# Classifying Computations on Multi-Tenant FPGAs

Mustafa Gobulukoglu
*University of California, San Diego*
La Jolla, United States
mgobuluk@eng.ucsd.edu

Colin Drewes
*University of California, San Diego*
La Jolla, United States
cdrewes@ucsd.edu

William Hunter
*Georgia Tech Research Institute*
Georgia, United States
bill.hunter@gtri.gatech.edu

Ryan Kastner
*University of California, San Diego*
La Jolla, United States
kastner@ucsd.edu

Dustin Richmond
*University of Washington*
Seattle, United States
dustinar@uw.edu

***Abstract*—Modern data centers leverage large FPGAs to provide low latency, high throughput, and low energy computation. FPGA multi-tenancy is an attractive option to maximize utilization, yet it opens the door to new security threats. In this work, we develop a remote classification pipeline that targets the confidentiality of multi-tenant cloud FPGA environments. We utilize an in-fabric voltage sensor that measures subtle changes in the power distribution network caused by co-located computations. The sensor measurements are given to a classification pipeline that is able to deduce information about co-located applications including the type of computation and its implementation. We study the importance of the trace length and other aspects that affect classification accuracy. Our results show that we can determine if another co-tenant is present with 96% accuracy. We can classify with 98% accuracy whether a power waster circuit is operating. Furthermore, we are able to determine if a cryptographic operation is occuring, differentiate between different cryptographic algorithms (AES and PRESENT) and microarchitectural implementations (Microblaze, ORCA, and PicoRV32).**

## I. Introduction

FPGAs are being deployed in data centers as compute offload engines for neural networks [1], genome sequencing [2], secure database transactions [3], networking [4], and homomorphic encryption [5]. These applications handle sensitive information, and have strict requirements on the confidentiality of their data and associated computations.

Cloud servers provide large, expensive FPGAs to maximize the processing power available to customers. Existing systems employ an "all or nothing" approach where users are allocated the entire FPGA. This leads to under-utilization when customers cannot fill the available space. FPGA virtualization has been proposed to maximize utilization and reduce cost [6].

Unfortunately, FPGA virtualization opens the door for new security problems. A recent class of remote FPGA attacks use voltage fluctuation sensors implemented on the programmable logic to sense changes in the local supply voltage. These minute voltage fluctuations provide information about other computations that use the same power distribution network. These voltage fluctuation sensors can use the power network as a covert channel [7], [8] or as a side-channel to extract cryptographic keys of co-located encryption cores using common power analysis techniques [7], [8]. These attacks rely on co-locating attacker hardware with a victim's hardware.

This work develops techniques to classify co-located computations using a voltage fluctuation sensor. This includes determining if there is another co-tenant, if that co-tenant is performing encryption, whether the co-tenant is utilizing a soft processor, and other questions that violate the confidentiality of the co-tenant. This a necessary precursor for performing attacks in a virtualized FPGA environment, where an attacker must identify a co-located core before performing an attack, or defending against them, where a provider recognizes malicious cores and terminates service.

We consider the scenario where the sensor is given a logically isolated region of the FPGA (as is common in cloud FPGA environments). We make no assumptions about design rule checks, and only assume that the attacker can use relative-location constraints. We study the ability for the attacker to use their allocated programmable logic to infer behaviors of other tenants on the same FPGA. We implement a voltage fluctuation sensor that measures the compute environment to produce a signal, and use those readings to classify co-tenant computations.

We demonstrate the ability to classify different types of computation cores that are logically isolated on the same FPGA system. We show that although the voltage fluctuation sensor is logically isolated from each victim tenant, features encoded as images from sensor readings are sufficient to train an image classifier. Our classifier can accurately assess if there is a co-tenant, determine if the co-tenant is executing certain types of applications, and derive characteristics of the implementation of that application. We study different aspects of the classification pipeline including signal length, signal conditioning techniques, and image transformations. We make the following contributions:

- Introduce the ability to classify co-tenant computations using a voltage fluctuation sensor.
- Study the effects that signal conditioning, signal length, and other parameters have on the classification problem.
- Quantify the amount of time required to accurately characterize co-tenant computation.

The remainder of the paper is organized as follows: Section II presents the threat model that we assume for our work. Section III introduces the Time-to-Digital Converter

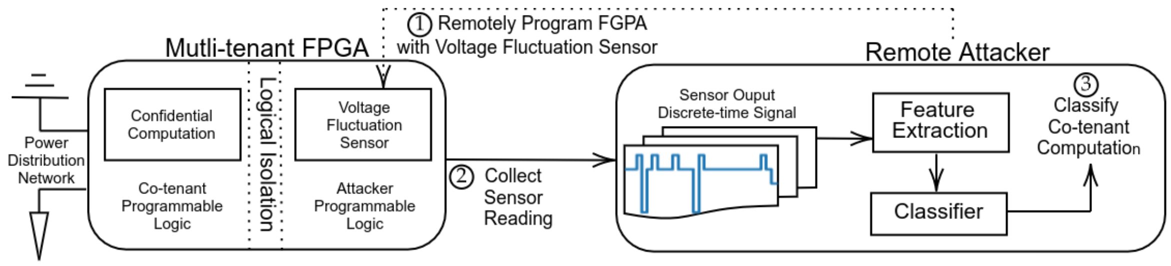

① Remotely Program FGPA with Voltage Fluctuation Sensor
Mutli-tenant FPGA
Power Distribution Network
Confidential Computation
Co-tenant Programmable Logic
Logical Isolation
Voltage Fluctuation Sensor
Attacker Programmable Logic
② Collect Sensor Reading
Remote Attacker
Sensor Ouput Discrete-time Signal
Feature Extraction
Classifier
③ Classify Co-tenant Computation

Fig. 1: Remote TDC Sensor Threat Model – Step ①: An attacker is given access to a remote multi-tenant FPGA and programs it with a voltage fluctuation sensor. Step ②: The sensor readings are gathered and sent to the attacker for analysis. Step ③: The attacker classifies the sensor readings to characterize aspects of the co-tenant's computation.

sensor. Section IV describes our classifier pipeline and how we make inferences. Section V presents experimental results that provide insight into the power side channel information of power traces, a CNN image classifier, several victim multi-tenant computations, and the accuracy of the network in classifying the described multi-tenant computations. Section VI describes related work. Finally, we conclude and summarize in Section VII.

## II. Threat Model

Figure 1 describes the proposed threat model. The attacker is provided access to a multi-tenant FPGA and co-locates with a victim tenant. The attack is performed completely remotely; it does not require physical access to the FPGA. The attacker develops a design with a voltage fluctuation sensor and programs it on the multi-tenant FPGA. We assume the system performs physical and logical separation of the tenants [9] and the attacker is restricted to system defined interfaces, e.g., those provided by a shell. The attacker gathers the sensor readings and classifies them to determine a characteristic of the co-located computation.

The attacker is a malicious adversary with the intention of classifying the computations of other tenants utilizing the same FPGA. That attacker is given an area of the programmable logic and can implement a voltage fluctuation sensor. Our voltage fluctuation sensor is a variant of a time to digital (TDC) sensor [10]. We assume that our sensor will pass any bitstream analysis techniques put in place to detect potential remote attacks [11]. Our sensor does not have timing path violations or combinational cycles and thus is potentially stealthier than RO sensors, but we have not experimentally validated this.

We do not make any assumptions about where the sensor is placed, e.g., the victim computation does not need to have one of its wires running through it [12]. However, the sensors are more sensitive to computations that are spatially closer [13], and thus, the proximity of the location of the target computation will effect our ability to classify information.

Our experiments only consider attacks on computation co-located on the same programmable logic, However, we note that similar attacks have been shown from the FPGA to a CPU on the same die [7], across dies on a 2.5D integrated package [14], and across chips on the same board [8]. Thus, we believe that our general techniques could be used for these other scenarios to infer more information about the overall system (granted they will likely not be as effective due to a reduction in sensor signal to noise).

We assume that the attacker has access to a set of representative IP cores that they intend to characterize. These IPs do not necessarily need to be the exact IP that the attacker will encounter, but they need to have similar computational characteristics. The attacker executes the representative IPs alongside a voltage fluctuation sensor in a controlled environment to build a corpus of labeled training data for each type of computation being targeted. After the training phase, the attacker no longer requires access to the IP cores and is ready to remotely launch the attack.

The attacker remotely uploads a bitstream containing a voltage fluctuation sensor. The sensor readings are captured and sent back to the attacker for analysis. The gathered sensor readings are transformed into an image and feed into the pre-trained neural network which classifies the computation (as described in Section IV).

## III. Time-to-Digital Converter

A Time-to-Digital Converter (TDC) is a circuit that senses transient voltage variations on a power distribution network (PDN) [10]. The power supply voltage fluctuations are due (in part) to power consumption from other computations using the same power distribution network. TDC sensors can be implemented using a series of linear delay elements placed in the FPGA logic. Most implementations use the fast-carry chain structure provided by an FPGA. The regular routing and circuit characteristics create a linear propagation delay medium that eliminates routing path inconsistencies that lead to non-linear delays [15]. The speed of propagation is a function of the voltage of the power distribution network.

The output of the TDC is a function of that propagation speed; it is a time-varying signal that can be analyzed to infer information about the system. This propagation speed is captured as a discrete-time signal which is referred to as a trace.

## IV. Classifier Pipeline

Traces from the TDC sensor are provided to a classification pipeline that infers the co-tenant computation using voltage fluctuations caused by that computation. First, the raw sensor data from the TDC is processed into numeric traces. Next, the

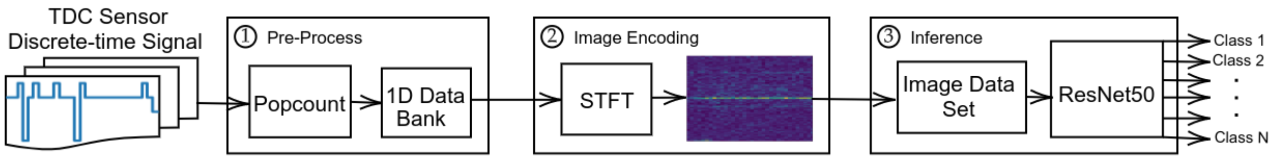


Fig. 2: Three-Stage Classification Pipeline – In the preprocessing stage the output of the TDC is transformed to produce processed traces. These signals are then encoded into spectrograms using a Short-Term Fourier Transform (STFT) function. In the classification stage, a ResNet50 classifier determines which application the signal represents.

traces are encoded into an image using frequency analysis. This image is then used for training and classification. The classifier pipeline is shown in Figure 2.

### A. Pre-Processing

The raw output sequence of the TDC sensor is a series of binary strings. This output is then transformed into a numeric signal through a “population count,” or binary hamming weight. Popcount can also be performed efficiently, which would be especially important if the attacker aimed to perform real-time classification (e.g., by implementing the classification pipeline on the remote FPGA).

### B. Short-Term Fourier Transform Encoding

Next, we encode the processed traces into images using a Short-Term Fourier Transform. The output image represents the magnitudes of the frequencies that are present for a given portion of the input signal. An STFT uses a window to take segments of a given input signal and performs Fourier transforms over each window to produce an image that encodes the frequency domain of the signal at a given time. These images are representations of another tenant’s computations on the FPGA which we wish to classify. An example STFT is shown in Figure 2.

### C. Inference

The final stage of the classification pipeline applies an inference model to the STFT image to predict which application the image is derived from. We use ResNet50 to infer the computations of co-tenants by classifying an STFT image into different classes. The classes correspond to some characteristic of the co-tenant computation. Our experiments aim to classify each STFT image into one of nine applications as described in Section V.

ResNet50 is a convolutional neural network (CNN) that uses a mix of convolutional, max-pooling, dropout, and fully-connected layers (50 in total). The convolutional layers form associations between image features and output labels. The dropout layers prevent over-fitting during the training phase of the network by randomly setting the edges of hidden nodes to 0 during the training phase. The output layer employs a Softmax function to select between the output spaces. The ResNet architecture uses residual learning where extra connections from shallower layers bypass some middle layers and connect to layers further down the network. This has been shown to increase the efficiency and accuracy of CNNs.

## V. Evaluation

This section describes experimental procedures that demonstrate the ability to classify co-tenant FPGA computations using a remotely uploaded voltage sensor. We evaluate different aspects of our TDC sensor and classification pipeline. This includes techniques to effectively extract information from the sensor, and trade-offs in the three stages of the classification pipeline. Our evaluation collects traces for each target computation on a remote system. We program the system with 9 unique co-tenant computations, gather training data, and train a classification pipeline that can accurately select the computation being computed by the co-tenant.

### A. Sensor Parameters

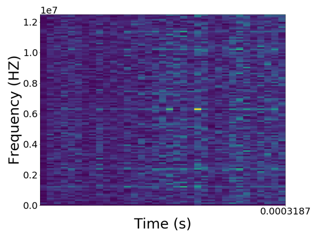
The TDC delay line sensor is implemented on a Xilinx ZYNQ XC7Z020-1CLG400C. The sensor employs a 64-bit delay line capture window. The delay elements are implemented as fast-carry adders which provide uniform and linear delay through the columns of an FPGA. Both the launch and capture clock domains operate at 100 MHz and are generated by an MMCM. The sampling rate is 25 MHz.

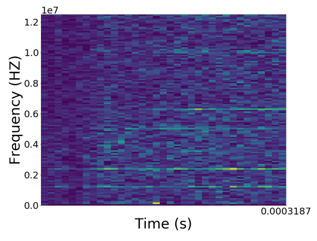
### B. Target Computations

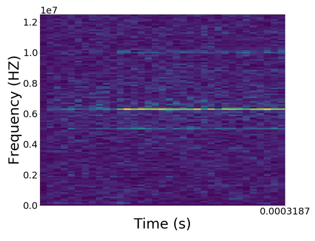
To study what characteristics of co-tenant computation can be classified, we deploy nine unique computations. One of these is a baseline, i.e., only the sensor is operating. Another is a power waster, which emulates a potential fault attack. We also look at many different implementations of cryptographic operations, including two different algorithms (AES and PRESENT) running on a variety of different architectural implementations (custom IP core, Microblaze, ORCA, and PicoRV). We describe each of these applications in more detail.

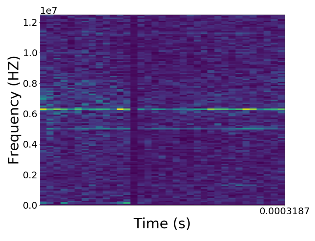
*1) Baseline:* The primary goal of the baseline is to model the lack of another co-tenant. The baseline only contains the voltage fluctuation sensor and associated data collection logic. This mimics a scenario where only the attacker is present on the FPGA.

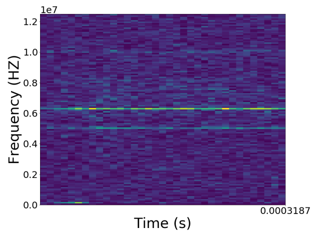
*2) Power Wasters:* A power waster is a malicious circuit with the sole purpose of aggressively consuming power. This design implements banks of ring oscillators which can be turned on to consume significant amounts of power. Such a circuit can cause voltage disruptions in the power distribution network and can be used as a covert channel or to induce faults. Classifying these circuits is important for removing these malicious computations from a multi-tenant hardware platform.

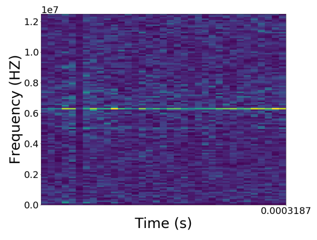

(a) Power Wasters

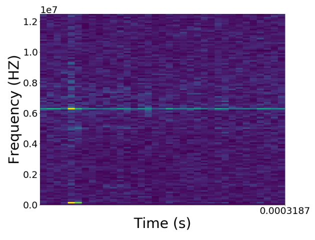

(b) AES


(c) ORCA-AES


(d) ORCA-PRESENT


(e) MB-AES


(f) MB-PRESENT


(g) PICO-AES

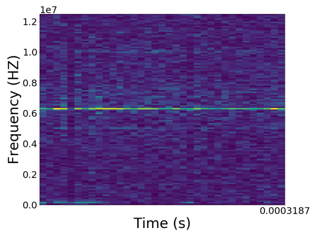

(h) PICO-PRESENT

Fig. 3: Image encodings from the three-stage classification pipeline. Examples of the STFTs output from the Image Encoding stage of the three-stage classification pipeline are provided for each co-tenant computation considered. These examples all utilize 16K samples for each given computation.

*3) Cryptographic Cores:* We study seven different implementation of crytographic computations. These include two algorithms (AES, PRESENT) and four different architectural implementations (custom IP core and as software running on ORCA, MicroBlaze, and PicoRV soft processors). This provides seven more unique applications (AES, ORCA-AES, ORCA-PRESENT, MB-AES, MB-PRESENT, PICO-AES, PICO-PRESENT).

### C. Data Collection and Training

*1) Target Computations:* Power traces gathered from the delay line sensor are used in the classification of 3 different hard IP cores and 6 different soft processor applications as described in the previous section. Each of these is clocked at 5 MHz and logically isolated from the voltage sensor on the same FGPA. Bitstreams containing both the delay line sensor and the victim IP are uploaded through a remote connection to the FPGA.

*2) Data:* 250 traces are remotely gathered from each of the 9 target computations for the duration of their operation for a total of 2250 traces. The traces for each target are randomly segmented into lengths of $N = 32, 64, 128, 256, 512, 1K, 2K, 4K, 8K$, and $16K$ clock cycles. This is done by starting at an arbitrary sample in the trace and collecting the following $N$ samples to encode for a given segment length. These segmented traces along with the unsegmented, whole traces are then encoded as STFTs. Figure 3 shows example STFTs produced from the processed traces. This is done for each segment length yielding a total of 11 data sets. A separate classifier is created for each data set resulting in 11 classifiers of 45000 images each, from which 9000 images are selected at random and 90% of these are used for training and 10% for testing.

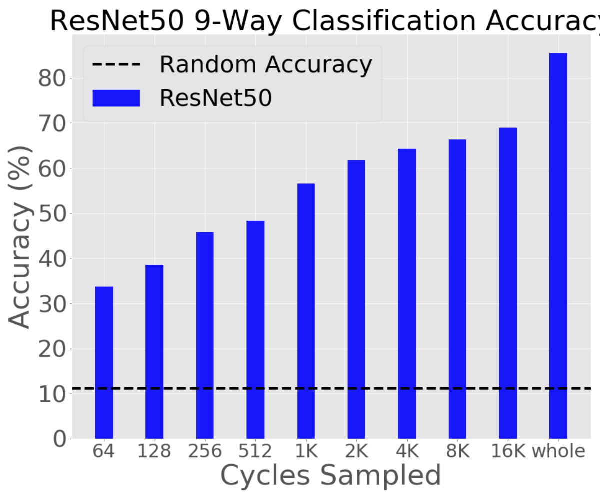


Fig. 4: The 9-way classification accuracies resulting from sampling a co-tenant's computations for various clock cycles. The clock cycles sampled range between 64 cycles to the entirety of the co-tenant's computation. Sampling for a longer duration yields better accuracy from the three-stage classification pipeline.

*3) ResNet50:* ResNet50 is implemented using Keras and is pre-trained on ImageNet. This allows for more efficient training because the network weights are closer to their global optimums in minimizing the network loss function. The network hyper-parameters are tuned using 10% of the training data as a validation set. From here, the same hyper-parameters are used for the rest of the 10 networks. Each classifier is trained for 50 epochs on its respective data set.

### D. Results

The overall classification results are presented in Figure 4. Our results show that a larger number of samples increases the accuracy. This is because more characteristic frequency

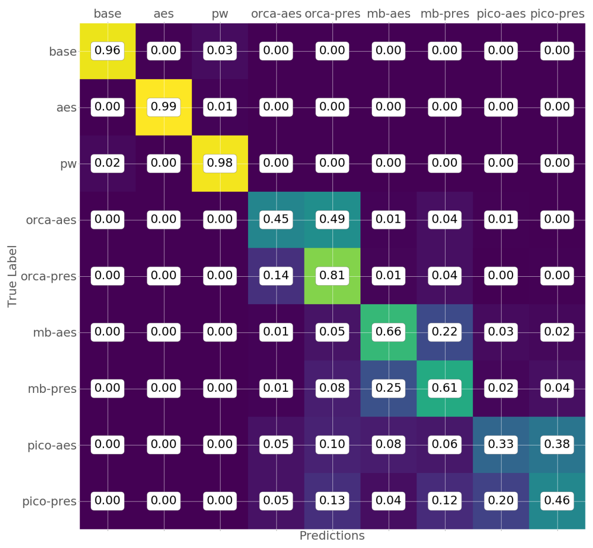


Fig. 5: The confusion matrix of a 16K STFT classifier. This matrix shows the classification accuracy for each of the 9 computations that are considered in this paper. It also provides insight into what misclassifications are commonly made for a given subset of the computations. The values for a given cell provide the percentage of predictions that a computation is classified as in decimal ranging from 0 to 1.0.

information relative to the computations of the co-tenant are captured when sampling for a longer duration. The best accuracy is achieved by using the whole STFTs which encode traces that sample for the entire duration of the co-tenant's computation. We achieve a 9-way classification accuracy of about 91% using these whole STFTs data set. However, it is important to note that the whole STFTs may introduce some bias to the classifier due to the fact that these computations vary in the number of execution cycles. So, the classifier may be internally making associations between the length of each computation and the feature space of the STFTs. Furthermore, this would require the attacker to know the duration of each computation a priori. Thus, we do not feel that it is realistic or consider the whole traces in our analysis.

The shortest segment length which does not contain this bias is the 16K cycle STFT data sets as this is the duration of the shortest computation. The 16K cycle STFT classification yields a 9-way classification accuracy of about 70%. The resulting confusion matrix from the 16K cycle STFT classification is presented in Figure 5. This classifier performs very well in distinguishing between various types of computations which consist of the baseline, AES-128 core, 1024 power waster chain, and any of the soft processor applications in a 4-way classification with an accuracy of about 97.6%. This is because the power signatures between these computations are relatively distinct from one another. In the baseline, nothing aside from the sensor on board the FPGA is consuming power in the PL fabric. The AES-128 core is a synchronous core with well defined periodic rounds, the 1024 power waster chain is an asynchronous core, and the soft processors are emulating an instruction set architecture which is distinct from any of the other hard IPs. None of the soft-processor applications are misclassified as any of the hard IP cores and the only source of confusion in this 4-way classification stems from the misclassification of the asynchronous power wasters as the baseline with a maximum error of 3%.

The 16K cycle STFT classifier also performs quite well in distinguishing between the three different soft processors. A given soft processor is mistaken for another at most 13% of the time. However, the difficulty comes in distinguishing what application is running on a given soft processor. The greatest source of confusion comes from AES running on the ORCA soft-processor, where this application is misclassified as PRESENT running on the ORCA about 49% of the time. These results argue for using soft processors to obfuscate the power side channel information leaked by a particular application. While an attacker may be able to classify the soft processor used by a co-tenant, it is very difficult to infer what application the co-tenant is running.

These results show that it is possible to classify the computations of co-tenants on a multi-tenant FPGA. Even by sampling only 32 cycles which is less than 0.2% of the execution time of any of the targets and less than even 0.0064% of the execution time for some of the soft processor applications, it is still possible to perform this 9-way classification with an accuracy 3x better than a random inference by using our proposed three-stage classification pipeline.

## VI. Related Work

### A. Remote Fault Injection Attacks

A co-tenant can exploit the PDN to cause denial of service or fault attacks by uploading a malicious circuit with the intention of causing voltage disruption in other tenant applications. These attacks induce large PDN voltage fluctuations by turning on circuits that consume substantial power, e.g., banks of Ring Oscillators, which cause a significant voltage disruption on the PDN and inject faults into other tenant cores. These faults occur due to the victim core registers' inability meet the setup and hold times required for proper operation due to the lack of power through the PDN. This can be used to perform fault attacks on security sensitive IPs [16], [17].

### B. PDN Covert and Side Channels

Several projects have demonstrated the ability to extract covert and side channel information from the PDN. Covert channels have been measured between cores on an FPGA [12], [18], between FPGA dies in a 2.5D package [14], and across chips on the same board [19]. Power analysis attacks have been demonstrated on co-located encryption cores [20] and launching a timing side channel on a encryption running on the ARM processor co-located on the same chip as the FPGA [7].

We show that it is possible to learn more general characteristics of the co-tenant computation. Our classification pipeline can answer questions like: is there a co-tenant? Is the co-tenant performing encryption? What sort of encryption? And how is the computation implemented? These are key questions that would be necessary to launch such a remote power analysis attack and generally violate the confidentiality of co-located tenants.

### C. Mitigation Strategies

It is possible to logically and physically isolate co-tenents on the FPGA programmable logic [9]. These mitigate some attacks that require close physical access, e.g., [12]. But many remote attacks are still possible as they do not have such constraints on sensor placement [7], [20]. Active fences [21] attempts to take this one step further by surrounding the co-tenant IP core with ROs. This induces noise into the local PDN, making it harder to extract the signal due to the computation of the core.

Krautter et al. [11] describe techniques that check the bitstream for circuit structures that resemble those that can be used to perform active fault or passive side channel attacks. They focus on structures like ring oscillators, those that induce timing violations, data to clock paths, and high fanouts. They argue these are indicative of RO and TDC circuits used in these attacks.

Yazdanshenas et al. [22] considers defenses against long wire crosstalk attacks in the context of multi-tenant execution. Their approach relies on encrypting the data before transmitting outside of isolated region. While this helps defend against some attacks [12], [23], it is still susceptible to attacks that do not require such proximity to the susceptible computation, e.g., [7], [8].

## VII. Conclusion

In this paper, we have proposed a three-stage classification pipeline for the remote classification of co-located applications. By leveraging a TDC sensor to collect power side channel information, it is possible to classify computations that are running on a multi-tenant hardware platform. The ability to classify computations provides a necessary precursor to many remote power side channel and power distribution attacks, such as those described by [8], [20]. By identifying the existence of security sensitive computations like encryption cores on a multi-tenant hardware platform, an attacker will have the capacity to know when it is possible to perform these attacks over the PDN.

## References

[1] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, *et al.*, "A configurable cloud-scale dnn processor for real-time ai," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–14, IEEE, 2018.

[2] Y.-T. Chen, J. Cong, Z. Fang, J. Lei, and P. Wei, "When spark meets fpgas: A case study for next-generation DNA sequencing acceleration," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[3] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, and R. Ramamurthy, "Transaction processing on confidential data using cipherbase," in *2015 IEEE 31st International Conference on Data Engineering*, pp. 435–446, IEEE, 2015.

[4] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 13–24, IEEE, 2014.

[5] T. Pöppelmann, M. Naehrig, A. Putnam, and A. Macias, "Accelerating homomorphic evaluation on reconfigurable hardware," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 143–163, Springer, 2015.

[6] Y. Zha and J. Li, "Virtualizing fpgas in the cloud," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 845–858, 2020.

[7] M. Zhao and G. E. Suh, "Fpga-based remote power side-channel attacks," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 229–244, IEEE, 2018.

[8] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "Remote inter-chip power analysis side-channel attacks at board-level," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, IEEE, 2018.

[9] 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 (SP'07)*, pp. 281–295, IEEE, 2007.

[10] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in fpgas," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 101–104, 2013.

[11] J. Krautter, D. R. Gnad, and M. B. Tahoori, "Mitigating electrical-level attacks towards secure multi-tenant fpgas in the cloud," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–26, 2019.

[12] 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.

[13] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user fpga environments," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 194–201, IEEE, 2019.

[14] 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, IEEE, 2019.

[15] A. Aloisio, R. Lomoro, S. Loffredo, V. Izzo, P. Branchini, R. Cicalese, and R. Giordano, "High-resolution time-to-digital converter in field programmable gate array," 2008.

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

[17] J. Krautter, D. R. Gnad, and M. B. Tahoori, "Fpgahammer: Remote voltage fault attacks on shared fpgas, suitable for dfa on aes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 44–68, 2018.

[18] D. R. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels in multi-tenant fpgas.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1394, 2019.

[19] I. Giechaskiel, K. Rasmussen, and J. Szefer, "C3APSULe: Cross-fpga covert-channel attacks through power supply unit leakage," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.

[20] 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, IEEE, 2018.

[21] J. Krautter, D. R. Gnad, F. Schellenberg, A. Moradi, and M. B. Tahoori, "Active fences against voltage-based side channels in multi-tenant fpgas.," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1152, 2019.

[22] S. Yazdanshenas and V. Betz, "The costs of confidentiality in virtualized fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2272–2283, 2019.

[23] 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, IEEE, 2018.