Extending FPGA Information Leaks with Trojan Phantom Circuits

Anthony Etim
Yale University
New Haven, CT, USA
anthony.etim@yale.edu

Shanquan Tian
Yale University
New Haven, CT, USA
shanquan.tian@yale.edu

Jakub Szefer
Yale University
New Haven, CT, USA
jakub.szefer@yale.edu

***Abstract*—Field-Programmable Gate Arrays (FPGAs) are increasingly used in data centers and in cloud computing for acceleration of various applications. However, cloud-based FPGAs could be programmed with malicious circuits to leak information. For example, existing work has shown that long-wire crosstalk can be abused to leak information in cloud-based FPGAs. However, long-wire crosstalk is limited to very small spatial distances where the receiver needs to be located next to the transmitter or victim on the same FPGA. This work shows how long-wire crosstalk can be extended to cross-FPGA information leakage with a novel Trojan phantom circuit. The phantom circuit is a self-contained circuit, isolated from rest of the FPGA logic. It uses crosstalk to spy on information within an FPGA and then exfiltrates the information across FPGAs by triggering RO stressors for cross-FPGA information transmission. The tested accuracy of the phantom circuits cross-FPGA information leakage channel can reach 90%. In addition to demonstrating a new security threat, this work also presents the first set of active monitoring and defense mechanisms for protection from cross-FPGA information leakage.**

***Index Terms*—FPGA Security, Hardware Attacks, Reconfigurable logic and FPGAs**

I. Introduction

In recent years, the use of FPGAs in public cloud computing data centers has exploded in size, from private deployments of FPGAs in projects such as Microsoft Catapult, to public deployments from companies such as Amazon Web Services (AWS). In settings such as AWS, users can upload their own hardware designs, and accelerate computations on the remote FPGAs. Many of the users' designs could be processing sensitive data and use encryption, or could be used for machine learning where users run their custom machine learning algorithms. In both of the example scenarios there is sensitive data, such as encryption keys or machine learning model parameters, respectively, which an adversary may want to steal.

Existing work has shown that information such as encryption keys can be leaked using long-wire crosstalk [1]. When victim and attacker are located next to each other on the FPGA, their logic may be mapped to wires which are physically adjacent, and cause crosstalk between each other. Attacker using Ring Oscillators (ROs) can measure delays induced in the wires due to crosstalk and learn information about the state of the victim's wire. Further, continuing FPGA research has shown that ROs can be used for other various purposes, from RO stressors [2] used to generate voltage and thermal changes, to RO sensors [3] used as receivers in information leaks through thermal channels. Moreover, ROs can be used to create Physically Unclonable Functions (PUFs) [4]. In all of these settings, use of ROs for information leaks is limited to within single FPGA and we classify these leaks as intra-FPGA information leaks.

Meanwhile, in this work, we explore novel inter-FPGA information leaks. We introduce new *phantom circuits*, which are malicious circuits which could be used for stealing information and sending it across FPGAs within same server. Phantom circuits for the first time combine multiple existing hardware FPGA threats to extend local, intra-FPGA information leakage to cross-FPGA or inter-FPGA information leakage. Phantom circuits are hardware modules unconnected to the rest of the FPGA circuit. Especially, they have no explicit inputs and outputs. Unlike most existing FPGA based malicious circuits, e.g., [2]–[4], our phantom circuits leverage an RO as a clock source and require no external clock. Clock gating or other means to disable the clock source thus cannot be used to disable phantom circuits. Further, when inserted into a victim design, the phantom circuit Trojan cannot be detected by checking wire connections, as they have no logical interaction with the victim circuit.

The phantom circuits combine existing approaches and give an example of an end-to-end means by which an attacker could leak information locally, within an FPGA, and send it across to a different FPGA in the same server. Our phantom circuits leverage the long-wire crosstalk effects, which have been recently explored in FPGAs [5] to steal information from the victim. Then, they utilize novel RO stressors to encode the leaked information into voltage changes of the FPGA chip. The voltage changes can then be observed by other receiver circuits on a different FPGA within the same server. Unlike most of the previous work, we develop an end-to-end means that can leak information from victim on one FPGA and send it to a different FPGA. This work solves number of technical challenges, such as developing a self-clocked RO long-wire crosstalk sensor, which enables the phantom circuit Trojan to be stealthy and harder to detect. Also, unlike prior work on cross-FPGA information leaks, we demonstrate how to collect, i.e. leak, the sensitive information from the victim locally and then send it across the FPGAs. Prior work mainly has focused

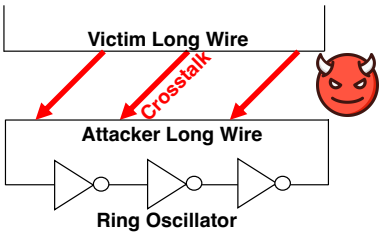

Fig. 1: Schematic diagram showing long-wire crosstalk between victim's long wire (top) and attacker's long wire (bottom). Attacker's long wire is part of an RO; the crosstalk is detected as changes in the frequency of the RO under control of the attacker.

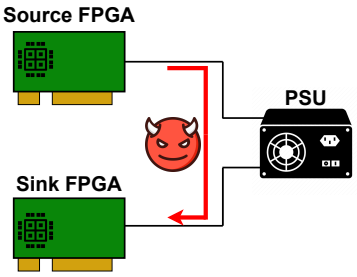

Fig. 2: Schematic diagram showing cross-FPGA information leakage through the shared power supply unit (PSU). The source FPGA generates voltage variations used to transmit the information, and the sink FPGA monitors voltage variations to receive information. In the case of our phantom circuits, it is the attacker's Trojan phantom circuit that modulates the power consumption of the source FPGA to transmit to the sink FPGA.

on analyzing cross-FPGA communication independent of how the data to be leaked is actually collected on the source FPGA.

Our work develops the first end-to-end demonstration that can leak information from victim on one FPGA and send it to a different FPGA. This highlights new set of threats in cloud-based FPGAs, and the need for new security mechanisms to be developed to protect cloud computing and other environments where FPGAs are used. Consequently, having demonstrated that local information leakage can be extended to cross-FPGA information leakage, this work also looks at counter-measures. It is currently difficult to analyze FPGA bitstreams and code to find malicious circuits. An antivirus program for FPGAs has been proposed [6], but the work may not catch all types of malicious circuits. Consequently, rather than try to find the phantom circuits, we propose defenders that focus on identifying and stopping the information leakage in an active manner.

The main characteristic which the defender can look at are systematic voltage changes in the operation of the FPGA, which could be signs of the transmitter modulating the voltage in order to achieve the cross-FPGA transmission. As one means for the defender to monitor voltage changes, we explore the use of CPU and motherboard voltage sensors available on today's servers. The sensors can easily be accessible via common Linux `lm-sensors` [7] software tool. The voltage data from various voltage domains monitored by `lm-sensors` could be used for discovery of the voltage changes that are signs of cross-FPGA communication. As an active defense triggered after the cross-FPGA transmission is suspected, the voltage changes in the system could be induced on purpose by the defender or the victim. We explore generation of disturbance by using CPU or GPU stressors to manipulate the system's voltage. Further, the victim circuit itself can generate large voltage disturbances as well, e.g., by using RO stressors, to create noise in the cross-FPGA channel.

A. Contributions

The contributions of this work are listed below:

1) We introduce novel phantom circuit Trojan circuits for information stealing using long-wire crosstalk; phantom circuits are first long-wire crosstalk circuits which are also self clocked using RO as a clock source.
2) We are first to combine long-wire crosstalk within FPGA with cross-FPGA information leakage to extend the distance of side-channel information leaks from intra-FPGA to inter-FPGA; the phantom circuits decode long-wire crosstalk and encode it into voltage changes that can be sensed across different FPGAs within a server.
3) We then demonstrate active detection mechanisms which could detect the cross-FPGA communication, and prototype defenses based on generating disturbance in the voltages using various types of stressors, which can prevent the cross-FPGA transmission from succeeding.

II. Background

This section discusses prior work in long-wire crosstalk in FPGAs and information leakage across FPGAs. Our phantom circuits are the first circuits to combine both of these effects in a novel way to create a security threat.

A. Long-wire Crosstalk

One of the major challenges in the design of FPGAs is the issue of crosstalk, particularly that caused by long wires. Crosstalk refers to the unwanted interference between two or more signals on a circuit, which can cause errors and impair the overall performance of the system. Moreover, crosstalk can also lead to information leakage, where sensitive data is inadvertently leaked to unintended parties [5].

Long wire crosstalk can be captured using ROs, which is a common method for exploiting the vulnerability of FPGAs to information leakage. As seen in Figure 1, the long wire leaks information to the adjacent RO wire through crosstalk, which affects the frequency of the RO, and changes in the RO frequency can be measured to learn the static signal on the victim's long wire. Existing work has shown attacks on cryptographic circuits using long-wires, and it is a difficult problem to prevent use of long-wires for sensitive information [8].

To mitigate the effects of crosstalk on FPGA designs, various techniques have been proposed, such as careful routing of wires to minimize crosstalk leveraging a combination of placement, routing, and obfuscation techniques to prevent

secret leakage on FPGA components [9]. At the same time, existing work has also shown that AWS defenses for ROs can be bypassed by use of novel ROs with latches or flip-flops [8], [10]. This makes defending or preventing long-wire crosstalk an open research problem.

B. Cross-FPGA Information Leakage

Information leakage is a critical concern in the design of FPGAs, particularly due to their reconfigurability and vulnerability to attacks. Figure 2 shows a schematic of the information leakage across two FPGAs sharing the same power supply unit. The sender can stress the power supply by running RO stressor circuits, and the receiver can measure the shared power supply voltage changes using RO sensors [11]. Cross-FPGA information leakage is a type of information leak attack that exploits the changes in the shared power supply unit (PSU) of FPGAs connected to the same server or workstation. For example, Giechaskiel et al. have proposed cross-FPGA covert channels to leak information from one FPGA to another [11]. However, the existing work has only shown covert channels, where transmissions is done on purpose. Our work focuses on side-channels, where the phantom circuit is used to collect side-channel information from the victim, and then encode it into cross-FPGA communication to extend the side-channel from local to cross-FPGA setting.

III. Threat Model

This work assumes a scenario where the adversary is able to insert the phantom circuit as a Trojan into the victim circuit. Since the phantom circuit is not connected to the victim circuit, it could even be inserted into the bitstream, as it does not depend on any logical connections to the victim's logic, not even the clock. This work further assumes the adversary is able to locate the phantom circuit's long wire next to victim's long wire from which information will be stolen. In addition, the phantom circuit uses large RO array, and we assume that there are sufficient FPGA resources available for the attacker's ROs. We assume that in single-tenant FPGA setting, this Trojan could be inserted during deployment or otherwise hidden in the code unbeknown to the victim.

In multi-tenant setting it would be actually much easier to insert the phantom circuit next to the victim as the attacker could be co-tenant on the same FPGA and not reacquire actual Trojan. In either case, we assume that the victim's sensitive information is carried by long wires, next to which the phantom circuits has its own long wires. We assume the sensitive information carried on the victim's long wire is not encrypted or otherwise protected, e.g., after sensitive data is decrypted, it has to be carried on the long wires for doing actual computation. Or, it could be the actual decryption keys that are carried on the long wires. Thus the information leaked by our phantom circuits could be both sensitive intellectual property, e.g. weights of machine learning models being written to block memories, or decryption keys.

To execute the cross-FPGA side channel, we assume the attacker controls a different FPGA in the same server. The second FPGA serves as the receiver of the information. We assume that attacker is able to synchronize such that the victim (with the phantom circuit Trojan) on the first FPGA executes at the same time as the attacker runs sensors on the second FPGA. The assumed setting of the two FPGAs, with victim and phantom circuit in first FPGA, and the attacker's sensors in second FPGA, is shown in Figure 3. Prior work has shown that, in cloud settings such as Amazon F1, it is quite easy to analyze and guess whether two FPGAs allocated to users are very likely to be on the same server [12].

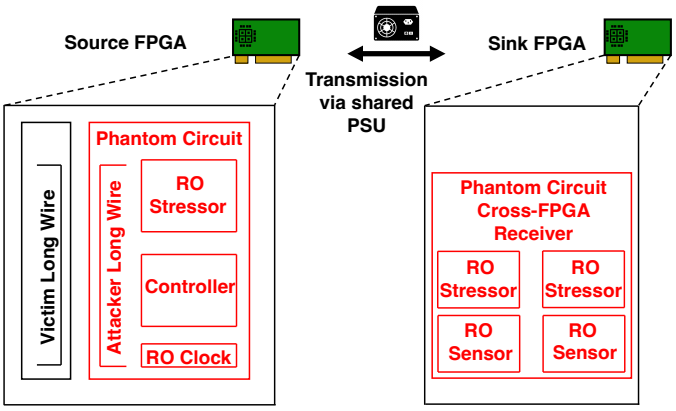

Fig. 3: Schematic of phantom circuit Trojan inserted within a source FPGA, and receiver circuit in the sink FPGA. The sizes of the modules and the FPGA are not to scale.

IV. Phantom Circuits

The goal of this work is to demonstrate how to extend intra-FPGA information leaks to inter-FPGA information leaks. Our work combines both ideas of side channels (to spy on victim within source FPGA) and covert channels (to actively send the information to the sink FPGA). In particular, this is first work to combine long-wire crosstalk with cross-FPGA data transmission via shared power supply.

The main components of the phantom circuit are the RO sensor used as the long-wire crosstalk receiver, and the RO stressor used as the cross-FPGA power covert-channel transmitter. The phantom circuit itself is instantiated inside a source FPGA on which the victim circuit is running, and there is separate sink FPGA, which could be used to recover the transmitted information. The placement of the phantom circuit is important as it uses crosstalk effect from long-wires being placed side by side to steal information. This is standard assumption in all existing work on long-wire crosstalk, which by design works if the victim and attacker are placed next to each other. The assumed setup is shown in Figure 3.

A. Stealing Information Through Crosstalk

Phantom circuits use crosstalk effect from long-wires being placed side by side to steal information [5]. The design involves 2 transmitter wires and 1 receiver wire. The transmitting wires are part of the victim. The information on the transmitting wires is the sensitive information that the

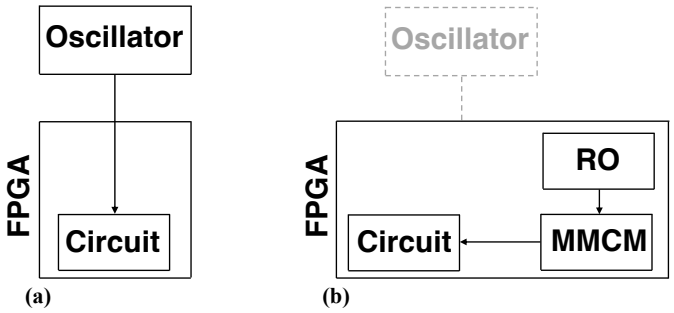

Fig. 4: (a) Typical FPGA circuit clocked using crystal oscillator available on the FPGA and (b) self-clocked circuit using RO and MMCM to generate stable clock signal.

phantom circuits aim to capture. The receiving wire is part of the RO sensor, which is inside the phantom circuit. The phantom circuit is unique due to the fact that it is completely isolated from the entire FPGA logic as it does not rely on the FPGA system clock as conventional methods currently use, e.g., [13]. Instead, we use a RO as a clock. The output from a ring oscillator is passed into a mixed-mode clock manager (MMCM) module primitive [14] which therefore makes the clock frequency further stable by our controller logic. By using RO clock, the phantom circuit has no connections to the other modules on the FPGA, not even clock connection. This makes it easier to insert as a Trojan, and also can make it harder to detect since no explicit inputs and outputs are used. To best of our knowledge, prior work on long-wire crosstalk never used self-clocked circuits for on the side of the RO sensor and receiver. Use of RO as a clock, vs. using external crystal oscillator, is shown in Figure 4.

B. Inter-FPGA Transmission of Information

Modern cloud FPGA deployments contain multiple FPGAs per server. Even if each whole FPGA is assigned to a different user (which is the single-tenant scenario), there are multiple users running on different FPGAs concurrently. We are first to show that the local side-channel information captured from long-wire crosstalk can be transmitted to a different FPGA by use of the PSU voltage-based communication channel. Prior inter-FPGA information leakage and communication has mainly been shown in covert channel setting [11], while we focus on side channels.

An example phantom circuit could consist of 5 power wasters, or RO stressors, each containing $2,000$ ROs allowing for covert transmission. More stressors with fewer ROs, or fewer stressors with more ROs should give similar results. These stressors can be turned on and off. When the stressors are turned on, the shared power supply is stressed, and other FPGAs connected to the power supply are provided with a lower voltage. On the other hand, when RO stressors are off, the voltage across shared power supply unit (PSU) returns to normal value. In our setting, the phantom circuit is the transmitter in the source FPGA. It stresses the shared PSU to achieve the cross-FPGA information transmission.

A sink FPGA can observe the voltage variations on the shared power supply by running RO sensors itself. The sink FPGA can consist, e.g., of 5 stressors and 4 receivers. Each stressor consists of 500 ROs while each receiver is made up of 5 ROs. The stressor ROs are present in the sink FPGA to stress the FPGA board's voltage regulator. As a result of the local stressing, the on-board voltage regulator is not able to mask the changes in the input 12V voltage coming from the shared PSU [11].

C. Design of Stealthy Phantom Circuits with Self-Clocked Circuits using ROs

Typical FPGA circuits are clocked from a clock signal coming from external crystal oscillators that may use phase-locked loops (PLLs) internally to produce stable output clock signals that are fed into the FPGA. Use of the clock requires at least one connection of a circuit to the FPGA's clock signal and clock tree.

In the phantom circuits, instead of using external clock, a RO inside the phantom circuit is used to provide a local oscillating signal that acts as a clock. This eliminates the need for an external clock source. Designing self-clocked circuits with ROs can be challenging, as the frequency and phase of the ROs can vary due to process variations, temperature changes, and aging effects. The use of ROs in self-clocked circuits can also introduce issues, such as increased jitter and clock skew, which can affect circuit performance. However, phantom circuits are simple and small circuits, and thus not affected by these typical issues. To help stabilize the clock signal, as seen in Figure 4, we feed the output of the RO into a MMCM module [14] to produce a stable clock for the phantom circuit.

Synchronization between the phantom circuit and the victim could be achieved by using the RO long-wire receiver for observing specific patterns in changes of victim's long wire that are used as trigger for data collection. Synchronization between the source and sink FPGAs can be achieved by the phantom circuit first transmitting a known pattern of bits, followed by the actual data that is being exfiltrated.

We believe ours is the first application for RO-based clock for use as reference clock for long-wire crosstalk measurements and resulting data transmission. Because phantom circuits are self-clocked and have no explicit logical inputs or outputs, they are difficult to find by examining logical connections within the victim. The phantom circuits do require an array of ROs, however, for complex victim designs this may be a small fraction of the area. Detection of phantom circuits by examining the victim's final circuit or the bitstream is left as future work. In case the phantom circuit is used in a multi-tenant setting, then it is separate from the victim and cannot be detected by analyzing any aspect of the victim.

V. Experimental Procedure

In the experiments, first, we tested the phantom circuits on the source FPGA separately from the sink FPGA, to

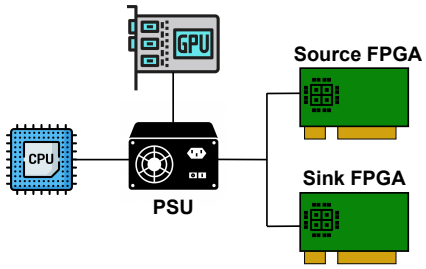

Fig. 5: Experimental setup: a shared power supply unit (PSU) is shared by FPGAs, as well as the server CPU and GPU.

determine the accuracy of the side-channel that uses the long-wire crosstalk within the FPGA. Then, second, we tested the combined end-to-end prototype to analyze the overall system and to determine the accuracy of the side channel information sent between two FPGAs. Figure 5 shows the experimental setup. In our setup, both sink and source FPGAs as well as the CPU and GPU all share the same power supply unit (PSU), which is typical in the cloud setting [11].

A. Side Channel Types Evaluated

In addition to the FPGA-based side channel leveraging shared PSU, we have also explored CPU and GPU based side channels. The evaluation of CPU and GPU was done to understand how the CPU and GPU also impact the shared PSU; and later the CPU and GPU is used as part of a defense procedure.

1) Main Phantom Circuits Side Channel: FPGA to FPGA: In our evaluated threat model, first, the phantom circuit in the source FPGA acts as receiver of the long-wire crosstalk to spy on sensitive information. The information could be for example encryption keys, which has been shown by prior work. The information is then stored inside the phantom circuit in registers or block RAM. To extend the side-channel and send the information to the second FPGA, the phantom circuit uses Manchester encoding, where a 1 becomes 10 and 0 becomes 01. The information encoded in this way is transmitted to the second FPGA. This is done through the use of the power-based channel. The source FPGA, here the phantom circuit, uses RO-based stressors to generate large voltage drop when transmitting a 1, and it stays idle (no voltage drops) when transmitting a 0. On the receiving end, the measurements on the sink FPGA are taken where counters record the oscillation count values of the sensor ROs [11]. For example, we take 500 RO measurements for each transmission of a single bit. The RO sensor uses the relative RO counts to differentiate between a 1 and 0 bit transmission. The relative RO counts of a 1 are usually lower than that of a 0 [5]. By using the relative RO counts, we can recover the original transmitted information.

2) CPU to FPGA Side Channel: We also explore use of CPU for side channel. We evaluate a side channel using CPU transmissions as a source, where heavy CPU loads replace the power draw of the FPGA source. Specifically, we use the open-source `stress` [15] program, which can be obtained through Debian-based Linux distribution package managers. By altering the number of threads that the stress program uses, we can examine the impact on the transmission from the CPU.

3) GPU to FPGA Side Channel: As another comparison, we also explore use of GPU for side channel. To evaluate the side channel using GPU transmissions as a source, we follow a process similar to that used for CPU transmissions. We utilize the open-source `gpu_burn` [16] program to stress the GPUs, fully utilizing their cores through Nvidia's CUDA platform. We compile and execute the `gpu_burn` program with the Nvidia drivers and CUDA versions mentioned in Section V-B.

B. Hardware Used

Our experiments were conducted using Xilinx Artix 7 AC701 boards and Kintex 7 KC705 boards which contain 28nm chips that are comparable, but have different optimizations [17]. While the Kintex 7 offers higher performance, the Artix 7 is designed for low power consumption [18]. Both FPGAs are equipped with a 200MHz oscillator and operate at a core VCCINT voltage of 1.0V. However, they use different regulators to convert the 12V PSU output to 1.0V [19] [20].

The computer used in our experiments comprises of 2 Xeon E5645 CPUs each with 6 cores and 12 threads running at 2.4GHz thus providing a total of 24 threads. There is also an Nvidia GeForce ZOTAC GT 430 GPU present in the system with a 1GB GDDR3 GPU memory comprising of 96 CUDA cores running at 0.7GHz. The GPU utilizes a Fermi Kepler architecture build on a 40nm process technology. The driver version used was Driver Version 390.157 while the CUDA Version was 8.0.61. Finally, the compile flag used was `compute_20`. For the shared power supply unit, we used a Corsair PSU with a load rating of 850W that has a Gold Certification that assures 90% efficiency at 50% load.

C. Self-Clocking with RO Setup

Since the phantom circuits use the output of a RO as a clock, the output of the RO is passed into the Xilinx Clocking Wizard v6.0 LogiCORE IP [14]. This logic core has the selection of MMCM primitive which was selected in our design to give a stable output clock frequency. Options to minimize the output jitter on the MMCM as well as using a clock safe start up control were selected when designing the primitive. Minimizing the output jitter minimizes the jitter on the output clocks, but at the expense of power and possibly output clock phase error. The Safe Clock Startup feature enables a stable and valid clock at the output [14]. From our experiments, the frequency of the RO was about 434MHz which served as the input frequency to the MMCM. The output frequency was set to 200MHz, similar to that of the native FPGA board clock.

VI. Experimental Results

In this section, we present the evaluation results.

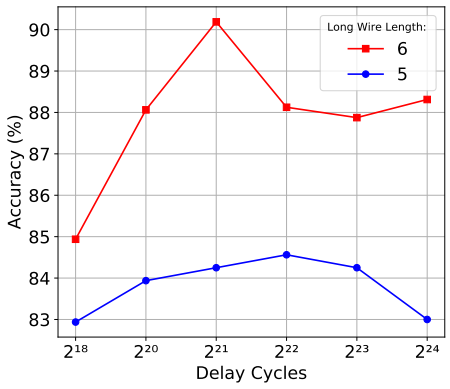

Fig. 6: Long wire accuracy varying the delay cycles transmission of long wire lengths of 5 and 6.

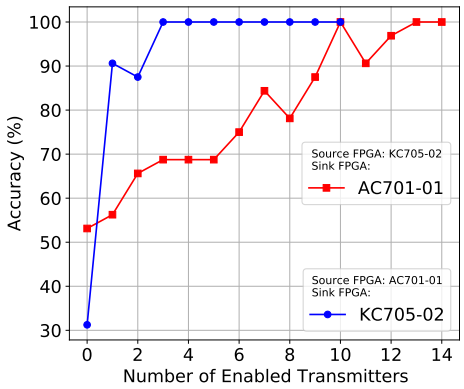

Fig. 7: Increasing the number of simultaneously enabled transmitters on the source FPGA board increases the accuracy of the cross-FPGA channel using ROs.

A. Long-wire Leakage Analysis

First, the phantom circuits uses long-wire crosstalk to obtain side-channel information from the victim. We assume that the victim and attacker long-wires can be placed next to each other.

For the long-wire crosstalk, two different lengths of long wires of length 5 and 6 were evaluated. We varied the delay cycles of each of these long wires. The delay cycles is the number of the RO oscillations on the receiving or sensing wire. From Figure 6, it can be seen as we increase the delay cycles the accuracy increases steadily and falls after 2^{21} cycles for a long wire of length 6 while for the long wire of length 5 the accuracy increases steadily until 2^{22} delay cycles are reached. For 2^{21} cycles and a long wire length of 6, the best accuracy of about 91% is obtained. The accuracy is computed based on the transmission of random 32 bit numbers across the long-wire crosstalk.

B. Cross-FPGA Channel Analysis

Based on the long-wire crosstalk results, we assume that the attacker is able to find long wires of sufficient length to leak the information from the victim. Once information is leaked, the attacker, i.e. phantom circuit, can use RO stressors on source FPGA and RO sensors on sink FPGA for the cross-FPGA portion of the leakage.

To test the cross-FPGA transmission, we altered the number of enabled transmitters on different types of source FPGAs. The AC701-01[1] has a maximum of 10 enabled transmitters while the KC705-02 has a maximum of 14. We observe in Figure 7 that when the KC705-02 serves as a sink, we achieve the highest accuracy with just 3 enabled transmitters on the source and for any additional transmitter, the accuracy remains the same. However, when the AC701-01 acts as a sink, the highest accuracy is achieved with about 10 enabled transmitters. The accuracy is computed based on transmission of random 32 bit numbers from the source to the sink FPGA.

[1]The suffix -01, -02, etc. is used to distinguish different FPGA boards of the same kind, for example AC701-01 and AC701-02 are two different AC701 boards available in our server.

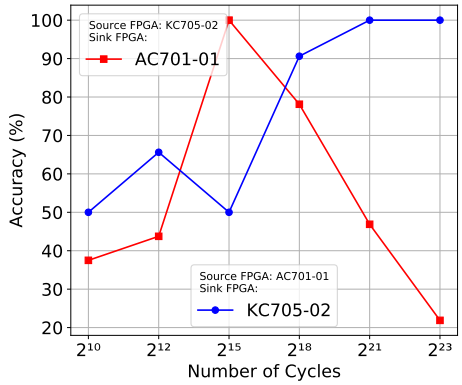

Fig. 8: Varying the number of cycles on the sink FPGA with 10 (when KC705 is the sink) and 14 (when AC701 is the sink) enabled transmitters respectively on the source FPGA.

Figure 8 also demonstrates that when the AC701-01 acts as a sink we achieve the highest accuracy with just 2^{15} cycles while when the KC705-02 acts as a sink the highest accuracy is achieved with 2^{21} cycles. The delay cycle for each FPGA is the number of cycles for how long we enable and disable the stressor ROs in the sink FPGA. The ranges were chosen experimentally which gives the best accuracy for the covert channel.

C. CPU and GPU Shared PSU Side Channels

As discussed before, we have also explored use of CPU and GPU with the shared PSU. The goal is to analyze if these could be used for side channels as well; and later we use CPU and GPU as sources of noise used in possible defenses.

1) CPU to FPGA Side Channel: To analyze use of CPU as information sender, we altered the number of CPU threads utilized by a stress program from 0, which equates to random measurements and no transmissions, to the maximum number of threads which were available on the computer. Specifically, we conducted the experiment on the CPU connected to PSU, which has a maximum capacity of 24 threads. We can observe in Figure 9 for both sink FPGAs as we increase the number

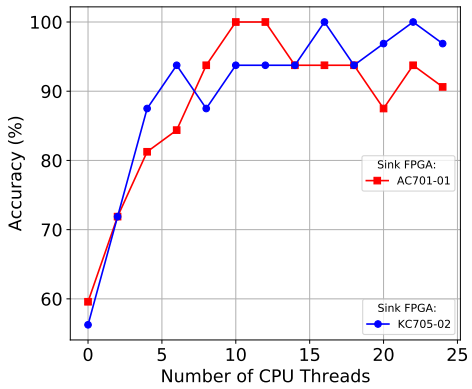

Fig. 9: Increasing the number of CPU threads increases the accuracy of the covert channel using ROs on the FPGA boards.

of threads, the accuracy increases. The maximum accuracy is reached and then drops after 16 threads on average.

2) GPU to FPGA Side Channel: To analyze use of GPU as information sender, the GPU was used as the transmitter while the KC705-02 and AC701-01 FPGA boards were used as receivers. We take 1500 measurements while using each FPGA as a receiver. While using the AC701-01 as a receiver, we observed the highest accuracy of 75% while with the KC705-02 we observed an accuracy of 100%. This highlights that the Kintex boards serve as better sink FPGAs while using GPUs.

VII. End-to-End Demonstration of Phantom Circuit Operation

To demonstrate complete phantom circuit operation, first, the phantom circuit is placed next to a simulated victim circuit. The long-wire crosstalk is used to receive a 256 bit Advanced Encryption Standard (AES) key. Previous work has shown crosstalk being used to leak AES keys [1]. Next, the received AES key is stored inside an internal register in the phantom circuit. Then, the RO stressors are activated. Each received bit is Manchester encoded into two bits. The RO stressors are turned on if the bit of Manchester encoding is 1, and remain idle if bit of Manchester encoding is 0. Following Figure 7 we use 10 and 14 transmitters when AC701 and KC705 are the transmitters, respectively. While the RO stressors on the source FPGA are activated, RO sensors on the sink FPGA are activated as well. Following Figure 8 we set the number of cycles per bit transmitted on the the KC705-02 board (sink FPGA) to be 2^{21} cycles. Note that all source and sink FPGAs as well as RO clock used for phantom circuits are set to be 200MHz.

A. Information Leakage Accuracy

In this section we report the combined accuracy of the phantom circuit. The phantom circuit on the AC701-01 acts as the source FPGA which leaks the transmitted information while the KC701-02 acts as the sink FPGA. The phantom circuits first leak a random 256 bit AES key locally through long-wire crosstalk, which in turn is sent to the sink FPGA through the cross-FPGA power covert channel. As seen in

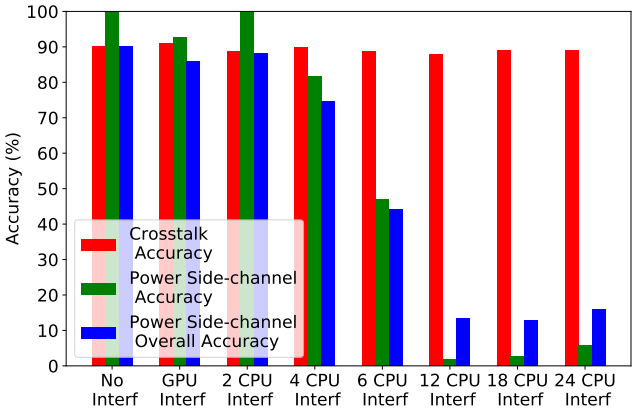

Fig. 10: Combined phantom circuit information leakage accuracy for random 256-bit AES key. Left side shows "No Interf" which means no interference or no defense. Other bars show accuracy when our different proposed defenses are used, defenses are discussed in Section VIII. In the lables, "Interf" means interference.

Long-wire Crosstalk (bit/s)	**Power Channel: Source AC701, Sink KC705 (bit/s)**	**Power Channel: Source KC705, Sink AC701 (bit/s)**
≤ 47.68 (channel error $\sim 10\%$)	≤ 0.1 (channel error $\sim 0\%$)	≤ 6.1 (channel error $\sim 0\%$)

TABLE I: Bandwidth for the long-wire crosstalk and different types of covert channel.

Figure 10, we achieve a cross-talk accuracy of about 90%, 100% for the power-side channel and finally about 90% for the overall accuracy with no interference. The effect of interference will be explained in Section VIII.

B. Bandwidth Analysis

As discussed before, we employ the use of the Manchester encoding for cross-FPGA transmission. With Manchester encoding, a 1-bit is encoded as a one followed by a zero indicating that the transmitters are enabled during one measurement period and then disabled during the consecutive period. This method however reduces the bandwidth by half but it allows us to distinguish between two consecutive measurements.

1) Long-wire Crosstalk Bandwidth Analysis: The bandwidth b_t of the long wire crosstalk is calculated as follows:

$$b_c = \frac{fc}{2 \cdot 2^t}$$

f_c represents the clock frequency and 2^t is the measurement period. Recall from Section V-C that the frequency of the FPGA board is 200MHz. We can see that 2^{21} cycles give the highest accuracy from Figure 6. The factor of 2 is used because we send two bits as a result of Manchester encoding. From Table I we observe 47.68 bit/s as the maximum value for bandwidth with nearly 10% channel error.

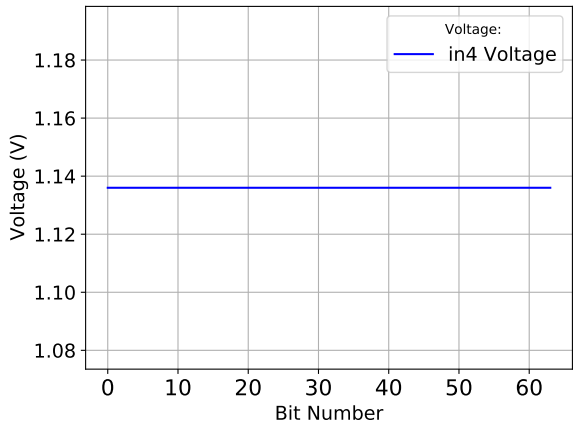

Fig. 11: CPU in4 voltage from server when no shared power supply unit covert transmission is occurring.

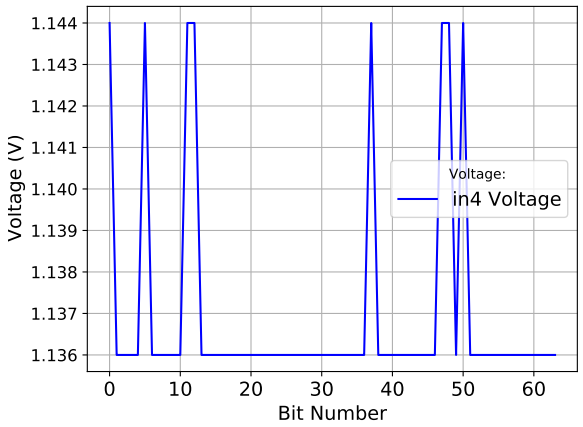

Fig. 12: CPU in4 voltage from server when shared power supply unit covert AC701 to KC705 leakage transmission is occurring.

2) Shared PSU Channel Bandwidth Analysis: The bandwidth b_t of the Power Covert Channel is calculated as follows:

$$b_p = \frac{fc}{2 \cdot 2^t \cdot M}$$

M represents the number of RO measurements taken in a single cycle. Recall from Section V that we take 500 RO measurements. In addition, we can see that 2^{15} and 2^{21} cycles result in the highest accuracy from Figure 8 when the AC701 and KC701 act as sinks respectively. Table I shows that with the AC701 acting as a sink, we observe 6.1 bit/s as the maximum value for bandwidth with nearly 0% channel error. With the KC705 acting as a sink, we observe 0.1 bit/s as maximum value for bandwidth with nearly 0% channel error.

3) Overall Bandwidth Analysis: The overall bandwidth is the minimum of the long-wire crosstalk bandwidth and the power covert channel bandwidth. This is given by:

$$min(\frac{fc}{2 \cdot 2^t}, \frac{fc}{2 \cdot 2^t \cdot M})$$

In Table I we observe 47.68 bit/s as the maximum value for bandwidth with nearly 10% channel error, however, the overall bandwidth is limited by the shared power channel.

VIII. Defending Against Phantom Circuits and Information Leaks

Having demonstrated the novel phantom circuits and ability to extend long-wire crosstalk information to cross-FPGA information leakage, we now propose defenses. Our defenses also apply to prior shared power supply unit covert channels [11]. Long-wire crosstalk can be simply defended by preventing placement of the Trojan next to victim circuit. While not as trivial to realize in practice, the long-wire crosstalk defense is not discussed further in this work, and we focus on the more novel defense for the share power supply unit covert channels.

A. Active Monitoring of Voltages

This paper provides the first active monitoring method for defending attacks that leverage shared power supply units. The voltage data from the CPU or the motherboard is one method that could be used to detect when an attack is happening. We utilize the open-source `lm-sensors` [7] program to collect the voltages from the system. The voltage information is collected at a rate of 10Hz. Figure 11 shows the voltage data in its idle state when there is no transmission between the source and sink FPGA. Since the system is idle, we expect the voltage to be stable. However, when a transmission occurs we see a change in the voltage levels as seen in Figure 12. One challenge for both the attacker and the defender is the interference from other activity on the CPU or GPU. The voltage drop may occur from an attack or the normal execution of programs on the CPU or GPU. On defense side, this means that CPU or GPU should be idle during the active monitoring. On the attacker side, this means that transmission will not work well when activity is happening on CPU or GPU. We utilize exactly this approach in our next step of the defense.

B. Information Leakage Disruption

To prevent the leakage of information during the power covert channel transmission, one defense mechanism would be to stress the CPU when an attack has been detected. Figure 10 shows that both the power-side channel and overall accuracy drop significantly when we stress the CPUs with 4 or more threads.

Another defense mechanism would be to stress the GPU when an attack has been detected. Figure 10 shows that both the power-side channel and overall accuracy drop significantly when we stress the GPU once the attack is detected. The stressing of the CPU and GPUs are used as a defense once an attack is detected and not to detect an attack themselves.

IX. Related Work

This section provides an overview of existing work on the different types of malicious circuits that can be instantiated in FPGAs, as well as the covert and side-channel attacks without physical accesses.

A. Malicious FPGA Circuits

The research of malicious circuits, or hardware Trojans, has seen significant growth in the past few decades, with the increasing complexity of modern Integrated Circuits (ICs) and the global collaboration in the semiconductor industry. Modern ICs usually contain billions of transistors and are fabricated by semiconductor foundries that collaborate internationally, thus, the research of malicious circuits is vital to secure the chips that power the basis of critical infrastructures.

Malicious circuits are the stealthy circuits implemented by attackers in the chip to accomplish designed attacks or for information leakage purposes. Traditionally, the field of hardware Trojan research mainly refers to the creation and defense of additional circuits at the transistor level that enable information leakage or damage. In [21], Jain et al. summarized the different types of hardware Trojans in ICs. Attackers are able to instantiate combinational, sequential, or analog malicious circuits in the IC fabrication process and can steal information or perform function manipulations. A more detailed survey on the hardware Trojans is provided by Xiao et al. [22], the authors summarized the hardware Trojan design, countermeasures, and threat models. The countermeasures include the Trojan detection, which aims to verify the fabricated ICs, and the design-for-trust, which adopts the prevention measures in the design phase. To compare various Trojan detection works, Shakya et al. [23] put forward a vulnerability analysis flow and benchmarks for the hardware Trojan research.

The re-configurable hardware, such as FPGAs, opens up new opportunities in the malicious circuit research. Besides the insertion of transistor-level Trojans in the chip fabrication process, FPGAs allow for the creation of logic components that cause logical malfunction. For the transistor-level Trojans, Mal-Sarkar et al. [24] investigated the hardware Trojans that can be inserted in the FPGA device production process, based on the diverse activation and payload characteristics. For the logic-level malicious circuits, Chakraborty et al. [25] proposed the direct modification on the FPGA configuration bitstream to insert hardware Trojans. It bypassed the pre-deployment verification step and could be used to steal information and cause severe malfunction. For specific hardware accelerators, Ye et al. [26] demonstrated the feasibility of adding malicious circuits into FPGA CNN accelerator, where attackers acquired the privilege to control the CNN classification results.

Our work, meanwhile, presents a new type of self-clocked malicious circuit that is the phantom circuit. We are also first to show how intra-FPGA side channels can be extended to inter-FPGA information leaks.

B. Remote FPGA Attacks

Previous research has explored the remote attacks targeting FPGAs in numerous aspects, including the covert and side channels through power, thermal and crosstalk, and degradation attacks that can damage FPGA itself [27]. This work is built upon the related research on the crosstalk effects and resource sharing problem.

In [28]–[30], the authors demonstrated that power and thermal sensors, like ROs and Time-to-Digital Converters (TDCs), can be leveraged to construct covert channels and steal information within cloud-based FPGAs or across cloud-based FPGAs, but never combining both ideas. Among others, Ramesh et al. were able to extract bytes of the final round key of 128-bit AES using a ring oscillator and then recover the original AES key by inverting the key schedule [1]. Trochatos et al. showcased thermal covert channels in a SmartSSD between SSD and FPGA [31]. In [32], Matas et al. presented the Degradation-of-Service attack on data center FPGAs, that is, the large energy wasting logic based on ROs could drain excessive power and cause the FPGA boards to shut down. Tian et al. showed how cloud FPGA infrastructures can be mapped by using PCIe contention [33]. Recently, in [34] the authors demonstrated how co-located FPGA accelerators can be fingerprinted by using the PCIe information. Giechaskiel et al. [5] put forward the crosstalk effects between adjacent long wires and shown the potential usages. Recently, Giechaskiel et al. [11] demonstrated the covert-channels between different FPGA devices through a shared power supply unit (PSU). The crosstalk and power supply sharing problem serve as the basis of our phantom circuit project.

X. Conclusion and Future Work

In this paper, we introduced the first self-clocked FPGA circuit that leaks information through long-wire crosstalk within an FPGA and then amplifies it for cross-FPGA transmission. First, the sensitive information is leaked via a long-wire side-channel to the attacker's phantom circuit. This circuit can be inserted as a Trojan in case of single-tenant cloud-based FPGAs, or it could be a separate, malicious tenant in case of multi-tenant cloud-based FPGAs. Second, the information leaked within the source FPGA is transmitted to the sink FPGA where receiver circuit can decode it. On the sender's side, the proposed phantom circuits are completely isolated from the rest of the FPGA. They require no explicit inputs and outputs, not even a clock. We further demonstrated that phantom circuit can leak sensitive information with an accuracy of about 90%. As a defense, we analyzed how CPU and motherboard voltage sensor data can be used to detect the shared power supply transmission in our setup. After detection, the transmission can be disturbed by running CPU or GPU stressors. Future work can explore in-depth defense mechanism, motivated by our threat model and demonstration. E.g., future work can explore the use of other FPGAs within cloud server for generation of noise on the shared power supply unit for attack prevention. Alternatively, servers with separate power supplies for the different FPGAs can be analyzed.

Acknowledgements

We would like to thank Ilias Giechaskiel for his discussions and contributions to early versions of the code and paper. This work was supported in part by National Science Foundation grants 2245344 and 1901901.

REFERENCES

[1] 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 *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 45–52, 2018.

[2] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on fpgas," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1111–1116, 2018.

[3] 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)*, pp. 133–137, 2010.

[4] S. R. Sahoo, S. Kumar, and K. Mahapatra, "A novel ropuf for hardware security," in *2015 19th International Symposium on VLSI Design and Test*, pp. 1–2, 2015.

[5] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: Information leakage and covert communication between fpga long wires," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 15–27, 2018.

[6] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale + fpgas," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, sep 2020.

[7] "Lm-sensors package." https://github.com/lm-sensors/lm-sensors.

[8] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "Measuring long wire leakage with ring oscillators in cloud fpgas," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 45–50, 2019.

[9] Y. Luo, S. Duan, and X. Xu, "Fpgapro: A defense framework against crosstalk-induced secret leakage in fpga," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, nov 2021.

[10] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: Cross-slr covert channels on multi-tenant cloud fpgas," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pp. 1–10, 2019.

[11] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "C3apsule: Cross-fpga covert-channel attacks through power supply unit leakage," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1728–1741, IEEE, 2020.

[12] S. Tian, I. Giechaskiel, W. Xiong, and J. Szefer, "Cloud fpga cartography using pcie contention," in *Proceedings of the International Symposium on Field-Programmable Custom Computing Machines*, FCCM, May 2021.

[13] J. Lamoureux and S. J. E. Wilton, "Fpga clock network architecture: Flexibility vs. area and power," in *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, FPGA '06, (New York, NY, USA), p. 101–108, Association for Computing Machinery, 2006.

[14] "Clocking wizard v6.0 logicore ip." https://docs.xilinx.com/r/en-US/pg065-clk-wiz/Clocking-Wizard-v6.0-LogiCORE-IP-Product-Guide.

[15] "Linux stress test." https://linux.die.net/man/1/stress/.

[16] V. Timonen, "Multi-gpu cuda stress test." https://github.com/wilicc/gpu-burn.

[17] "Xilinx, inc., "7 series fpgas data sheet: Overview (ds180)"." https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf.

[18] "Xilinx, inc., "7 series product brief." https://www.xilinx.com/publications/prod_mktg/7-Series-Product-Brief.pdf.

[19] "Ac701 evaluation board for the artix-7 fpga (ug952)." https://www.xilinx.com/support/documentation/boards_and_kits/ac701/ug952-ac701-a7-eval-bd.pdf.

[20] "Kc705 evaluation board for the kintex-7 fpga (ug810)." https://www.xilinx.com/support/documentation/boards_and_kits/kc705/ug810_KC705_Eval_Bd.pdf.

[21] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2021.

[22] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 1, pp. 1–23, 2016.

[23] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, pp. 85–102, 2017.

[24] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware trojan attacks in fpga devices: threat analysis and effective counter measures," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, pp. 287–292, 2014.

[25] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware trojan insertion by direct modification of fpga configuration bitstream," *IEEE Design & Test*, vol. 30, no. 2, pp. 45–54, 2013.

[26] J. Ye, Y. Hu, and X. Li, "Hardware trojan in fpga cnn accelerator," in *2018 IEEE 27th Asian Test Symposium (ATS)*, pp. 68–73, IEEE, 2018.

[27] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud fpgas: A survey," *arXiv preprint arXiv:2005.04867*, 2020.

[28] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Remote power side-channel attacks on bnn accelerators in fpgas," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1639–1644, IEEE, 2021.

[29] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud fpgas," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 298–303, 2019.

[30] S. Tian, S. Moini, A. Wolnikowski, D. Holcomb, R. Tessier, and J. Szefer, "Remote power attacks on the versatile tensor accelerator in multi-tenant fpgas," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 242–246, IEEE, 2021.

[31] T. Trochatos, A. Etim, and J. Szefer, "Covert-channels in fpga-enabled smartssds," *ACM Trans. Reconfigurable Technol. Syst.*, dec 2023. Just Accepted.

[32] K. Matas, T. La, N. Grunchevski, K. Pham, and D. Koch, "Invited tutorial: Fpga hardware security for datacenters and beyond," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 11–20, 2020.

[33] S. Tian, I. Giechaskiel, W. Xiong, and J. Szefer, "Cloud fpga cartography using pcie contention," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 224–232, 2021.

[34] C. Fang, N. Miao, H. Wang, J. Zhou, T. Sheaves, J. M. Emmert, A. Sasan, and H. Homayoun, "Gotcha! i know what you are doing on the fpga cloud: Fingerprinting co-located cloud fpga accelerators via measuring communication links," 2023.