# A Practical Remote Power Attack on Machine Learning Accelerators in Cloud FPGAs

Shanquan Tian*, Shayan Moini†, Daniel Holcomb†, Russell Tessier† and Jakub Szefer*
*Yale University, New Haven, CT, USA Email: {shanquan.tian, jakub.szefer}@yale.edu
†University of Massachusetts, Amherst, MA, USA Email: {smoini, dholcomb, tessier}@umass.edu

***Abstract*—The security and performance of FPGA-based accelerators play vital roles in today's cloud services. In addition to supporting convenient access to high-end FPGAs, cloud vendors and third-party developers now provide numerous FPGA accelerators for machine learning models. However, the security of accelerators developed for state-of-the-art Cloud FPGA environments has not been fully explored, since most remote accelerator attacks have been prototyped on local FPGA boards in lab settings, rather than in Cloud FPGA environments. To address existing research gaps, this work analyzes three existing machine learning accelerators developed in Xilinx Vitis to assess the potential threats of power attacks on accelerators in Amazon Web Services (AWS) F1 Cloud FPGA platforms, in a multi-tenant setting. The experiments show that malicious co-tenants in a multi-tenant environment can instantiate voltage sensing circuits as register-transfer level (RTL) kernels within the Vitis design environment to spy on co-tenant modules. A methodology for launching a practical remote power attack on Cloud FPGAs is also presented, which uses an enhanced time-to-digital (TDC) based voltage sensor and auto-triggered mechanism. The TDC is used to capture power signatures, which are then used to identify power consumption spikes and observe activity patterns involving the FPGA shell, DRAM on the FPGA board, or the other co-tenant victim's accelerators. Voltage change patterns related to shell use and accelerators are then used to create an auto-triggered attack that can automatically detect when to capture voltage traces without the need for a hard-wired synchronization signal between victim and attacker. To address the novel threats presented in this work, this paper also discusses defenses that could be leveraged to secure multi-tenant Cloud FPGAs from power-based attacks.**

***Index Terms*—Cloud FPGA, FPGA Security, Hardware Accelerators, High-Level Synthesis, Machine Learning Security, Vitis**

## I. Introduction

FPGAs in the cloud offer low latency, high throughput and energy efficiency for the acceleration of compute-intensive applications. Numerous Cloud FPGA accelerators have been demonstrated and commercialized, such as search engine accelerators [1] and machine learning accelerators [2].

Public cloud providers, such as Amazon Web Services (AWS) [3] and Alibaba Cloud [4], provide access to commercial FPGAs in their data centers and have adapted Xilinx development tools for building and running custom accelerators on their platforms. Since February 2020, AWS has supported the Xilinx Vitis development environment [5], a unified all-in-one platform including Vivado and SDAccel, to reduce the design effort of FPGA application development. With Vitis, the top ports of the user's logic are standardized, along with the memory allocation, and data communication between the accelerator modules and FPGA hardware shell.

FPGA multi-tenancy, in which multiple independent users share logically isolated portions of the FPGA fabric at the same time, is being actively explored by Cloud FPGA vendors and researchers [6]. This approach has the potential to fully utilize FPGA resources, increasing vendor profits.

Improved implementation tools have allowed for faster hardware accelerator development cycles for Cloud FPGAs. Machine learning accelerators, including domain-specific instruction set architecture (ISA) based accelerators, have been developed. However, to date, the security of machine learning accelerators in commercial public Cloud FPGAs (i.e. AWS F1) has not been fully studied. Previous research has focused on remote attack prototyping using local boards in laboratory settings rather than state-of-the-art compute environments in the cloud. Accelerators generated with the Vitis framework typically exhibit a block-based structure that performs easily-recognizable operations. For example, shells used by Cloud FPGAs often include standardized ports and protocols interfaced to a user's custom logic design. As this work shows, with more standardized operations, a malicious co-tenant can still easily detect and observe the operation of the victim user's logic on the same FPGA.

We show that malicious co-tenants are able to insert on-chip sensors, such as time-to-digital converters (TDCs), into a state-of-the-art development framework, allowing for remote power attacks in a public cloud environment. Although power-based attacks have been previously demonstrated [7]–[9], we specifically focus on the AWS F1 platform and accelerators generated using the Xilinx Vitis framework. The regular interface structure of these designs make them an inviting target for security attacks. The specific contributions of this work include:

- Design of a practical power trace collection scheme that can be inserted into a Vitis project to collect voltage trace signatures in Cloud FPGAs.
- Collection of power signatures of the FPGA shell and accelerators, providing the first security analysis of designs generated using Vitis tools on AWS F1.
- Analysis of three existing, typical, Vitis "victim" accelerator modules. Power traces are captured using an "attacker" accelerator module to identify different voltage patterns, or fingerprints, related to operation of the shell, DRAM, and the accelerator modules.
- Demonstration of a new *auto trigger* mechanism for triggering the capture of power traces based on FPGA shell or accelerator activity.

This work was supported in part by NSF grant 1901901.

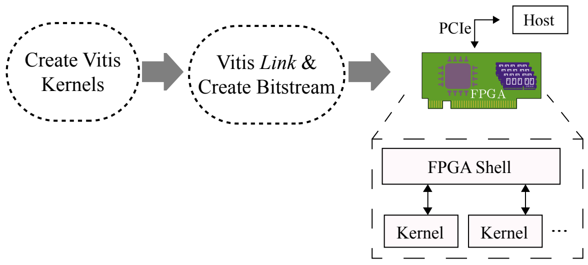


Fig. 1: Xilinx Vitis design flow in AWS F1 FPGA-accelerated instances. Custom modules are "linked" along with the FPGA shell to generate the bitstreams and AFIs.

- Discussion of potential defenses for securing multi-tenant accelerators in Cloud FPGAs.

## II. Background

This section provides background on Cloud FPGAs, Xilinx Vitis tools, and accelerators used for evaluation.

### A. Cloud FPGAs and AWS F1 Instances

This work focuses on AWS F1 instances which include virtual machines with access to Xilinx UltraScale+ `VU9P` FPGAs. The design tools are provided in the `aws-fpga` repository [10], including adapted Xilinx scripts targeting an AWS F1 platform. Although AWS currently only supports single-tenant configurations in which each user is given access to the full FPGA, multi-tenant FPGA use is actively being explored by the research community [6]. The use of multi-tenancy allows for improved resource utilization and FPGA sharing [11]. Multi-tenancy can be achieved by assigning each user's logic to its own, fixed reconfigurable region, or multiple users' designs can be compiled together for optimal resource allocation.

### B. Xilinx Vitis Flow in Cloud FPGAs

Figure 1 shows the Vitis design flow on AWS F1. HLS and RTL modules created by designers are encapsulated as "kernels" with a standard interface, including clock and reset signals, AXI4-based control and data ports. Developers do not need to instantiate port connections. Vitis generates the control logic to handle the communication between the FPGA shell (SH) and kernels, links all modules using the `link` command, and performs compilation and synthesis of the whole design. On AWS F1, the final `awsxclbin` file generated by the synthesis and implementation process can be used to program the FPGAs. As kernels are created and encapsulated before linking with the FPGA shell, logical isolation between the signals of different kernels is expected unless otherwise specified.

### C. Three ML Accelerators Used for Evaluation

Xilinx Vitis provides a library of templates to help developers design accelerators for different algorithms. In our experiments, we adopted the Vitis systolic array [12] and vector addition [13] accelerator templates, since matrix multiplication and vector processing present heavy computational workloads in many convolutional neural network models. Since developers often write their own accelerators rather than using template code, a domain-specific ISA based accelerator. the Versatile Tensor Accelerator (VTA) [14], was evaluated and analyzed. VTA is an open-source hardware accelerator for machine learning (ML) algorithms, implemented using C++ and OpenCL. The analyses of both custom accelerator and Vitis templates provide important perspectives on the security of machine learning accelerators in Cloud FPGAs.

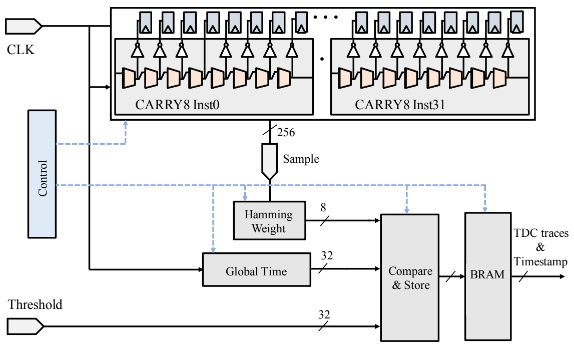


Fig. 2: Diagram of the Compressed TDC RTL Kernel. The control module includes the trigger logic and additional functions. The Vitis standard kernel ports are not shown.

### D. Threat Model

A variety of attacks have been demonstrated on Cloud FPGAs. The attacks can be broadly classified into disruptive attacks [15] and information leakage attacks, which include thermal attacks [16], cross-talk based attacks [17] and power or voltage based attacks [18].

In this work, we assume a victim and an attacker can be co-tenants on a multi-tenant FPGA, and thus share hardware resources, including the power distribution network (PDN). Although FPGA "antivirus" programs can check bitstreams for malicious circuits [19], e.g. ring oscillators (ROs) or time-to-digital converters (TDCs), such protections can be bypassed by clever attackers who develop ROs or TDCs that are not easily detected [20]. We assume that a malicious user is able to load their design onto a Cloud FPGA and collect side-channel information, in our case voltage traces from a TDC-based voltage sensor kernel we developed.

## III. Remote Attacks on Cloud FPGA Accelerators

This section first presents an overview of our *Compressed TDC RTL Kernel (CTRK)*, then describes our system implemented in the AWS F1 environment.

### A. Practical Power Trace Measurement on AWS F1

To perform power attacks in a Cloud FPGA environment, we use a *Compressed TDC RTL Kernel (CTRK)* to capture extended power traces.

*1) Compressed TDC RTL Kernel (CTRK):* As shown at the top of Figure 2, our TDC sensor is implemented using 32 `CARRY8` carry chain logic elements. The clock signal traverses multiple carry chain elements during a clock cycle and sets these flip-flop outputs to 1. The Hamming weight of the 256-bit output sample can be used to measure the traversal time of the clock signal. This value is correlated with the voltage levels across the FPGA's PDN as higher voltage results in reduced carry chain element delay and samples with a higher Hamming weight.

The Compressed TDC RTL Kernel (CTRK) captures and stores only "interesting" data (e.g. voltage drops above a certain threshold) and stores only the 8-bit Hamming weights of each sample. Since random noise accounts for the biggest variation in TDC traces and informative drops are sparse, we skip the collection of uninformative random fluctuations and only store TDC values lower than a pre-determined threshold, along with corresponding timestamps. CTRK operation contains two steps:

- Step-1 CTRK execution without other active accelerators or with the other accelerators being idle. Based on this data, the threshold $T_{ave}$ for Step 2 is computed, where $T_{ave}$ equals the average of the TDC traces.
- Step-2 Collection of extended (long-term) TDC traces when the accelerators are running. Only TDC samples below $T_{ave}$ and timestamps are stored in memory.

CTRK implementation is shown in Figure 2. The sampling rate of the 32 `CARRY8` carry chain logic elements is set to 5 clock cycles. The `Global Time` module is a counter with one increment every 10 clock cycles. For a clock frequency of 125 MHz, the 32-bit `Global Time` counter, which can be expanded if needed, covers $\sim 2.5$ minutes in one run which is more than sufficient for the tested victim modules' operation duration. The threshold can be set to 256 (maximum value) during Step 1, so the CTRK captures all the samples, and the data collected can be then used to choose a proper threshold to skip uninformative traces in Step 2. Once the threshold is selected, the `Compare & Store` module stores the 8-bit samples below that threshold, along with the timestamp for each value, in the Block RAMs (`BRAM` in the figure).

*2) Threshold for CTRK:* The size of Block RAMs for TDC outputs and timestamps is tunable in our design to accommodate different needs and monitoring environments. The value of the CTRK threshold depends on the allocated Block RAM space and the desired trace monitoring duration, since the sampling speed is faster when the threshold is bigger. When the threshold increases, the TDC traces of larger Hamming weights and smaller voltage drops are recorded, thus the sampling speed increases as they are more common. To achieve the desired duration and collect as much information as possible, we tune the threshold carefully such that the Block RAM is almost filled and traces cover the duration of the run of the victim module that the attacker is trying to monitor.

### B. System Diagram

Figure 3 shows a system diagram of a hardware architecture implemented on AWS F1. As introduced in Section II-B, Vitis encapsulates the TDC kernel and accelerators (VTA, systolic array or vector addition) and generates a top-level module of the user's custom logic (CL) to control and coordinate the kernels. We assume the CL module generated by Vitis provides logical isolation between the signals of different kernels. As

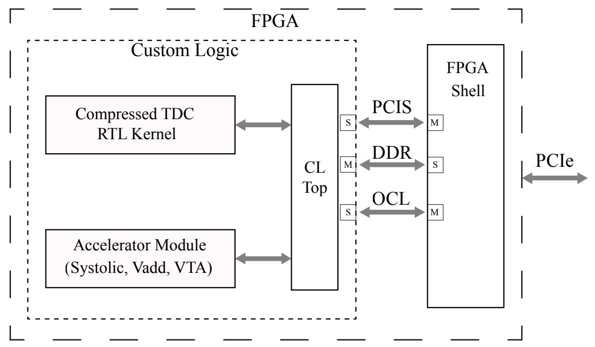

FPGA
Custom Logic
Compressed TDC RTL Kernel
Accelerator Module (Systolic, Vadd, VTA)
CL Top
PCIS
DDR
OCL
S
M
S
M
S
M
FPGA Shell
PCIe

Fig. 3: Overview of the experimental setup on AWS F1. Only some of the SH-CL ports are listed for brevity.

shown in Figure 3, the custom logic communicates with the FPGA shell through interfaces provided by AWS. The PCIe Slave (PCIS) interface is used for inbound PCIe transactions from the host computer to the FPGA. DDR interfaces with the DRAM DDR4 controllers on the FPGA board, and OCL interacts with the AWS OpenCL runtime library for OpenCL Kernel accesses [21]. Figure 3 does not show all the ports between the CL and the shell [22].

Our experiments were conducted on the CentOS-based FPGA Developer AMI 1.9.1 (Xilinx tools 2020.1) on an `f1.2xlarge` instance. The FPGA bitstreams were built using the official `aws-fpga` toolkit with the FPGA shell version `v04261818` [10]. The three accelerators were evaluated separately, each time by running accelerator module together with the CTRK on the FPGA. For each experiment, the collected TDC traces covered the whole process of running the FPGA accelerator from the host computer. To further our understanding about host computer and FPGA accelerator interactions, we added logic into the CL to monitor the first transaction times of PCIS, DDR, OCL ports for `write` and `read` operations.

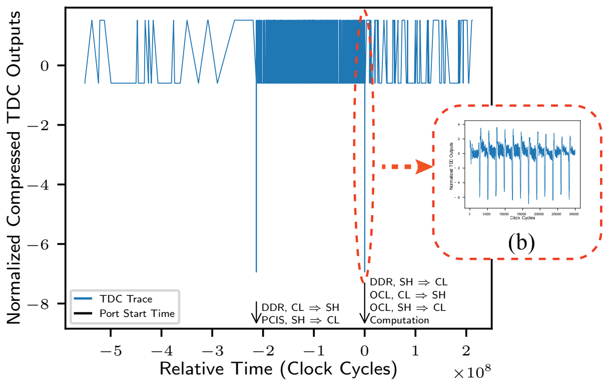
There were two steps for each evaluation. First, the proper threshold for CTRK was determined by running preliminary collections, as discussed in Section III-A2. Then, the accelerator module ran in parallel with the configured CTRK. The compressed TDC traces and port monitoring results are shown in Figures 4, 5, 6 and 7. Please note that for an actual attack, there would be no access to the ports of the shell or the victim. The logic for port monitoring is only used during development to help with understanding the voltage fluctuations recorded by the TDC with respect to the operations in the FPGA. The experimental results of the auto-triggered attack are described in Section IV-B.

## IV. Evaluation of Typical Accelerators in Cloud FPGAs

This section presents the evaluation results for the three victim accelerators on AWS F1. The clock frequency was set to 125 MHz in all experiments.

### A. Power Traces Analysis

The TDC outputs in all experiments are normalized, in which $x = (x - \mu)/\sigma$, $x$ are the TDC samples, $\mu$ is the mean and $\sigma$ is the standard deviation.



(a)

Fig. 4: (a) The voltage trace of sample execution of the VTA accelerator module in AWS F1. The readouts from compressed TDC and the relative time is normalized. (b) The zoom-in figure of the accelerator computation which happens exactly at the same time as the big drop in (a). Please note that the magnitude of x-axis is $10^8$, and the TDC trace in the small figure only covers $\sim 10^4$.

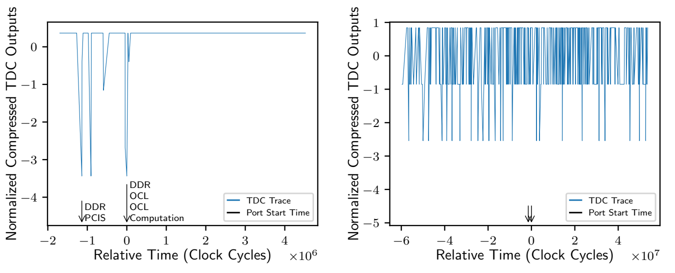
*1) VTA:* Figure 4 shows the TDC trace overview of the VTA accelerator computation on AWS F1. Due to the design of CTRK, the trace only shows voltage drops below the threshold. Because a timestamp (clock cycle) is stored for each sample, the full voltage trace can be reconstructed with the voltage changes shown at correct times on the x-axis. In addition to the normalization of the TDC outputs, the relative time is adjusted by setting the kernel computation start time to 0.

The duration of the accelerator computation is very short compared to the whole overview, as shown in the zoom-in window in Figure 4. The x-axis magnitude is $10^4$ in Figure 4b and $10^8$ in Figure 4a, so the TDC trace details in the latter look like a single drop in the former at time 0 (details are shown in Figure 9).

Figure 4a labels the transaction start times of the CL ports, including PCIS, DDR and OCL. Near time $-2 \times 10^8$, at the instance of the first big trace drop, port PCIS starts moving data from SH to CL and DDR moves data from CL to SH. Since PCIS is used to send data from the host computer to the FPGA board, and DDR is used to control FPGA board DRAM, we can infer that the `CL Top` control logic reads data from the host computer and stores it in the DRAMs. Subsequently, the DDR (SH to CL direction) and OCL ports start transfers at the same time as accelerator computation starts, indicating that the accelerator starts running and reads data from DRAMs at this time.

*2) Systolic Array and Vector Addition:* The TDC traces for the systolic array and vector addition are shown in Figure 5. For the systolic array in Figure 5a, the traces and labels are similar to the VTA experiment, except that the time scale is much smaller. In Figure 5b, the informative voltage traces cannot be distinguished from noise as vector addition consumes fewer resources than matrix multiplication for the same input, and does not generate noticeable voltage drops.

*3) Comparison:* Figure 6 shows a comparison of preparation times, that is, the time interval between the start of SH-CL



(a) Systolic Array (b) Vector Addition

Fig. 5: Compressed TDC voltage traces for systolic array and vector addition modules. (a) For the systolic array, the big voltage drops correspond with the data movement activities between the shell and custom logic. (b) The TDC traces for the vector addition module does not demonstrate any large drops. The arrows label the start time of ports, which are the same as in (a).

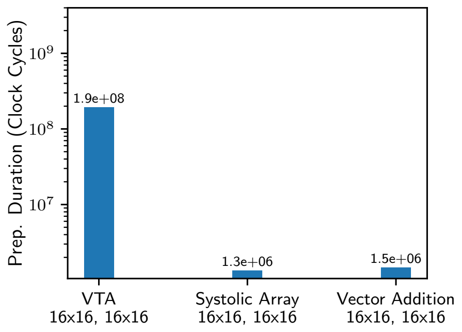


Fig. 6: Comparison of preparation time for different accelerators. The preparation time is the time between when the first DDR and PCIS activity occurs, and the time when the voltage drop due to the start of accelerator occurs.

port activity, e.g., DDR and PCIS activity, and accelerator computation for the three accelerators considering the same input data size. VTA requires a much longer time than the systolic array and vector addition modules, although they have input matrices of the same size. As shown in Figure 7, VTA preparation times change with different input sizes, but they are all much longer than the other accelerators in Figure 6. The preparation time is also related to the data matrix shape and not just the total data size. For example, the preparation time of input data (8x256, 256x256) is shorter than (16x32, 32x32). The comparison suggests that the system-level structure of the accelerator execution environment, its logic flow, and its data structures play major roles in execution efficiency. As shown, sometimes bigger data actually takes less time to move to the DRAM on FPGA. Experiments were repeated multiple times to confirm the times shown in Figure 6 and Figure 7 are consistent.

## *B. Auto-Triggered Remote Attacks on AWS F1*

The design of the auto trigger takes into account that the two TDC trace drops, induced by the start of SH-CL communication or accelerator computation, are distinguishable from the idle state and other activities, as shown in Figure 4. Figure 8 shows the process of an auto-triggered TDC trace collection.

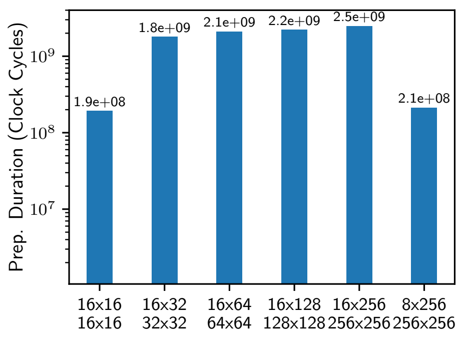


Fig. 7: The preparation time for different input data sizes of VTA.

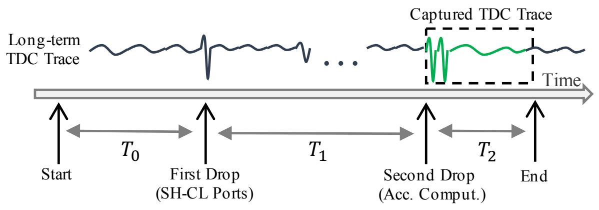


Fig. 8: The measurement process for auto trigger mechanism. The trace waveform is not actual data.

At the beginning, the auto trigger starts monitoring the TDC fluctuations and detects the first drop (the start of SH-CL communication) after an unknown duration $T_0$. It again detects the second drop (the start of accelerator computation) after $T_1$ and then takes measurements for $T_2$. In this case, the two big drops can be detected by selecting correct thresholds.

For example, in Figure 4, the auto trigger threshold can be set as the corresponding TDC output to the normalized value $-4$ to recognize the starting time of CL ports and accelerator computation. For each experiment, the attacker needs to test different threshold values to set the TDC trigger correctly until the attack succeeds.

The evaluation result is shown in Figure 9b. The data collected with the auto trigger is highly similar to the data collected with the hard-wired trigger (Figure 9a). Using the similar reconstruction method described in [23], the power traces collected with the auto trigger mechanism are sufficient to reveal the matrix multiplication parameters. To the best of our knowledge, this is the first, auto-triggered attack that is performed on a public Cloud FPGA platform (i.e. AWS F1), and does not require a hard-wired trigger between the victim and attacker.

## V. Related Work and Discussion

### A. Related Work

Previous research has explored information leakage through thermal, cross-talk, and power side-channels [24]–[28]. These works, however, have presented very specific attacks on specific accelerators or modules. Our work takes a more generic approach. We focus on three distinct accelerators implemented in Cloud FPGAs.

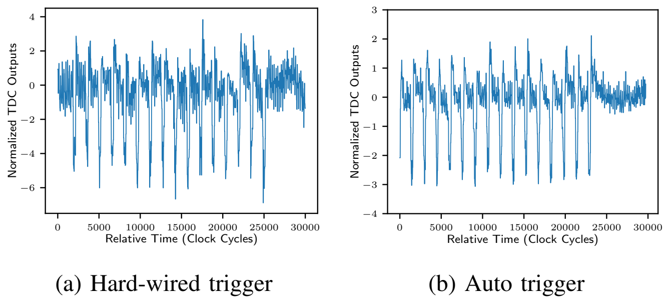


Fig. 9: The TDC measurements collected using hard-wired trigger and auto trigger.

Our work significantly expands upon recent side channel attacks in multi-tenant FPGAs. Boutros et al. [29] demonstrated that model redundancy and FPGA timing margins make ML fault injection difficult. Moini et al. [7] extracted machine learning (ML) input data by sensing FPGA voltage fluctuations with TDCs. This work used a simple custom-made neural network with binary parameters, facilitating input extraction. Hua et al. [30] determined the ML algorithm model parameters by observing off-chip accesses to DRAM. This approach requires tight control of algorithm implementation and bus-level observability. The previous approaches were either executed on local FPGAs and not on commercial cloud platforms, or triggered using hard wires to determine when the side-channel attack should start.

### B. Discussion

*1) Security Defenses:* Based on our evaluation, it is possible to use power-based voltage sensors to monitor on-FPGA voltage traces for an extended time. This monitoring provides an overview of the FPGA hardware execution process. We also demonstrate an automatic triggering mechanism. To defend against such attacks, an active fence circuit to generate power draw [31] could be added to confuse the attacker. In particular, since the two big voltage drops are induced by the start of SH-CL communication and accelerator computation, the active fence circuit should generate random power draws that are as deep as the drops we observe in normal kernels, thus confusing attackers about when to trigger their data capture.

*2) Performance Observations:* Voltage monitoring on Cloud FPGAs is challenging. Built-in functions for live power consumption metrics provided by cloud vendors, i.e. `fpga-describe-local-image` in AWS F1, are inadequate as they are coarse-grained measurements that are refreshed every few seconds by design. Our work offers a practical runtime performance observation tool since CTRK has the ability to evaluate and monitor custom accelerators in Cloud FPGA settings. With CTRK, the voltage traces for the execution of custom accelerators can be captured from the start of host computer transactions to the end of accelerator computation. As shown in Figures 6 and 7, the preparation time required before a kernel starts can be compared using three accelerators with different input sizes. The results suggest that logic code designs and data structures make a significant impact on overall performance. The preparation process is closely related to the characteristics of accelerator designs and the Vitis `link` mechanism. This optimization investigation will be addressed in future work.

## VI. Conclusion

This paper provided a power side channel security analysis of three existing accelerators developed in Xilinx Vitis tools, and deployed on the AWS F1 Cloud FPGAs that have more stable power and are more difficult to perform attacks compared to local FPGAs. We presented a TDC-based extended power trace collection mechanism that captures power traces for the duration of the machine learning inference process of the target co-tenant victim. A practical, automatically triggered remote power attack was also presented, which allowed for triggering the power trace collection without need for a hard-wired connection between the victim and attacker. Our results showed that malicious users can create voltage sensors as RTL kernels within state-of-the-art compute environments on AWS F1, and collect power traces to steal information, such as matrix shapes used in the machine learning models executed by the co-tenant victim. In addition, our run-time performance observations provided an new evaluation of the three machine learning accelerators and showed varying performance overheads based on the design of the accelerators.

## References

[1] Microsoft Research , "Project Catapult," https://www.microsoft.com/en-us/research/project/project-catapult/, 2021, Accessed: 2022-03-20.

[2] Amazon Web Services, "Aws powers f1 insights," https://aws.amazon.com/f1/, Accessed: 2022-03-20.

[3] ——, "Amazon EC2 F1 Instances," https://aws.amazon.com/ec2/instance-types/f1/, Accessed: 2022-03-20.

[4] Alibaba Cloud, "Elastic compute service: Instance type families," https://www.alibabacloud.com/help/doc-detail/25378.html, 2019, Accessed: 2022-03-20.

[5] V. Kathail, "Xilinx Vitis unified software platform," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020, pp. 173–174.

[6] Z. István, G. Alonso, and A. Singla, "Providing multi-tenant services with FPGAs: Case study on a key-value store," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 119–1195.

[7] S. Moini, S. Tian, J. Szefer, D. Holcomb, and R. Tessier, "Remote power side-channel attacks on BNN accelerators in FPGAs," in *Design, Automation and Test in Europe Conference*, ser. DATE, February 2021.

[8] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-Dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant FPGAs," in *Usenix Security Conference*, 2021.

[9] Y. Zhang, R. Yasaei, H. Chen, Z. Li, and M. A. A. Faruque, "Stealing neural network structure through remote FPGA side-channel analysis," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 16, pp. 4377–4388, Aug. 2021.

[10] Amazon Web Services, "Official repository of the AWS EC2 FPGA hardware and software development kit," https://github.com/aws/aws-fpga, 2019, Accessed: 2022-03-20.

[11] O. Knodel, P. R. Genssler, F. Erxleben, and R. G. Spallek, "FPGAs and the cloud–an endless tale of virtualization, elasticity and efficiency," *International Journal on Advances in Systems and Measurements*, vol. 11, no. 3-4, pp. 230–249, 2018.

[12] Xilinx Vitis, "Vitis Example Systolic Array," https://github.com/Xilinx/Vitis_Accel_Examples/tree/master/cpp_kernels/systolic_array, 2020, Accessed: 2022-03-20.

[13] ——, "Vitis example vector addition," https://github.com/Xilinx/Vitis_Accel_Examples/tree/f72dff9eea45a76e9ee0713774589624e2b52c9f/cpp_kernels/simple_vadd, 2020, Accessed: 2022-03-20.

[14] T. Moreau, T. Chen, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "VTA: an open hardware-software stack for deep learning," *arXiv preprint arXiv:1807.04188*, 2018.

[15] T. La, K. Pham, J. Powell, and D. Koch, "Denial-of-service on FPGA-based cloud infrastructure-attack and defense," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2021, no. 3, pp. 441–464, 2021.

[16] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2019.

[17] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier wires: Exploiting FPGA long wires for covert- and side-channel attacks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–29, Sep. 2019.

[18] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Power side-channel attacks on BNN accelerators in remote FPGAs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 357–370, 2021.

[19] T. M. La, K. Matas, N. Grunchevski, K. D. Pham, and D. Koch, "FPGADefender: Malicious self-oscillator scanning for Xilinx UltraScale+ FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 3, pp. 1–31, 2020.

[20] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "Measuring long wire leakage with ring oscillators in cloud FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2019.

[21] Amazon Web Services, "AWS shell interface specification," https://github.com/aws/aws-fpga/blob/master/hdk/docs/AWS_Shell_Interface_Specification.md, 2022, Accessed: 2022-03-20.

[22] ——, "AWS FPGA custom logic ports," https://github.com/aws/aws-fpga/blob/master/hdk/common/shell_v04261818/design/interfaces/cl_ports.vh, 2022, Accessed: 2022-03-20.

[23] 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)*. IEEE, 2021, pp. 242–246.

[24] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1007–1010.

[25] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1345–1362.

[26] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[27] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium (USENIX Security)*, 2016, pp. 601–618.

[28] 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)*. IEEE, 2018, pp. 1111–1116.

[29] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from Hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs," in *International Conference on Field-Programmable Technology (FPT)*, 2020.

[30] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[31] J. Krautter, D. R. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Active fences against voltage-based side channels in multi-tenant fpgas," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–8.