will often introduce bursty noise rather than adding large power swings that change rapidly over a very long period of time.

V. FPGA-to-CPU Attack

Modern SoCs integrate many discrete components, such as CPUs, DSPs, GPUs, crypto accelerators, and others into one integrated circuit. SoCs offer a number of benefits over traditional discrete devices. As they integrate multiple specialized accelerators over high bandwidth on-chip networks, SoCs enable smaller and more energy efficient systems. One downside of traditional SoCs lies in the fact that developing multiple configurations of SoCs to suit various applications incurs high non-recurrent engineering (NRE) costs. The solution adopted by both Intel and Xilinx is to integrate reconfigurable FPGA fabric into an SoC.

In SoCs, it is common for modules on the same die to share the same power supply. For example, in an SoC with CPUs and FPGA fabric, both CPU and FPGA power supply rails are often shared. This is the case for the Xilinx Zynq SoC on the Zedboard that we use for experiments [33]. In this case, it is reasonable to assume that switching activities from the CPU or other modules on an SoC will cause a voltage drop in the FPGA logic. In this section, we demonstrate that this is indeed the case for the Zedboard by using our RO-based power monitor. We leverage this FPGA-to-CPU power side-channel to defeat a traditional timing-channel protection scheme that delays outputs.

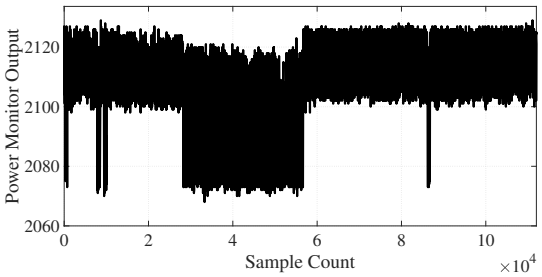


Fig. 12. A power monitor trace from a CPU execution of `strcmp`, which is shown by the increased power consumption between samples 23,000 and 59,000.

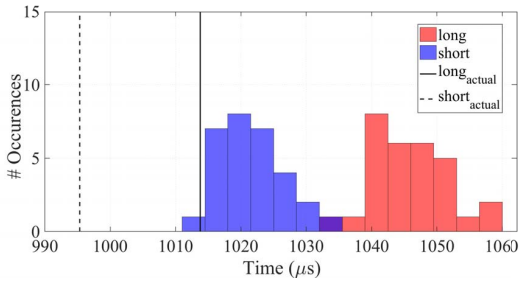


Fig. 13. A histogram of the estimated execution times from the power traces of long and short program executions. The lines represent the actual execution times from `clock_gettime`.

A. Measuring CPU Activity from the FPGA

We implement the RO-based power monitor on FPGA fabric on the Zynq SoC with the placement shown in Figure 3. To minimize the noise from other activities outside the CPU, we do not instantiate any logic on the FPGA except for the RO-based power monitor and the control logic (e.g. state machines, Xillybus IP, FIFOs, etc.) necessary to communicate with software on the ARM CPU.

On the dual-core ARM CPU, we run the Xillinux operating system which is based upon Ubuntu LTS 12.04 for ARM [29]. We begin by implementing a simple program which defines an 8,000-character string and uses `strcmp` to compare the string to itself. Figure 12 shows one power trace recorded using a 13-cycle sampling period while running the program. The program execution is clearly visible approximately between samples 23,000 and 59,000.

In fact, the power trace shows more than just the `strcmp` function executing. Using a system timer `clock_gettime`, we find that the `strcmp` function call takes approximately 300 microseconds to execute. However, from our trace, we observe that the CPU is busy for around 4 milliseconds, suggesting that the power trace includes additional activities in our test program such as program setup code, memory allocation, copying text, etc. The results suggest that we can observe both user-level program execution and privileged code execution in the operating system.

To evaluate the accuracy of timing measurements through the FPGA-based power trace, we also experimented with long

and short strings as the inputs to the `strcmp` function. The 'long' function call took 1,013.769 μs to execute and the 'short' function call took 995.337 μs when measured with `clock_gettime`. In this experiment, we insert a delay before and after the call to `strcmp`, which creates periods of low CPU activity. The delay allows us to differentiate the `strcmp` function call from the rest of the program execution. We recorded 30 power traces for the 'long' function call 30 traces for the 'short' function call.

Figure 13 shows the histogram of the estimated execution times from the power traces. The histogram shows a clear shift in the execution time between the 'long' function call and the 'short' function call. In fact, the execution time difference observed from the power traces is quite close to the difference of 18.4 μs from `clock_gettime`. The standard deviation is 5.60 μs and 5.42 μs for the 'long' case and the 'short' case, respectively. The result suggests that the FPGA power monitor is able to distinguish changes in program execution time in the order of a few microseconds, which we believe is accurate enough to enable common timing side-channel attacks on CPU processes [34].

However, the absolute execution time estimated from a power trace does not completely agree with the system clock time. In the 'long' case, the estimated execution time is 32.5 μs longer than the system clock time on average. In the 'short' case, the difference is 26.4 μs. Since the execution time from the power trace is consistently longer than the system clock time, we attribute the discrepancy to additional computations related to system calls. While there is a discrepancy in the absolute execution time, most timing side-channel attacks are based on the relative time difference in program execution.

B. Attack against Timing-Channel Mitigation Techniques

In addition to traditional power side-channel attacks, an adversary can use the capability to measure the internal execution time of an application running on a CPU to perform timing channel attacks. Here, we briefly discuss the intuition of the timing attack through power side channels.

Timing channel attacks infer confidential information by observing data-dependent timing variations from sources such as caches [35], program execution [36], and network latencies [34], [37]. Compared to power analysis attacks, these timing attacks are easier to perform remotely as the timing of external behaviors such as inputs and outputs may be observed without direct physical access to the system.

For remote timing channel attacks, a common threat model assumes that only external behaviors can be observed. As a result, the timing channel can be eliminated if observable results of a computation are forced to return only in constant time intervals corresponding to the worst-case execution time. For example, quantization (i.e. mitigating timing channels by quantizing computations into constant-time chunks) was proposed as a countermeasure against Brumley and Boneh's remote timing attack on RSA [34]. Similarly, multiple protection methods have been proposed that mitigate timing channels by ensuring that computations are observably constant-time [38]–[41]. At a high level, such countermeasures wrap computations in a timing mitigator that delays the return time of the computation, reducing or removing the data-dependent timing channel.

We note that the FPGA-based power side channel breaks the basic assumption in the above timing-channel protection schemes. As demonstrated by Figure 12, the execution time of a program on a CPU can be remotely and fairly accurately measured through the power side channel via the power monitor on an FPGA. Thus, even if quantization-based countermeasures eliminate externally observable timing variations, the actual computation time can be obtained from a power trace if the delays inserted by the timing mitigator result in a different power consumption level compared to the sensitive computation. In fact, timing mitigators typically insert delays by simply stalling, which usually consumes less power compared to the actual computation. Therefore, today's delay-based timing channel protection schemes are likely to be vulnerable against power side-channel attacks.

C. Example Attack on an RSA

Here, we present a concrete attack example that shows how the FPGA power side channel can be used to break the delay-based timing mitigation techniques. We again consider a square-and-multiply based implementation of RSA's modular exponentiation, as presented in Listing 1. In this case, we assume a threat model reflected in Figure 1b. A user or system process implements a constant-time version of the modular exponentiation by ensuring that each iteration of the loop executes in constant time. He or she does so by adding a delay function such that each loop iteration executes at the worst-case execution time. This delay-based scheme provides more efficient and portable protection compared to the always-multiply scheme, which always performs the multiply and discards the result if unnecessary. The always-multiply scheme consumes energy to perform unnecessary computations and needs to ensure that a compiler does not optimize away the multiply operations whose results are discarded.

The modular exponentiation is implemented on a CPU using the square-and-multiply algorithm to operate on a 4,096-bit key, with values being represented using the GNU Multiple Precision library (`gmp`). The time for each iteration of the loop requires approximately 460 microseconds if the corresponding bit is set to 1 and 220 microseconds if the bit is 0. Thus, we inserted a delay function to each iteration to ensures that each iteration of the RSA loop (and thus the entire computation) takes the longer execution time.

In this attack, we assume that an attack program is allowed to run on a CPU and access its logic instantiated on the FPGA. To perform a timing attack on the victim program (RSA), the attack program constantly records the power monitor readings from the FPGA, using the sampling period of 50 cycles. While the power monitor is recording, the RSA modular exponentiation starts execution as a separate process on the CPU. Figure 14 shows a zoomed-in view of the power trace on 32 iterations of the square-and-multiply loop. The first 16

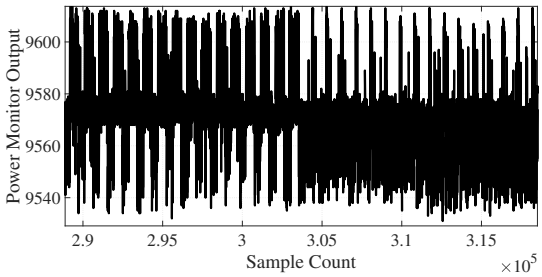


Fig. 14. A zoomed-in view of the power trace for constant-time modular exponentiation in software.

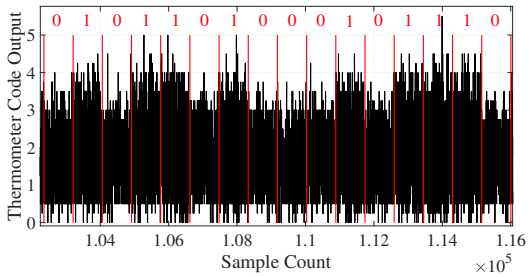


Fig. 15. A zoomed-in view of the RSA power trace captured using the delay-line circuit for the same key bits shown in Figure 11.

bits (16 LSBs) of the secret key are all set to 0, while the next 16 bits are all set to 1. The power trace clearly shows that the first half of the trace, corresponding to 16 bits of 0, has periods where the power consumption is significantly lower (i.e. the power monitor output is higher) than the latter half of the power trace. These periods correspond to the delays inserted for timing channel protection. On the other hand, periods with low power consumption are much shorter in the second half of the trace, suggesting that there no significant idle time. Thus, an attacker can simply identify the corresponding key bit for each iteration by looking at the trace even if the execution time of the program is fixed.

VI. Discussion

The very nature of FPGAs provides users with fine-grained control over reconfigurable fabric. As heterogeneous FPGA-based systems become more widely deployed in systems ranging from cloud datacenters to mobile platforms, the security implication of allowing untrusted parties to remotely implement custom circuits on an FPGA becomes more significant. We demonstrate that the FPGA introduces a new security vulnerability by allowing untrusted users to remotely implement a power monitor that is capable of inferring the level of dynamic switching activities. This power monitor circuit can not only measure the power consumption of circuits on the reconfigurable fabric, but also observe the power consumption of other modules such as a CPU (or a GPU) on a heterogeneous SoC device. While we explored a few possible attacks enabled by these circuits, we believe that this capability can be exploited in many potential security attacks as discussed in Section II-C.

A. Alternative On-Chip Power Monitors

While we use ROs as the circuit in our power monitor due to their simplicity and ease-of-implementation, there exist other circuits that are sensitive to propagation delays. The other prominent circuit is the delay line coupled with a TDC, as discussed in Section III. Zick et al. originally proposed a delay-line based voltage monitor with the intent to detect voltage-attacks on FPGAs [23]. Gnad et al. further extended the delay line TDC to characterize transient voltages on FPGAs [24], showing that switching activities on an FPGA can cause a visible signal in a delay line TDC.

At a high level, delay-line monitors operate based on the same principle as RO-based monitors. However, instead of measuring propagation delay via the oscillation frequency of a RO, delay lines use a long chain of logic elements connected in series. A signal (generally a clock signal) is passed through the chain. After a fixed time period, the logical value of all of the nets between each sequentially-connected logic element is latched. The propagation delay, and thus transient voltage, can be measured by observing how far along the chain the signal propagates within the fixed period. In other words, the power consumption can be estimated by observing where the boundary between all 1 values and all 0 values is located as in a thermometer code.

To understand the trade-offs between the RO-based power monitor and the delay-line based power monitor, we implemented a delay-line power monitor and tested its performance. The logic elements in our power monitor consists of a chain of CARRY4 primitive logic cells intended for use in carry-lookahead adders [42]. This is so as to optimize the speed of signal propagation between logic elements, increasing the power resolution of our monitor. Each CARRY4 logic cell contains four chained MUXCYs, which we use as our buffers. We thus connect together a series of 32 CARRY4 logic cells to obtain a chain of 128 buffers. A delayed clock signal (generated from passing the clock through an initial series of open latches) is fed into the delay line. At every clock cycle, the net between each buffer is latched and read as our power monitor reading. In this implementation, we can achieve a sampling frequency of 100MHz on the Zedboard.

1) Attack on RSA: We repeated the same simple power analysis (SPA) attack on the RSA modular exponentiation circuit in Section IV by placing a delay line power monitor on the FPGA fabric[4]. Using the delay line, we recorded a power trace of a modular exponentiation. One representative trace is shown in Figure 15. The results show that the remote SPA attack is also possible with the delay-line based power monitor.

2) Trade-off between ROs and Delay-Lines: While we demonstrate that both ROs and delay lines can be used to

[4]For the experiment, we did not physically isolate the power monitor via fences as our earlier experiments showed that physical isolation is ineffective against power side-channel attacks.

enable remote power side-channel attacks, there are certain trade-offs that exist between the two designs.

Achievable Sampling Frequency. As discussed and evaluated in Section III, there exists a trade-off between the RO-based power monitor's measurement accuracy and the sampling frequency that it can achieve. With short sampling periods, quantization noise begins to significantly decrease the measurement accuracy of the RO power monitor. In contrast, the delay-line monitor can achieve sampling frequencies up to the clock frequency supplied to the FPGA fabric without any change in measurement accuracy. However, the resolution of the delay line is fixed and cannot be dynamically adjusted.

Achievable Power Resolution. For simple power analysis attacks on RSA, both power monitor designs provide sufficient resolution to observe the changes in power consumption. In the RO-based power monitor, the resolution can be improved to some extent by increasing the sampling period. Over a long period, even a small change in the propagation delay will lead to a difference in the oscillation count. In that sense, the RO count can be expressed as a linear function of the level of power consumption as discussed Section III. In the delay line, the smallest delay variation that can be detected is determined by the propagation delay between two adjacent buffers in the delay line, which is fixed. However, it is difficult to state which design is more sensitive to voltage transients, as the quantization error of a power monitor depends on its implementation. A RO circuit with many inverters may yield lower power resolution than a delay line with small wire delays in-between buffers, while a poorly-routed delay line may yield lower power resolution than a small RO circuit.

Robustness of Design and Implementation. One major weakness of the delay-line power monitor lies in the complexity of the implementation and the sensitivity of the measurement to the exact placement and routing of the delay line on an FPGA. In order for the delay line to work effectively, the logic cells used for buffers and the wire lengths between buffers must be the same in order to provide equivalent propagation delays between each successive latch. Furthermore, the clock skew to each latch along the delay line must be minimized to ensure that all latches sample at the same time. In our design, achieving such precise timing constraints requires the exact placement and routing of each logic cell. Moreover, such careful customization is likely necessary for each FPGA where a delay line is implemented. On the other hand, the RO circuit only requires simply two LUTs in a loop and can still be used even if all ROs are not identical. As a result, the RO-based power monitor is much easier to implement and port to different FPGAs. An adversary can implement the RO-based power monitor even when placement and routing constraints cannot be used.

B. Limitations of FPGA-Based Power Side Channel

The cost of remotely measuring power consumption via on-chip circuits is resolution. The FPGA-based power monitor circuits cannot match the micro-volt and pico-second level accuracy of high-resolution oscilloscopes utilized in traditional power analysis attacks. It is not clear if FPGA-based power monitors are accurate enough for attacks that require very fine-grained resolution. Furthermore, while we observe that the power consumption of a CPU is observable in an SoC context, we cannot assume the same for systems utilizing discrete CPUs and FPGAs. We believe that our work represents a starting point. Further studies will be needed to fully understand the capabilities and the limitations of the FPGA-based power side channel.

C. Countermeasures

Potential countermeasures to our attacks fit into one of two categories: making the victim logic or process more resilient to power side-channel attacks or making it more difficult for untrusted users to construct power monitoring circuits on an FPGA. While we believe that effective countermeasures can be deployed for specific attack circuits and examples in this paper, general protection against FPGA-based power side-channel attacks needs further studies.

Countermeasures against traditional power analysis attacks have been well-studied [43]. Multiple methods exist to either hide the power signal by inserting random noise or adding dummy operations or masking power consumption by randomizing intermediate values. These countermeasures can also be applied to circuits on an FPGA or software on a heterogeneous SoC to make them more resilient against the FPGA-based power side channels. However, these countermeasures often come with overhead in performance and energy [44], which physically-secure systems traditionally did not have to pay. Some hardware countermeasures such as using dynamic logic also cannot be implemented on typical FPGA devices.

An alternative countermeasure is for the system administrator, such as a cloud-FPGA provider, to check an FPGA design before placing the logic onto a physical FPGA, and disallow a malicious design with a FPGA-based power monitor. For example, the RO-based power monitor may be detected by analyzing a netlist for combinational loops. Preventing users from specifically defining place-and-route constraints may also be used to reduce the effectiveness of delay-line power monitors. However, there also exist legitimate uses of both combinational loops (e.g. ring-oscillator-based physically unclonable functions that generate unique secret keys [45]) and user-defined place-and-route for design optimizations (e.g. optimizing timing constraints for large carry chains). Performing detailed analyses is also time-consuming on netlists and near-impossible for encrypted bitstreams. Furthermore, attackers may develop more clever circuits that can remotely measure power consumption. Therefore, reliably detecting malicious circuits while allowing legitimate use cases seems non-trivial.

VII. Related Work

The power side-channel analysis attack is a widely studied topic since the early work by Kocher et al. [4]. Many power analysis attacks and countermeasures have appeared in literature [44]. We use previously proposed power analysis methods to demonstrate attacks on FPGA-based systems. However, the

traditional power analysis attacks assume physical access or proximity to the victim system to measure power consumption. In this paper, we demonstrate that the power analysis attack on confidential data can be performed remotely in software (i.e. without physical proximity) by programming an FPGA to implement a power monitor circuit.

Concurrent studies to this work also demonstrate remote side-channel attacks using an FPGA. Schellenberg et al. [46] use a delay-line power monitor to perform a differential power analysis attack (DPA) attack on a co-located AES module. Ramesh et al. [47] manually route a long wire adjacent to a victim AES module and use crosstalk to perform a DPA-like attack. In contrast, our study shows attacks on RSA using the RO-based power monitor, which can be built without placement and routing constraints, in addition to the delay-line monitor. We also provide a detailed characterization study for the RO-based monitor to understand its capabilities and limitations. In addition to the FPGA-to-FPGA attack, this paper shows that an FPGA-to-CPU attack is also viable on an SoC platform.

Alternative side-channels have been used to attack electronic devices without directly probing or modifying the system. Attackers can leverage electromagnetic, acoustic, or optical emanations to steal confidential information from various devices [48]–[51]. While these attacks do not require direct physical access to a target, they do require physical *proximity*. The remote attack in this paper can be performed in software over the network even without physical proximity.

Ring oscillators on an FPGA have been studied as a way to measure process, voltage, and temperature (PVT) variations for non-security applications. For example, Ruething et al. [27] and Franco et al. [25] characterize the temperature-dependency of ROs in FPGAs while Yu et al. [26] examine process-variation dependencies. Zick and Hayes [21] present a RO-based circuit with the goal of quickly characterizing PVT variations across devices. Finally, Barbareschi et al. [20] characterize how design parameters, such as the number of stages, affect RO frequencies.

The ROs on an FPGA have also been used in security applications, but not for power analysis attacks. For example, Physical Unclonable Functions (PUFs) can be implemented using ROs to turn process variations into a unique fingerprint or secret for each chip [52]. Masle and Luk [19] proposed using ROs to detect a power measurement attack by monitoring a voltage variation triggered by a power measurement circuit inserted in the FPGA's power rail. Hoque [22] demonstrated using ROs as power monitors in the context of identifying hardware Trojans. Sun et al. [53] demonstrate that ROs can act as FPGA-based covert-channel receivers by measuring temperature-dependent frequency variations. In this paper, we use and characterize the ROs on an FPGA in the context of power side-channel attacks.

This paper also demonstrates that the delay line can be used to build a power monitor on an FPGA and perform a simple power analysis attack. Using delay lines as a way to measure voltage transients was first proposed by Zick et al. [23] and further characterized by Gnad et al. [24]. However, these previous studies have not investigated the delay line in the context of power side-channel attacks.

As FPGAs are recently adopted in cloud computing infrastructures, there is only a small body of work that specifically studies remote attacks on cloud-based FPGAs. Gnad et al. [15] demonstrate a remote availability attack on an FPGA by inducing voltage emergencies by performing a large number of synchronized switching activities on the FPGA. Their attack aims to introduce faults in BRAM cells, resulting in a denial-of-service attack. In contrast, this paper shows remote power side-channel attacks on confidentiality.

VIII. Conclusion

This work challenges the long-standing assumption that power analysis attacks require physical proximity to a system and cannot be performed in software remotely. We show that an on-chip power monitor can be implemented by programming a modern FPGA in software and that it can measure the power consumption of circuits on the FPGA and programs running on the same SoC. In particular, we introduce a ring oscillator based power monitor and provide detailed characterization of its performance. We demonstrate that the RO-based power monitor can be implemented without place and route constraints and used to successfully perform a power analysis attack on an RSA cryptomodule even when the power monitor and the RSA modules are physically separated to different regions of the FPGA. Additionally, the power monitor can observe the power consumption of programs on a CPU and be used for attacks against a timing-channel mitigation countermeasure. While this work introduces a new vulnerability through FPGAs, the full capacity and limitations of the FPGA-based power side channel as well as effective countermeasures need to be further investigated in future work.

References

[1] "Amazon EC2 F1," https://aws.amazon.com/ec2/instance-types/f1/, Amazon.com, Inc, accessed: 2017-10-28.

[2] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, D. Firestone, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Y. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, "Configurable clouds," *IEEE Micro*, vol. 37, no. 3, pp. 52–61, 2017.

[3] J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, and S. Jiang, "SDA: Software-defined accelerator for large-scale DNN systems," in *2014 IEEE Hot Chips 26 Symposium (HCS)*, 2014, pp. 1–23.

[4] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *19th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, 1999, pp. 388–397.

[5] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling FPGAs in the cloud," in *11th ACM Conference on Computing Frontiers (CF)*, 2014, pp. 3:1–3:10.

[6] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow, "FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack," in *22nd Annual International Symposium on Field-Programmable Custom Computing Machines*, 2014, pp. 109–116.

[7] O. Knodel, P. Lehmann, and R. G. Spallek, "RC3E: Reconfigurable accelerators in data centres and their provision by adapted service models," in *9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 19–26.

[8] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA accelerators for efficient cloud computing," in *7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2015, pp. 430–435.

[9] S. Ma, D. Andrews, S. Gao, and J. Cummins, "Breeze computing: A just in time (JIT) approach for virtualizing FPGAs in the cloud," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2016, pp. 1–6.

[10] S. Yazdanshenas and V. Betz, "Quantifying and mitigating the costs of FPGA virtualization," in *27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–7.

[11] S. Trimberger and S. McNeil, "Security of FPGAs in data centers," in *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, 2017, pp. 117–122.

[12] E. Hallett, "Xilinx isolation design flow." [Online]. Available: https://www.xilinx.com/applications/isolation-design-flow.html

[13] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems," in *2007 IEEE Symposium on Security and Privacy (S&P)*, 2007, pp. 281–295.

[14] C. Delimitrou and C. Kozyrakis, "Bolt: I know what you did last summer... in the cloud," in *22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017, pp. 599–613.

[15] D. R. E. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–7.

[16] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *6th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2008, pp. 239–252.

[17] "7 series FPGAs PCB design guide," Xilinx, Inc., 2017. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug483_7Series_PCB.pdf

[18] S. Pant, "Design and analysis of power distribution networks in VLSI circuits," Ph.D. dissertation, The University of Michigan, 2008.

[19] A. L. Masle and W. Luk, "Detecting power attacks on reconfigurable hardware," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 14–19.

[20] M. Barbareschi, G. Di Natale, and L. Torres, *Implementation and Analysis of Ring Oscillator Circuits on Xilinx FPGAs*. Springer International Publishing, 2017, pp. 237–251.

[21] K. M. Zick and J. P. Hayes, "Low-cost sensing with ring oscillator arrays for healthier reconfigurable systems," *ACM Transactions on Reconfigurable Technology Systems (TRETS)*, vol. 5, no. 1, pp. 1:1–1:26, Mar. 2012.

[22] T. Hoque, "Ring oscillator based hardware trojan detection," Master's thesis, University of Toledo, Toledo, Ohio, USA, 2015.

[23] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *2013 ACM International Symposium on Field Programmable Gate Arrays (FPGA)*, 2013, pp. 101–104.

[24] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of transient voltage fluctuations in FPGAs," in *2016 International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 12–19.

[25] J. J. L. Franco, E. Boemo, E. Castillo, and L. Parrilla, "Ring oscillators as thermal sensors in FPGAs: Experiments in low voltage," in *2010 VI Southern Programmable Logic Conference (SPL)*, 2010, pp. 133–137.

[26] H. Yu, Q. Xu, and P. H. W. Leong, "Fine-grained characterization of process variation in FPGAs," in *2010 International Conference on Field-Programmable Technology (FPT)*, 2010, pp. 138–145.

[27] C. Ruething, A. Agne, M. Happe, and C. Plessl, "Exploration of ring oscillator design space for temperature measurements on FPGAs," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 559–562.

[28] "Zedboard hardware user's guide," Avnet, Inc., 2014. [Online]. Available: http://zedboard.org/sites/default/files/ documentations/ZedBoard_HW_UG_v2_2.pdf

[29] "Xillybus product brief," Xillybus, Ltd., 2017. [Online]. Available: http://xillybus.com/downloads/xillybus_product_brief.pdf

[30] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[31] "RSA and modular exponentiation cores," Helion Technology Limited, 2017. [Online]. Available: http://www.heliontech.com/modexp.htm

[32] S. Trimberger, "Detection of power analysis attacks," Nov. 18 2014, United States Patent 8,892,903. [Online]. Available: https://www.google.com/patents/US8892903

[33] "Zedboard power distribution and decoupling system," Avnet, Inc., 2012. [Online]. Available: http://zedboard.org/sites/default/files/documentations/Zedboard%20Decoupling%20121001.pdf

[34] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *12th Conference on USENIX Security Symposium - Volume 12 (SSYM)*, 2003, pp. 1–1.

[35] D. J. Bernstein, "Cache-timing attacks on AES," The University of Illinois at Chicago, Tech. Rep., 2005.

[36] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *16th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, 1996, pp. 104–113.

[37] A. Bortz and D. Boneh, "Exposing private information by timing web applications," in *16th International Conference on World Wide Web (WWW)*, 2007, pp. 621–628.

[38] B. Kopf and M. Durmuth, "A provably secure and efficient countermeasure against timing attacks," in *22nd IEEE Computer Security Foundations Symposium*, 2009, pp. 324–335.

[39] D. Zhang, A. Askarov, and A. C. Myers, "Language-based control and mitigation of timing channels," in *33rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2012, pp. 99–110.

[40] A. Askarov, D. Zhang, and A. C. Myers, "Predictive black-box mitigation of timing channels," in *17th ACM Conference on Computer and Communications Security (CCS)*, 2010, pp. 297–307.

[41] D. Zhang, A. Askarov, and A. C. Myers, "Predictive mitigation of timing channels in interactive systems," in *18th ACM Conference on Computer and Communications Security (CCS)*, 2011, pp. 563–574.

[42] "Vivado design suite 7 series FPGA libraries guide," Xilinx, Inc., 2012. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug953-vivado-7series-libraries.pdf

[43] T. Popp, S. Mangard, and E. Oswald, "Power analysis attacks and countermeasures," *IEEE Design Test of Computers*, vol. 24, no. 6, pp. 535–543, 2007.

[44] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., 2007.

[45] X. Xin, J. P. Kaps, and K. Gaj, "A configurable ring-oscillator-based PUF for Xilinx FPGAs," in *14th Euromicro Conference on Digital System Design (DSD)*, 2011, pp. 651–657.

[46] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *2018 Design, Automation, and Test in Europe Conference and Exhibition (DATE)*, 2018.

[47] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, "FPGA side channel attacks without physical access," in *26th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018.

[48] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2002, pp. 29–45.

[49] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): Measures and counter-measures for smart cards," in *2001 International Conference on Research in Smart Cards: Smart Card Programming and Security (E-SMART)*, 2001, pp. 200–210.

[50] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *J. Cryptol.*, vol. 30, no. 2, pp. 392–443, Apr. 2017.

[51] S. Tajik, H. Lohrke, J.-P. Seifert, and C. Boit, "On the power of optical contactless probing: Attacking bitstream encryption of FPGAs," in *2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1661–1674.

[52] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *44th ACM/IEEE Design Automation Conference (DAC)*, 2007, pp. 9–14.

[53] J. Sun, R. Bittner, and K. Eguro, "FPGA side-channel receivers," in *19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2011, pp. 267–276.