Gotcha! I Know What You Are Doing on the FPGA Cloud: Fingerprinting Co-Located Cloud FPGA Accelerators via Measuring Communication Links

Chongzhou Fang
University of California, Davis
Davis, CA, USA
czfang@ucdavis.edu

Ning Miao
University of California, Davis
Davis, CA, USA
nmiao@ucdavis.edu

Han Wang
Temple University
Philadelphia, PA, USA
han.wang.hw@temple.edu

Jiacheng Zhou
University of California, Davis
Davis, CA, USA
jczhou@ucdavis.edu

Tyler Sheaves
University of California, Davis
Davis, CA, USA
tsheaves@ucdavis.edu

John M. Emmert
University of Cincinnati
Cincinnati, OH, USA
john.emmert@uc.edu

Avesta Sasan
University of California, Davis
Davis, CA, USA
asasan@ucdavis.edu

Houman Homayoun
University of California, Davis
Davis, CA, USA
hhomayoun@ucdavis.edu

ABSTRACT

In recent decades, due to the emerging requirements of computation acceleration, cloud FPGAs have become popular in public clouds. Major cloud service providers, e.g. AWS and Microsoft Azure have provided FPGA computing resources in their infrastructure and have enabled users to design and deploy their own accelerators on these FPGAs. Multi-tenancy FPGAs, where multiple users can share the same FPGA fabric with certain types of isolation to improve resource efficiency, have already been proved feasible. However, this also raises security concerns. Various types of side-channel attacks targeting multi-tenancy FPGAs have been proposed and validated. The awareness of security vulnerabilities in the cloud has motivated cloud providers to take action to enhance the security of their cloud environments.

In FPGA security research papers, researchers always perform attacks under the assumption that attackers successfully co-locate with victims and are aware of the existence of victims on the same FPGA board. However, the way to reach this point, i.e., how attackers secretly obtain information regarding accelerators on the same fabric, is constantly ignored despite the fact that it is non-trivial and important for attackers. In this paper, we present a novel fingerprinting attack to gain the types of co-located FPGA accelerators. We utilize a seemingly non-malicious benchmark accelerator to sniff the communication link and collect performance traces of the FPGA-host communication link. By analyzing these traces, we are able to achieve high classification accuracy for fingerprinting co-located accelerators, which proves that attackers can use our method to perform cloud FPGA accelerator fingerprinting with a

high success rate. As far as we know, this is the first paper targeting multi-tenant FPGA accelerator fingerprinting with the communication side-channel.

CCS CONCEPTS

• **Security and privacy → Side-channel analysis and countermeasures**.

KEYWORDS

Cloud FPGA, fingerprinting, communication links, machine learning

1 INTRODUCTION

In recent decades, cloud computing has gained great popularity due to its considerable computation power, storage capacity, and pay-as-you-go features. With public cloud services being used, cloud users do not need to set up and maintain their own infrastructure, which greatly reduces their costs. Also, infrastructure-as-a-service (IaaS) public cloud services that are open to public users enable lots of newly emerging applications that require massive computation power, e.g., simulation [32], deep learning [9], etc. The increasing computation demands incurred by artificial neural networks further motivate the inclusion of hardware accelerators including GPUs [49], FPGAs[35], ASICs [11]. Among the various hardware accelerators, CPU-FPGAs have become a prevalent heterogeneous architecture to perform computation-intensive workloads due to

their programming flexibility, energy efficiency, and high performance. An increasing number of cloud providers have launched commercialized FPGA-based products in the past five years, like AWS [7], Microsoft Azure [39], Alibaba Cloud [6]. To further maximize the utilization of FPGAs, multi-tenancy FPGA infrastructure in the cloud is potentially preferred by the commercial world, which can improve FPGA utilization efficiency by fitting multiple users' designs onto a single FPGA at the same time. It is widely investigated in academia and a promising infrastructure.

However, with notable benefits comes new security threats. There have been research works regarding security attacks on FPGAs, such as bitstream fault injection [50], hardware trojan [37], rowhammer attacks [55], etc. In recent years, remote attacks on cloud FPGAs have also emerged. Security problems are especially severe on multi-tenant FPGA clouds, where circuits from multiple users can be placed on the same FPGA. There are attacks such as power side-channel attacks on remote FPGAs [40, 47], fault attacks [5], etc. There are also other attacks targeting revealing information about cloud infrastructures [52, 53]. These attacks compromise the security of FPGA cloud users and cloud service providers, causing trust issues about FPGA clouds.

As far as we know, all of the existing attacks targeting cloud FPGA user applications demand the knowledge of the co-located victim FPGA circuits, which is non-trivial. Taking the fault attack proposed in [36] as an example, as a prototype of a remote fault injection attack, it is based on the assumption that the co-located victim FPGA circuit is an AES circuit. Similarly, Moini *et.al* [40] leverages the power side-channel traces to recover the MNIST inputs [38] based on the knowledge that the victim FPGA circuit is a binarized neural network (BNN) accelerator. Hence, such prior attacks do not fully present the vulnerabilities and risks in FPGAs cloud, which can lead to the negligence of security challenges and limit corresponding defence solutions.

In response, this research aims to explore the possibility of fingerprinting victim circuits with passive side-channel information from the communication link, i.e., Peripheral Component Interconnect Express (PCIe) in shared FPGAs. PCIe [43] is used to connect peripheral devices including FPGAs with host machines, and is open for user interaction. We present that stressing the shared communication link can help reveal the I/O patterns of victim circuits co-locating in the same FPGA board. The deduced knowledge of victim circuits can further enable prior proposed attacks [36].

To achieve this, we design a measurement accelerator (in the remainder of this paper, we refer to FPGA circuits deployed by a users as 'accelerators') to stress the PCIe and conduct read/write to host memory blocks in the FPGA cloud (Intel DevCloud [30] in this work). Then we measure the PCIe bandwidth of our measurement accelerator when different victim accelerators are running on the same FPGA. Once we collect the side-channel traces from PCIe, we leverage machine learning techniques to train a model to classify the victim accelerators. Furthermore, this work looks into the impact of the contention level from benchmarks on fingerprinting success rate. Lastly, we implement a prototype of the fingerprinting attack to infer the co-located victim FPGA circuit in cloud infrastructures for future research.

In summary, the contributions of this work are listed below:

- We present a new attack targeting multi-tenancy FPGA clouds, which can help attackers gain additional knowledge about applications in the FPGA cloud and aid further attack attempts.
- We implement a proof of concept attack accelerator as well as its host program, which is able to capture the unique communication fingerprints of co-located accelerators.
- Four classification algorithms are included to obtain comprehensively assessment of closed-world fingerprinting success rate, reaching as high as around 90% by random forest. We also evaluate our method in an open-world setting, where the success rate reaches around 80%.
- By validating the proposed attack method, we reveal a novel security vulnerability in communication links of heterogeneous computing systems and provide insights into possible enhancements of such basic hardware and software components.

The remainder of this paper is organized as follows. Basic background knowledge such as cloud FPGA and FPGA security is introduced in Section 2. The threat model containing our assumptions is provided in Section 3. Our attack method and its implementation details are shown in Section 4. Section 5 offers evaluation results. We provide discussion about several defense approaches and future works. in Section 6. Related works are reviewed in Section 7. Finally, we provide a conclusion in Section 8.

2 BACKGROUND

2.1 Cloud FPGA

Field programmable gate arrays (FPGAs) are integrated circuits that can be programmed after being manufactured. With its great computation power and reprogrammable feature, it is often used to host hardware circuits for custom applications, such as machine learning accelerators. Recently, FPGA resources are starting to be provisioned by cloud service providers. Cloud FPGAs usually operate in two modes: acceleration-as-a-service (AaaS) and FPGA-as-a-service (FaaS) [14]. In AaaS mode, FPGAs are pre-configured by the service provider and are offered to users to only accelerate specific computations. On the other hand, the FaaS mode provides users access to the whole FPGA fabric and enables users to program it remotely with greater flexibility. Recently, the concept of multi-tenancy FPGA clouds start to appear [14], which presents a utilization model where a single FPGA in the cloud can be shared by multiple users and can be accessed by these users at the same time.

Cloud FPGAs provide a convenient way for customers to access high-end FPGA resources remotely. Different cloud FPGA providers are offering different types of FPGA resources, e.g. Intel provides users access to Arria 10 FPGAs and Stratix 10 FPGAs on DevCloud [1], AWS provides access to Xilinx Virtex UltraScale+ FPGAs with their F1 instances [8], Alibaba Cloud provides access to Xilinx Kintex UltraScale FPGAs and Arria 10 FPGAs [6], etc.

2.2 Security Problems of Cloud FPGA

In multi-tenant FPGA clouds, it has been proposed that circuits from multiple users can be placed on the same FPGA, which makes FPGA resource utilization on the cloud more efficient. However, recent

Gotcha! I Know What You Are Doing on the FPGA Cloud

CCS '23, November 26–30, 2023, Copenhagen, Denmark

research works have shown that cloud FPGAs are vulnerable to various types of side-channel attacks in a multi-tenant setting. Once the security of these FPGA accelerators is compromised, sensitive data or secret keys they are processing can be revealed, which may lead to unwanted data leakage and potentially harm the profits of cloud providers. The following types of attacks are studied most extensively in literature:

2.2.1 Long-wire Side-Channel Attack. Long wires, one type of FPGA routing resources that are used to connect configurable logic blocks (CLBs), have been proved to be a source of side-channel information leakage. In [19], the authors find that when a long wire on FPGA is transmitting a logical 1, the delay of the nearby long wires is shorter than when it is transmitting a logical 0. Based on this phenomenon, the authors propose to measure the delay of long wires by connecting ring oscillators (ROs) to them. When the target long wire is transmitting a logical 1, the delay of the nearby long wires will decrease, which causes the frequency of the ROs to increase. By monitoring the frequency change of the ROs in a fixed time interval, the authors successfully recovered 99% of the bits that are being transmitted in the target long wire. Similarly, in [48] the authors recovered the secret key of an AES implementation using the long-wire side-channel attack.

2.2.2 Power Side-Channel Attack. In certain FPGA circuits (e.g. cryptographic circuits), power consumption may be influenced by data being processed in the circuits hence this information may be monitored and used to recover secrets, e.g. cryptographic keys. Normally, deploying power side-channel attacks requires physical access to the FPGA boards in order to assess the system's power usage. Although direct access to cloud FPGAs is not achievable, Zhao *et.al* [57] propose a power side-channel attack using FPGA as a power monitor. The authors created an RO-based on-chip power monitor and prove that the RO-based FPGA power monitor may be utilized for a power analysis attack on an RSA crypto module on the same FPGA. Furthermore, in [27], the authors propose a new design for the RO-based power sensor, which can measure the internal voltage in nanosecond scale. They are able to successfully retrieve the secret key of an AES encryption circuit using the power side-channel.

Power side-channel can also be used for accelerator fingerprinting, as shown in [25]. However, in this paper we will show that communication side-channel can be a better option, which has less stringent requirements for attackers and can achieve better performance.

2.2.3 PCIe Side-Channel Attack. PCIe contention side-channel has been utilized before to retrieve secret information from CPU-GPU systems [51]. In [52], the authors used PCIe contention to perform an attack on the AWS server. The authors observe that the difference in locations of PCIe slots in the PCIe topology can result in disparate latency and bandwidth. Based on this, they are able to detect the bandwidth change when different FPGAs in the same sever attempt simultaneous memory accesses to generate PCIe contention and successfully reverse-engineer the locality of different FPGAs in the same AWS server. However, unlike our work, [52] focuses on revealing infrastructure information instead of revealing information about applications on the same FPGA.

The existing attacks targeting the FPGA cloud presume the knowledge of co-located victim circuit is provided. While in fact this information is nearly impossible to be obtained directly. In response, our work focuses on inferring co-located FPGA-accelerated workloads using PCIe contention side-channel information. Previously proposed attacks will benefit from our attack method.

2.3 Intel FPGA Cloud Environment

DevCloud is a cloud platform managed by Intel [1] to support research and education about FPGAs, GPUs, AI acceleration, etc. In this paper, all the development is done on DevCloud. We choose DevCloud because it provides us access to high-end commercial FPGA devices including Arria 10 and Stratix 10 FPGAs on the cloud, and we can utilize various off-the-shelf toolchains, including high-level-synthesis (HLS) [13], OpenCL [3] and OneAPI [31].

Accelerators in Intel FPGAs are usually called accelerator functional units (AFUs), which are connected to an interface layer called FPGA interface unit (FIU). In Intel's host-FPGA systems, PCIe serves as the low-level hardware component, and communication is orchestrated by their core cache interface protocol (CCI-P) [2]. All these low-level protocol details can be agnostic to developers, enabling them to focus on the development of AFUs. Our attack method also does not rely on features of their low-level implementation, and we don't need to hack into these systems managed by FPGA cloud providers.

3 THREAT MODEL

In this paper, we investigate the potential of fingerprinting victim circuits in a multi-user FPGA cloud environment [47]. We follow certain assumptions that are used in previous works [5, 47]. Specifically, multiple circuits implemented by different users are placed together on the same FPGA that connects to the same host. How this can be achieved is similar to cloud instance co-location attacks [16, 17, 42]. There are no direct connections or communications between circuits placed by different users. However, the communication links between hosts and connected FPGAs (i.e., PCIe) are shared and communication modules and protocols on top of physical layer are the same. Our work aims to capture the security issue caused by the sharing of the communication link among multiple users.

The service providers are assumed to be benign and neutral, i.e., they will not attempt to modify user-uploaded circuits. All applications as well as their host programs are executed as provided. Since our attack accelerator will not perform any sensitive operations, it may require special detection mechanisms to defend. We also assume cloud service providers will not terminate our accelerators and host programs. Since the operations proposed in this paper only involve I/O operations that are seemingly normal and no sensitive operations on other users will be triggered, this can be a valid assumption.

We assume attackers and victims have the same privileges in the cloud, i.e., attackers do not have access to more features than victims do. In our settings, the attackers' goal is to obtain information about co-located user applications, which is an important but missing task in existing cloud FPGA side-channel attack works. We consider two different settings: (1) closed-world setting, where attackers can

limit the range of accelerators running in the cloud; (2) open-world setting, where accelerators unknown to the attacker are involved. In both cases, the attacker has access to a server and an FPGA connected in the same way as in a cloud FPGA server.

Benign users are not supposed to be aware of the existence of malicious attackers in the system and hence will not terminate their own accelerators after our attack accelerators are launched. We assume these victim accelerators are constantly running on the FPGA and processing continuous streams of inputs. This assumption is made solely for convenience, since our attack does not require any timing information regarding victim execution life time. These victim accelerators can be either operating on encrypted data or plain original data, but victims will interact with the host through I/O operations. We aim to show that the difference in I/O access patterns can be captured by our proposed attacker accelerator.

In summary, attacks in our target system can be concluded as follows:

(1) Non-malicious users submit and deploy their accelerators on the cloud, which will keep running for a relatively long period of time;
(2) Attackers submit their malicious accelerators whose goal is to collect performance trace information of communication links and perform classification tasks to determine the exact type of co-located accelerators.

4 METHOD AND IMPLEMENTATION

In this section, we introduce the design of the proposed fingerprinting attack in FPGA cloud, which consists of attack preparation and online fingerprinting as shown in Figure 1. The key idea of this work is to capture the execution fingerprints of FPGA circuits by launching a measurement accelerator to measure the bandwidth of communication links and deducing the running victim circuits with machine learning techniques. The whole workflow of our fingerprinting attack consists of several steps:

(1) Run victim accelerators locally with our proposed measurement circuit to collect data;
(2) Pre-process the collected I/O measurement of possible victim accelerators and train a machine learning-based classifier with the offline collected data set;
(3) Launch the previously used benchmark to the cloud as accelerators and collect I/O measurements;
(4) Pass online I/O traces to the trained classifier to obtain fingerprinting results.

4.1 Measuring Communication Performance

In this section, we will introduce the implementation details of the benchmark used to stress the shared communication link and monitor I/O bandwidth. The observation of the benchmark reflects the I/O patterns of co-located victim circuits, which can be further leveraged to reveal the type of victim and used for our proof-of-concept (PoC) fingerprinting attack in the FPGA cloud. We will implement our PoC benchmark accelerator as well as the master host program under OpenCL [41] framework. The benchmark consists of two parts: the master host program located in CPU which orchestrates the execution of accelerators, and accelerator circuits in FPGA which stress PCIe communication link.

4.1.1 Host Program Design. In our PoC benchmark, the host program is responsible for:

(1) Assign appropriate resources for the operation of accelerator kernels;
(2) Invoke and orchestrate accelerator kernels;
(3) Measure kernel performance using low-level function calls.

There are 3 design parameters in our host program: BUFFER_NUM, BUFFER_SIZE and REPEAT_NUM. The workflow of our host program is defined as follows. First, the host program will allocate BUFFER_NUM memory trunks of size BUFFER_SIZE. Then, these BUFFER_NUM memory trunks will be accessed by the FPGA accelerator in a pre-defined order. Each of the BUFFER_NUM memory trunks will be read and written by the accelerator, with traffic passing through the communication link. During the operation to a memory trunk, the time it takes to execute the kernel will be recorded using profiling APIs provided by OpenCL. This information will be further used for calculating the bandwidth of the communication link when all operations to a memory trunk are finished. The operations to a single memory trunk may be repeated for REPEAT_NUM times and averaged to cancel the effect of noise. Finally, the BUFFER_NUM measurement of bandwidth will be combined together to form a trace with length BUFFER_NUM. The pseudo-code for the host program is shown in Figure 2.

4.1.2 Measurement Accelerator Design. The measurement accelerator we use in this paper focuses on measuring the I/O bandwidth performance. Similar to previous works that target measuring PCIe performance [43] or stressing the PCIe connection [52], our measurement accelerator implementation follows a similar method and stresses the PCIe communication link via massive read and write communication. There is one design parameter called ACCESS_NUM that controls how much data is written to the host. The code of our benchmark accelerator implemented as an OpenCL kernel is shown in Figure 3.

First, our benchmark accelerator takes in an address pointer (dst). dst is defined as a pointer pointing to pre-allocated host memory. By doing so, we guarantee that our FPGA accelerator will be able to access legally allocated host memory and the generated traffic will pass the FPGA-host communication link. Our kernel then obtains an arbitrarily assigned index to access the host memory space. The exact index is not important in our implementation, and we only use the OpenCL API get_global_id() for convenience.

Second, our accelerator enters an execution loop where host memory is accessed multiple times via an array update operation. The same location (dst[id]) in host memory will be updated ACCESS_NUM times, where ACCESS_NUM is a design parameter of our benchmark accelerator. The operation listed in Figure 3 ensures that a certain amount of data is transferred and the compiler will not optimize out the operation since every time there will be a new value written to the host memory.

4.2 Data Processing

In the offline data collection phase (Step 1 of Figure 1), the attacker will create a co-location environment and run benchmarks together with potential victim accelerators to collect a performance trace data set. The collected data traces will be normalized and organized
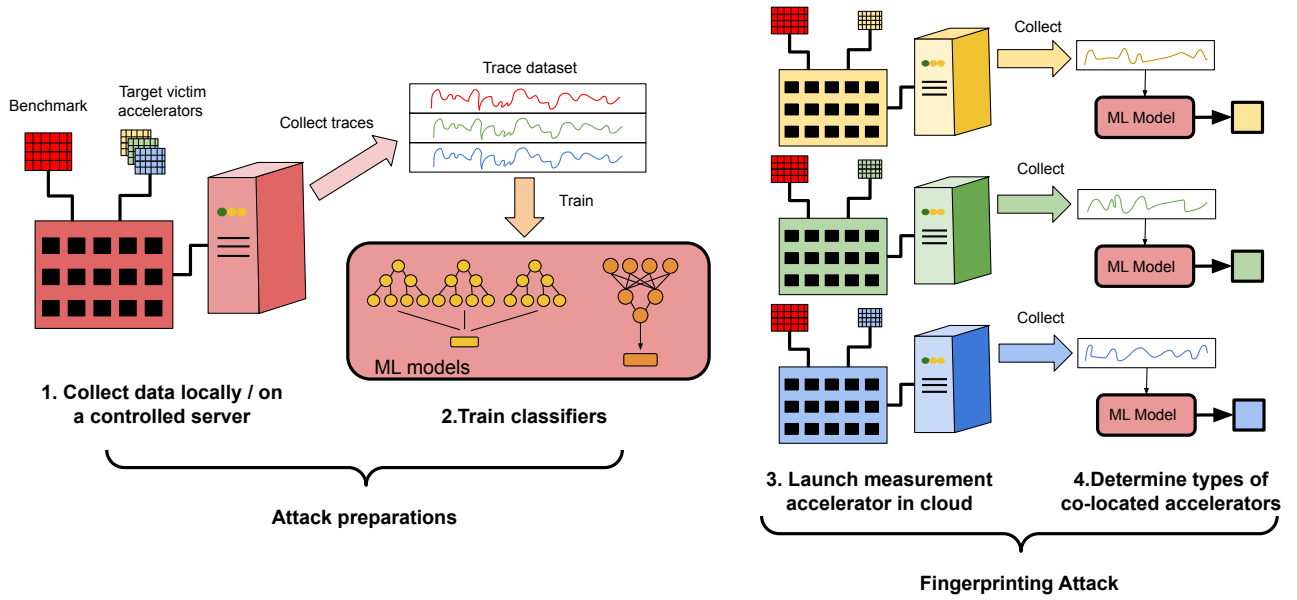
Gotcha! I Know What You Are Doing on the FPGA Cloud

CCS '23, November 26–30, 2023, Copenhagen, Denmark

Figure 1: Diagram of our FPGA fingerprinting attack.

```
Allocate buffer[BUFFER_SIZE][1..BUFFER_NUM];
trace = [];
for(i = 1..BUFFER_NUM) {
  t_i = 0;
  for(j = 1..REPEAT_NUM) {
      call accelerator and operate on buffer[i];
      t_i += time of kernel execution;
  }
  t_i /= REPEAT_NUM;
  trace.append(1 / t_i);
}
return trace;
```

Figure 2: The pseudo code of our host program.

```
__kernel void mem_kernel(__global int4 *dst) {
    int id = get_global_id(0);
    for(long i = 0; i < ACCESS_NUM; i ++) {
        dst[id] = (int)dst ^ dst[id];
    }
}
```

Figure 3: The OpenCL code of our benchmark kernel.

in the same data set. Each trace will be labelled according to the types of corresponding victim accelerators.

The diagram of our data processing flow is shown in Figure 4. We can see that each data point within a trace corresponds to the measurement result of kernel execution on an assigned buffer. All the data points will be combined as a feature vector and be fed to machine learning models for further processing.

The resulting data set will consist of all the collected traces, where each row represents one trace. There will be BUFFER_NUM +1 columns in each row, with BUFFER_NUM entries for collected bandwidth data and 1 entry for label. The data set will be fed to the machine learning models for training.
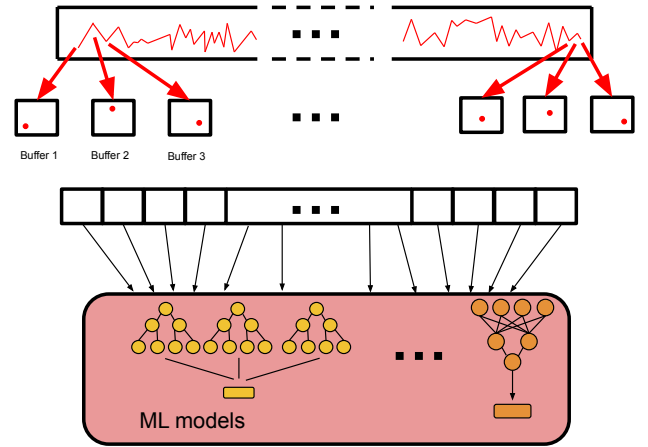

Figure 4: The diagram of our data processing flow.

4.3 Classifiers

The collected traces are 1-D vectors with a fixed dimension, since the number of data points are automatically defined by BUFFER_NUM in the implementation of benchmark circuit. We explore multiple types of machine learning models to assess the potential leakage of the side-channel incurred by the shared communication link. Since we are performing classification tasks and we aim to reduce the costs of attackers by collecting as little data as possible, we select

small models that tend to perform well under these scenarios, e.g. Random Forest [28] which has been used in fingerprinting tasks [45] and also compare the performance of more complex models, e.g. Convolutional Neural Networks. The models we examine in this paper include:

(1) 1D-Convolution [34]: 1 convolution layer, followed by a batch normalization layer, a ReLU layer, 3 layers of fully connected perceptrons [18] and a Softmax layer [26].
(2) Multi-layer Perceptron (MLP): 3 layers of fully-conected perceptrons [18].
(3) Support Vector Machine (SVM) [12]: classic model that is implemented in popular machine learning libraries [4, 46].
(4) Random Forest [28]: classic model that is implemented in popular machine learning libraries [4, 46].

For more practical usage in real world, i.e., the open-world scenario, we find that Random Forest can still achieve relatively high accuracy rates even with the existence of unseen accelerator traces. We will demonstrate this in our later evaluation.

4.4 Implementation of PoC

The PoC system is built on Devcloud [1], using Intel FPGA SDK for OpenCL [30]. In the OpenCL toolchain, every OpenCL kernel will be synthesized into customized FPGA circuits. We choose OpenCL because: (1) compared to traditional Verilog RTL design flow, C/C++-based development is faster and sufficient since we don't need to optimize for performance; (2) according to Intel's documentation [3], as long as different kernels are executed in different OpenCL command queues, they can be executed concurrently which conveniently creates an application co-locating environment that satisfies our need. Besides, we chose to perform our experiments on Devcloud because commercial cloud providers have yet to deploy multi-tenancy FPGAs, despite the possibility of their availability in the future. Nonetheless, our research demonstrates the serious danger posed by the PCIe side-channel when multi-tenancy FPGAs become accessible to users.

In our PoC implementation, victim kernels and accelerator kernels will be defined as two unrelated OpenCL kernels running concurrently on the same FPGA. Victim kernels are configured to run continuously to model victim accelerators that process data streams. They operate on host memory spaces different from those allocated for our benchmark accelerator.

Since the execution environment for offline data collection and online attack is the same, in our PoC implementation the collected data set will be divided to training set and test set, where training set will be used to train the classifier models and the classification results on the test set can emulate the classification accuracy of online attack.

5 EVALUATION

5.1 Experiment Settings

5.1.1 Hardware Environment. All our FPGA-related experiments are conducted on Intel DevCloud [1]. DevCloud allows free SSH access to their servers and FPGA resources from academic users. In our experiments, we select to use nodes with Xeon CPUs and Intel Arria 10 series FPGAs. The environment version is development

stack release 1.2.1. To compile our OpenCL kernels, we utilize the tool-chain provided by Intel, which is available on these nodes. Results are all obtained from node named s005-n007.

5.1.2 I/O Measurement Collection. The experiment process is controlled by our host program. After the initial setup of OpenCL environments (getting platform information, setting up context, command queues, etc.), we launch a victim kernel which will run continuously during the experiment. Meanwhile we also launch the proposed benchmark to perform multiple measurements on the PCIe communication link and gather data. In each measurement, we initialize a new memory buffer item in host memory and execute benchmark kernel for BUFFER_NUM times, aggregate acquired data and calculate the average bandwidth as the result of this measurement. For each accelerator, we collect 50 traces, with BUFFER_NUM points in each trace (default value 100). Since victim FPGA circuits and our benchmark circuits (both are synthesized from OpenCL kernel implementation) run concurrently and there is no synchronization step between the two kernels, our experimental setting resembles a multi-tenant FPGA cloud setting.

5.1.3 Victim Accelerators. In our experiments, we select 8 different FPGA-accelerated workloads provided by Xilinx Vitis Accelerator Example repository [56] and FPGA-synthesized GPU workloads from Rodinia benchmark [10]. We make necessary modifications to deploy them on DevCloud. Detailed description and abbreviation codes are listed in Table 1. These accelerators cover different critical areas for FPGA accelerators, including image processing, signal processing, numerical simulation and neural network acceleration thus can serve as representative workloads.

5.1.4 Classifier Settings. The collected data will be further analyzed by our learning models. In our experiment, we build several different models based on Python machine learning libraries like Pytorch [44] and Scikit-learn [46]. The configurations of these models are listed as follows:

- Random-forest: RandomForestClassifier() from Scikit-learn library [46] is used.
- SVM: SVC() classifier from Scikit-learn library [46] is used.
- MLP: built in Pytorch [44], using learning rate 0.001, cross-entropy loss and stochastic gradient descent (SGD) optimizer, being trained for 1500 epochs.
- 1D-Convolution: built in Pytorch [44], using learning rate 0.001, cross entropy loss and Adam optimizer [33], being trained for 1500 epochs.

5.1.5 Attacking Scenarios. In this paper, we consider two attacking scenarios, i.e., closed-world and open world scenario. For closed-world testing, we assume all accelerators are known to the attacker, and we consider the fingerprinting problem as an *n*-way classification problem, with *n* being the number of accelerators in this closed-world. For the more realistic open-world scenario, we consider it as a binary classification problem (since most attackers will have only one specific target for side-channel attacks) and each classifier will be trained to recognize a single accelerator.

Under both attacking scenarios, we split all collected traces into 7 : 3 for training and testing respectively to obtain accuracy data. Additionally, for open-world scenario, we manually remove traces

Table 1: Descriptions of our victim accelerators.

Name	Code	Function
adder	A	Adder implemented using FPGA. It reads inputs from input buffer, computes results and writes back to a output buffer.
apply_watermark	AW	Image processing circuit. It reads an image from input buffer, adds a watermark andhiheh hack to a output buffer.
fir	F	Signal processing circuit. It reads input and coefficient data from input buffer and performs finite impulse response (FIR) filtering, then writes output back to output buffer.
matmul	M	Matrix multiplication circuit. It reads two matrice A and B from input buffer, calculates AB and writes back to output buffer.
convolute	C	Convolution accelerator. It reads an image and filter weights from input buffer, performs convolution and writes the results back to output buffer.
vector_addition	V	This accelerator reads two arrays from input buffer, performs parallel vector addition on the two buffers and writes the results back to output buffer.
noisegen	NG	An accelerator that generates random traffic between host and FPGA.
hotspot	HS	An accelerator employed from Rodinia benchmark [10] that performs thermal simulation by iteratively solving differential equations.

from certain classes in the training set, making these accelerators agnostic to the classifier. The test set will still include traces from these classes to simulate the real-world scenario, where traces from unknown accelerators are collected.

In our experiments, we aim to answer 2 research questions (RQs):

(1) *RQ1:* Does our measurement circuit capture the communication patterns, and what is the accuracy of fingerprinting?
(2) *RQ2:* How do the parameter settings of benchmark impact the attacking results?

5.2 Results

5.2.1 RQ 1: To answer RQ 1, we first present the visualization of measured PCIe side-channel traces and fingerprinting accuracy in Figure 5 and Table 2, respectively. All data are obtained from the 8 FPGA-accelerated workloads mentioned above.
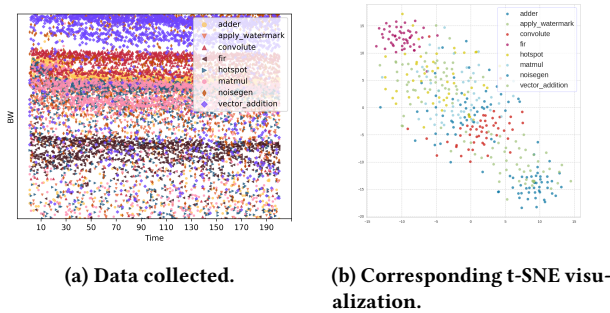

(a) Data collected.

(b) Corresponding t-SNE visualization.

Figure 5: Plain data visualization and t-SNE visualization of performance traces collected by our benchmark.

We first present the collected traces with using t-Distributed stochastic neighbour embedding (t-SNE) [54], which is a widely used data visualization method. It projects high-dimensional data to a 2-D plane and can show how the data points can be clustered. In Figure 5, we collect and visualize the bandwidth traces of our benchmark accelerator when it is running concurrently with one

of the 8 victim circuits. In Figure 5 (a), bandwidth data are normalized with minimum and maximum bandwidth values in the data set and range between 0 and 1. We can see that traces belonging to different accelerators are separable, which indicates that our benchmark circuit is able to capture the unique communication patterns existing in the execution of the victim accelerators and generate fingerprints for each of them. The bandwidth difference exposes a vulnerability of inferring the co-located victim circuit. T-SNE results in Figure 5 (b) also prove that the data traces are separable.

Based on findings in Figure 5, which indicates that these traces contain information that can help differentiate different accelerators, we further consider two fingerprinting attacking scenarios, i.e. closed-world and open-world scenarios. Closed-world fingerprinting aims to classify the types of accelerators within a known accelerator set, while open-world fingerprinting only interests in one sensitive accelerator and classify others as "unrelated".

Closed-world.

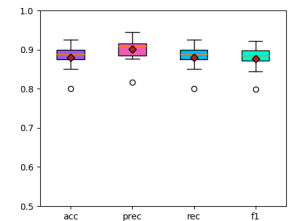| Model	Test Acc.
Random Forest	**88**%
SVM	69%
MLP	55%
1D-Convolution	26%

Table 2: Accuracy results.

Figure 6: Cross-validation results of Random Forest model.

Table 2 shows the closed-world classification accuracy performance of our selected models. Among our models, Random Forest achieves the highest classification accuracy, reaching 88% accuracy in this task. The 10-fold cross-validation results of our Random Forest model is provided in Figure 6. From the distribution of model metrics like accuracy, precision, recall and F1-score we can see that

the model, we can see that the model performance is relatively stable. There are similar but different FPGA workload fingerprinting works [15, 25], where the authors utilize power side-channel to perform classification of different cryptographic cores. Compared to their works, we focus on fingerprinting general computing circuits and utilize a different side-channel. Also, our implementation stays at HLS level.

We select the two models with the best accuracy performance, i.e. SVM and random forest, and provide further details to show how well our model performs in this specific task. Confusion matrices are provided in Figure 7. It shows how accurate the selected models are able to classify each of the victim accelerators. Values in each cell of the confusion matrix represent the number of samples of each (Predicted Label, True Label) pair. We can see from the figure that both models have acceptable accuracy performance (69.2% and 88.3%) and are able to differentiate the 8 accelerator classes, although random forest works better with fewer misclassified samples and outperforms with a great margin. This could be due to the intrinsic features of the data traces, which are potentially more suitable for the algorithm of random forest and decision trees [26].

Open-world. Then we also consider open-world fingerprinting scenario, where the attacker only has one specific target to recognize, and there are traces belonging to unseen accelerators during training process. During the experiments, we randomly select labels to remove and repeat the experiments multiple times to obtain the average accuracy performance data regarding classifiers corresponding to each class of victim accelerators. The accuracy results are shown in Figure 8. We increment the number of unseen accelerators during training process and collect accuracy results for classifiers targeting different accelerators. We can see that the accuracy drops as the number of unknown accelerators increases. However, as long as the attacker has partial knowledge about accelerators in the system, when half of the traces are from unseen accelerators the fingerprinting accuracy can still maintain at around 80% or higher. From Figure 8, we can also see that (1) some accelerators are more vulnerable than others (e.g., our model on `fir` consistently achieves high fingerprinting success rate), highlighting the importance of providing protection when victim is `fir`; (2) when there are less types of accelerators, the fingerprinting success rate is higher and it is more important to provide defence.

In the experiments above, we only use standard min-max scaling pre-processing and standard models. With further customization (filtering data, modifying predictive models), the accuracy can be potentially higher, resulting in a higher success rate and lower costs for continuing side-channel attacks.

Summary. Our benchmark accelerator is able to capture communication patterns of co-located accelerators on FPGA and use these generated fingerprints to classify at a higher accuracy. This accuracy performance is sufficient for use in a real-world scenario. From the classifier side, we find that Random forest model achieves the highest classification accuracy and can reach 88% classification accuracy. Surprisingly, the most complicated model, 1D-Convolution, achieves the worst classification accuracy performance. In our experiments, simpler models like random forest and SVM achieve significantly better accuracy performance. This could be potentially attributed to the limited number of traces in our data set.

5.2.2 *RQ 2:* As mentioned in Section 4, our benchmark has 4 different design parameter:

- `ACCESS_NUM`, which corresponds to how much traffic is generated by benchmark accelerator.
- `REPEAT_NUM`, which is the number of times the kernel is executed and it relates to our measuring granularity and data stability.
- `BUFFER_SIZE`, which determines how large each buffer is.
- `BUFFER_NUM`, which relates to how many data points are collected within one performance trace.
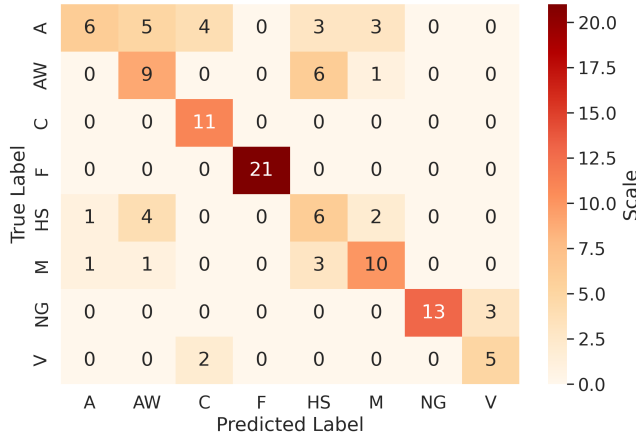
The following parameter settings:

- `ACCESS_NUM`= 1000,
- `REPEAT_NUM`= 10,
- `BUFFER_SIZE`= 4 Bytes,
- `BUFFER_NUM`= 100.

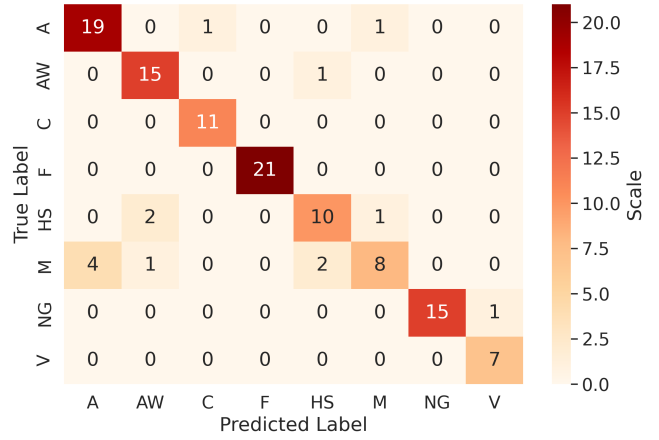will be later referred to as our *default setting*.

In this experiment, we screen all parameters and provide t-SNE visualization and compare their classification accuracy with the one under the default setting. To make visualization results clearer, we drop the simplest accelerator (`noisegen`) and the most complex accelerator (`hotspot`) and only perform attacks on the remaining 6 accelerators. To explore the effects of each of the 4 parameters, we fix the other 3 parameters to the default settings and vary the value of the target parameter, then collect data on the 6 victim accelerators. The t-SNE visualization of the collected data traces as well as classification accuracy traces of our 4 models regarding the 4 parameters are shown in Figure 9 - 12. Corresponding accuracy performance of the 4 models are provided in Figure 13. Figures corresponding to configurations that are identical with default settings are omitted to avoid repetition, the results are the same as in Figure 5 (b). In these figures, we obtain t-SNE results from normalized communication bandwidth data. Overall, the use of different parameter values results in varying trace patterns and can hence affect the accuracy of different models. The analysis of the results we obtain in this experiment is shown as follows.

`ACCESS_NUM`. In Figure 9, the influence of the parameter ACCESS _NUM is shown. From Figure 9 (a) - (d), we can observe a change in the visualization results, i.e. the traces collected from different victim accelerators show different separability. This indicates that the ability of our accelerator benchmark to capture the differentiable patterns in I/O operations existing in our victim accelerators can vary according to `ACCESS_NUM`. We can see that after `ACCESS_NUM` $\geq$ 2000, data points from several accelerator classes tend to be mixed together. By looking at Figure 13 (a), we can see that the random forest model achieves the highest accuracy result, reaching an accuracy over 90% at `ACCESS_NUM`= 1000. SVM achieves over 85% accuracy and MLP achieves over 70% classification accuracy, both at the same point. However, the 1D-Convolution model is only able to achieve 60% accuracy at its highest. We can also see that as `ACCESS_NUM` increases, except 1D-Convolution model, the accuracy generally increases first (though the accuracy of our random forest model stays over 90% relatively stably). After reaching the highest accuracy at `ACCESS_NUM`= 1000, the accuracy starts to drop.

We speculate that the reason behind the fingerprinting accuracy difference is due to the measurement granularity differences when `ACCESS_NUM` ranges from 250 to 4000. Initially, the growth

Gotcha! I Know What You Are Doing on the FPGA Cloud

CCS '23, November 26–30, 2023, Copenhagen, Denmark

(a) Confusion matrix of SVM classifier.

(b) Confusion matrix of Random Forest classifier.

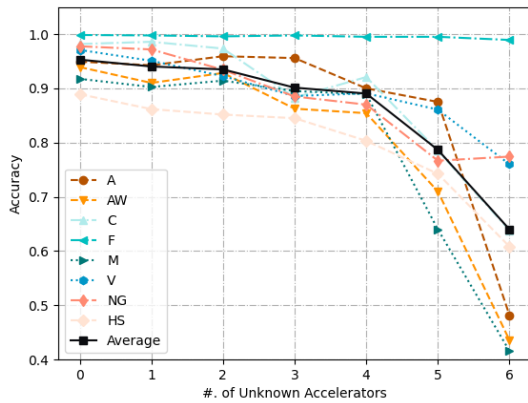**Figure 7: Confusion matrices of the two classifiers.**

Figure 8: Open-world accuracy results.

of `ACCESS_NUM` introduces more data to be read and written hence the effects of noise can be better cancelled and communication patterns can be better captured until `ACCESS_NUM` reaches 1000. However, since the execution time of our benchmark accelerator also increases as `ACCESS_NUM` grows, the measurement will become more coarse-grained since the change in I/O performance variance of victim accelerators within this execution time period will be amortized. After a certain point (in our experiment, between 1000 and 2000), the extended execution time of benchmark accelerator causes the benchmark circuit to lose the ability to accurately capture victim communication patterns, thereby inducing a drop in classification accuracy.

`REPEAT_NUM`. For parameter `REPEAT_NUM`, the t-SNE visualization results are shown in Figure 10. Same as in Figure 9, data points belonging to different accelerator classes in Figure 10 (a) - (d) are separable, where the clearest clustering results appear at

`REPEAT_NUM=` 5 and `REPEAT_NUM=` 10 (see Figure 5). The classification results for the machine learning models in Figure 13 (b) also match this observation, with the highest classification results achieved at `REPEAT_NUM=` 5 and `REPEAT_NUM=` 10, where the accuracy of random forest is again over 90% and the highest accuracy results of SVM and MLP are around 85% and 70%, respectively. In Figure 13 (b) we can observe a similar trend as in Figure 13 (a), where the accuracy first increases to an optimal point and starts to drop as `REPEAT_NUM` grows.

The explanation for the accuracy trend is similar. As `REPEAT_NUM` determines how many times our benchmark is executed when operating on a memory buffer, increasing `REPEAT_NUM` will: (1) cancel the effects of noise and obtain a more precise measurement of the performance; (2) extend the time it takes to operate on a single buffer, i.e. the time it takes to generate a data point in the performance trace. As the execution time of the accelerator kernel task is relatively short, when `REPEAT_NUM` is low, the measurement will be finished within a short period of time and the dynamic communication patterns cannot be captured by our benchmark accelerator. This, combining the influence of noise, is the reason why all models achieve poor classification accuracy results at `REPEAT_NUM=` 1. When `REPEAT_NUM` increases, the communication patterns start to be captured. However, if `REPEAT_NUM` is too large, same as the situation in `ACCESS_NUM` the whole measurement process becomes too coarse-grained. Changes in the I/O communication traffic may be amortized, thus classification models cannot extract detailed information from the collected traces.

`BUFFER_SIZE`. The experimental results of trace visualization and classification results under different `BUFFER_SIZE` settings are shown in Figure 11. We can see that all the `BUFFER_SIZE` settings we use are able to preserve the layering information in the victim accelerators. From Figure 13 (c), we also observe that the influence of parameter `BUFFER_SIZE` is not as much as `ACCESS_NUM` and `REPEAT_NUM`. However, there is an optimal point for the random forest and SVM to work on (`BUFFER_SIZE=` 4 Bytes). We will keep using this empirical value since it is the best work point for our
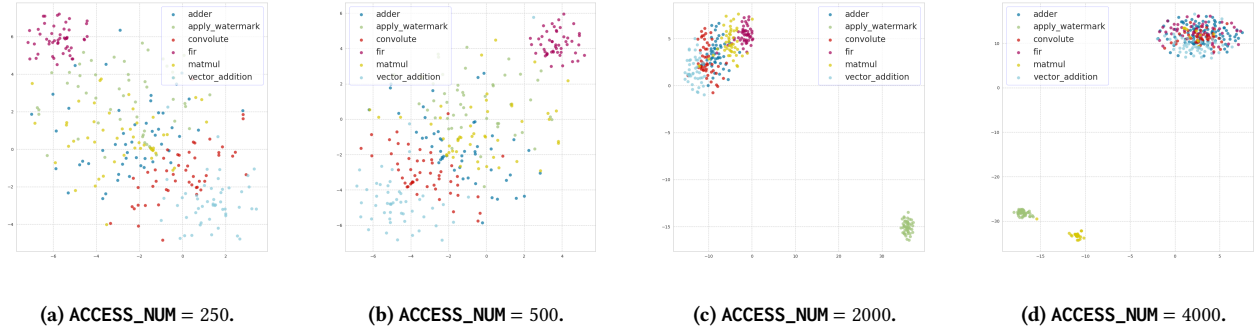
(a) ACCESS_NUM = 250.   **(b) ACCESS_NUM = 500.**   **(c) ACCESS_NUM = 2000.**   **(d) ACCESS_NUM = 4000.**

Figure 9: Traces and corresponding accuracy results under different ACCESS_NUM settings.

(a) REPEAT_NUM = 1.   **(b) REPEAT_NUM = 5.**   **(c) REPEAT_NUM = 20.**   **(d) REPEAT_NUM = 50.**
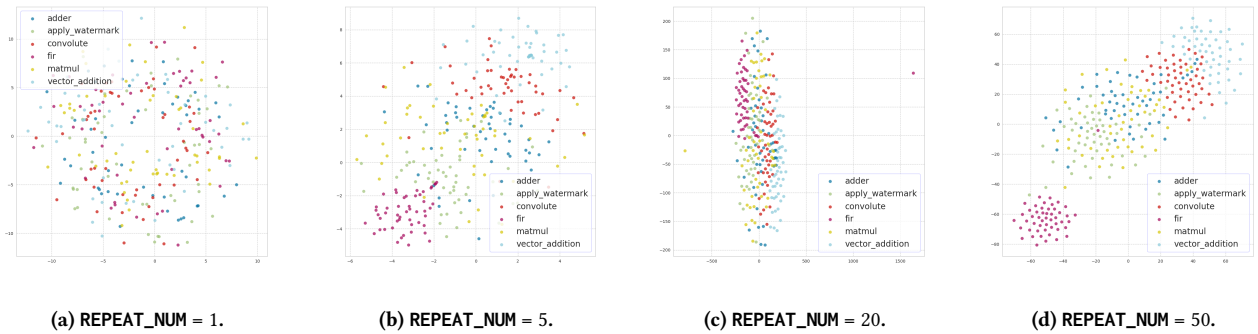
Figure 10: Traces and corresponding accuracy results under different REPEAT_NUM settings.

(a) BUFFER_SIZE = 1 Bytes.   **(b) BUFFER_SIZE = 2 Bytes.**   **(c) BUFFER_SIZE = 8 Bytes.**   **(d) BUFFER_SIZE = 16 Bytes.**
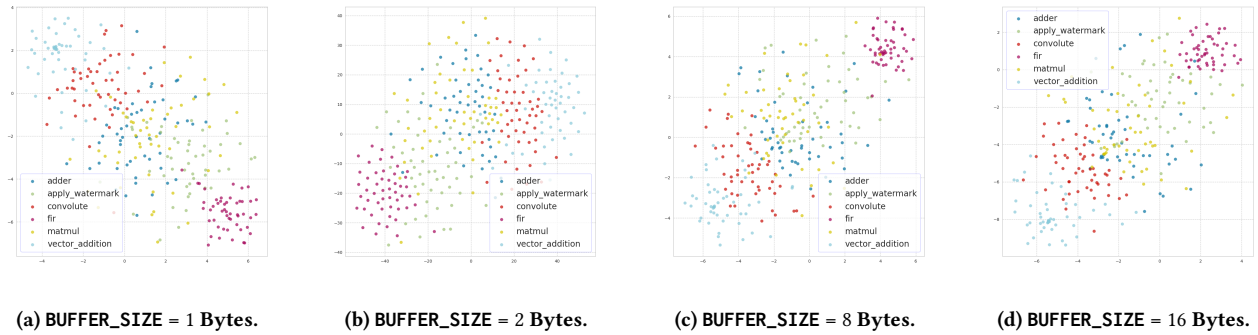
Figure 11: Traces and corresponding accuracy results under different BUFFER_SIZE settings.

most accurate model. However, as BUFFER_SIZE increases beyond the optimal point, there is a slight drop in classification accuracy. This can be due to certain details of the implementation of low-level runtime drivers.

BUFFER_NUM. Results of varying the parameter BUFFER_NUM is shown in Figure 12. By increasing BUFFER_NUM, a longer period of execution of the victim accelerators will be probed and the trace can include more information. However, surprisingly, from Figure 13 (d), the classification accuracy does not change much. With

BUFFER_NUM= 50, our random forest classifier is able to reach over 90%. Other models have a similar trend of accuracy.

Summary. In our experimental results, we show that for a relatively wide range of parameter choices, random forest model is able to achieve satisfying classification accuracy. This helps loose the constraints on attackers' benchmark accelerator implementation. Under ACCESS_NUM= 1000, REPEAT_NUM= 5, BUFFER_NUM= 100, and BUFFER_SIZE= 4Bytes our model is able to achieve the highest accuracy. However, in real world, under some other hardware or
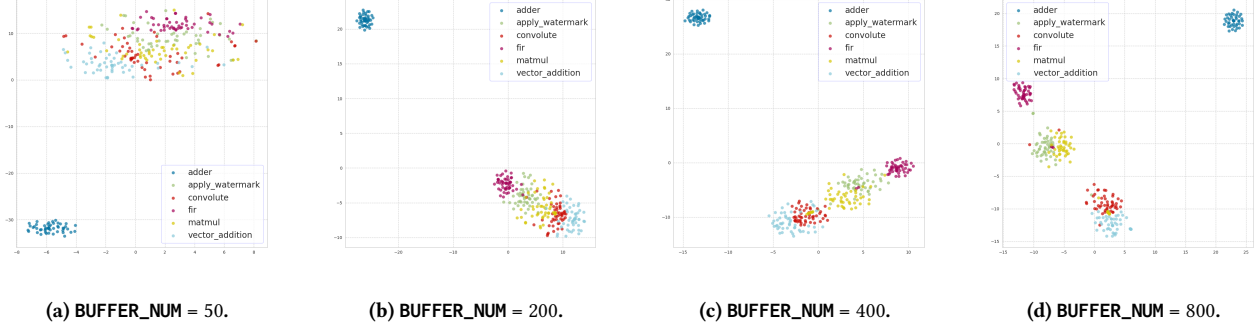
Gotcha! I Know What You Are Doing on the FPGA Cloud

CCS '23, November 26–30, 2023, Copenhagen, Denmark

(a) `BUFFER_NUM` = 50.

(b) `BUFFER_NUM` = 200.

(c) `BUFFER_NUM` = 400.

(d) `BUFFER_NUM` = 800.

Figure 12: Traces and corresponding accuracy results under different `BUFFER_NUM` settings.

software settings (different FPGA models, communication link hardware, or a different heterogeneous computing software stack) these values may vary. To maximize attack performance, attackers are recommended to conduct some offline screening prior to launching the attack to obtain near-optimal parameters. This parameter search does not need to be accurate, since our most powerful model can achieve over 92% accuracy performance under a relatively wide range of attack accelerator parameter choices in the selected accelerator set, which is sufficient for fingerprinting tasks. From the benchmark accelerator side, we conclude that:

(1) Our benchmark accelerator is able to capture the I/O patterns of each of the victim accelerators.

(2) Both `ACCESS_NUM` and `REPEAT_NUM` affect the granularity of measurement and can significantly influence the performance of classification models. There are optimal values for these two values, as shown in Figure 13 (a) and Figure 13 (b).

(3) Buffer related parameters `BUFFER_SIZE` and `BUFFER_NUM` have less influence on classification accuracy. However, the optimal parameter values still exist.

6 DISCUSSION

6.1 Mitigation

The intrinsic cause of the security vulnerability revealed in this paper is the different communication or I/O patterns of accelerators. The different access patterns of accelerators can serve as unique fingerprints of these accelerators. What our benchmark accelerator and host program do is to stress the communication link (i.e. PCIe) and obtain performance measurement trace results that contain information about these fingerprints. This information is further extracted by machine learning models and helps achieve high classification accuracy.

The mitigation to our proposed fingerprint attack can be done by enhancing the FPGA-host interface [e.g. FPGA interface manager (FIM) in Intel cloud FPGAs [1]]. Instead of transmitting raw data, messages travelling through the communication link should pass another security layer for obfuscation. In this obfuscation layer, the communication pattern will be distorted, where random latency/burst will be inserted to make communication patterns unrecognizable. Policies targeting introducing such distortions with minimum performance overhead will be our future work.

From the host side, we can also modify the underlying platforms (OPAE [29], OpenCL [3], etc.). By changing how the driver handles data movement between host server and FPGA, communication pattern obfuscation can also be achieved.

6.2 Attack and Defence Suggestions

6.2.1 For Attackers. In our proposed attack, one prerequisite for attackers is to obtain servers and FPGAs that are identical to the servers and FPGAs used in FPGA clouds. In reality, instead of purchasing hardware and building up the system locally, it's better for attackers to use the cloud itself for data collection. By running data collection steps in the cloud multiple times and recording the underlying hardware and software platform, the attackers can eventually have a set of models that are able to cover heterogeneous hardware and software platforms in the cloud. Doing this step on the victim cloud is more realistic and economic, considering the high cost to set up required hardware and software environments locally.

6.2.2 For Regular FPGA Cloud Users. The fingerprinting attack we propose relies on the intrinsic features of victim accelerators, and we make an assumption that attackers are aware of the target accelerator and can limit the range of accelerators running on the cloud. Therefore, to defend against the proposed fingerprinting attack, FPGA cloud users should be careful about using existing public intellectual property cores (IPs) since these IPs are possibly already in the attackers' database. To achieve this, these users can modify their accelerators and insert noisy I/O or computation operations (additional writes to an unimportant memory location, inserting additional computation between two I/O operations, etc.) to distort the performance traces the attacker may obtain.

In the meantime, it is worth noting that exploiting this security vulnerability also relies on physically residing on the same FPGA where victim accelerator is running. The simplest way for users to avoid being attacked is to obtain ownership of the whole FPGA board as well as the hosting server. This may result in higher costs in deploying FPGA accelerators (since it requires users to pay more to cloud service providers), but it completely eliminates the threat of side-channel-related attacks induced by sharing FPGA resources with unknown users.
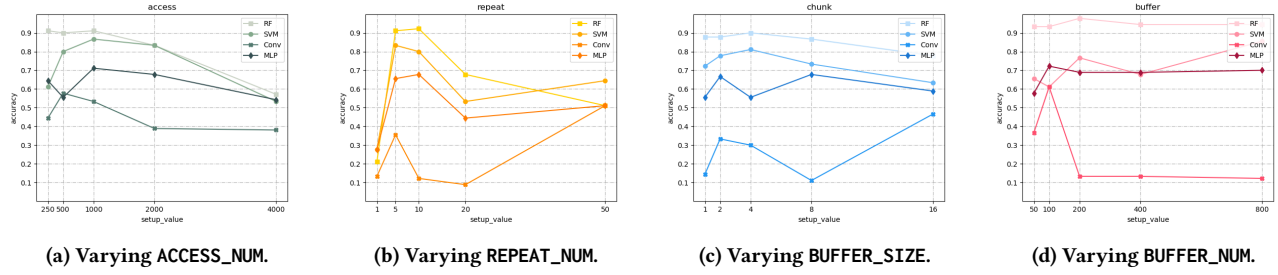
(a) Varying `ACCESS_NUM`.   (b) Varying `REPEAT_NUM`.   (c) Varying `BUFFER_SIZE`.   (d) Varying `BUFFER_NUM`.

Figure 13: Traces and corresponding accuracy results under different `BUFFER_NUM` settings.

6.2.3   For Cloud Service Providers. We suggest cloud service providers enhance their infrastructure interface as mentioned in Section 6.1. Though this may add additional performance overhead, it can efficiently prevent the proposed attack.

Also, FPGA cloud service providers can consider improving their scheduling policy to scatter users' FPGA accelerators on the cloud. It can dramatically reduce the chance of victims' malicious accelerators co-locating with victims' accelerators and hence mitigating side-channel attacks or fingerprinting attacks that require attackers and victims to be placed together.

6.3   Future Work

Future work will be dedicated to developing mitigation technologies against this side-channel. We will come up with both hardware and software-based mitigation strategies. For hardware defence, we will consider deploying low-overhead noise injection circuits. For software defence, we will modify the underlying heterogeneous computing frameworks like OpenCL [41] or OPAE [29] to obfuscate the communication patterns of computation accelerators on board. Besides, cloud scheduler-level defence can be employed to securely schedule/migrate instances.

7   RELATED WORK

FPGA side-channel attacks. Several kinds of remote attacks targeting cloud FPGAs have been proposed recently. One major type is long-wire attack, where attackers utilize leakage in long wires to probe information transmitted inside the circuit. [19] uses the delay difference of nearby wires to probe the signal being transmitted on the long wire, since logical 1 and logical 0 on long wires can lead to different delays of nearby wires. The authors use ROs to capture this difference and use collected information to recover the bits being transmitted on the target long-wire. [48] performs a similar attack and recovers the secret key of an AES crypt circuit. [20] provides detailed tests of several RO designs and validate the efficiency of these variants of long-wire attacks. Defence mechanisms are proposed as well to mitigate long-wire attacks. Remote power side-channel attack is another type of FPGA side-channel attack. In [57], it is performed by programming an on-chip RO-based power monitor to reveal the secret key of a RSA crypto module. This paper also shows that by using the RO-based power monitor, it is possible to perform an FPGA-to-CPU attack on the same SoC. Power side-channel attacks have also been proven to be feasible in

production environment [24], where researchers retrieved AES key information from an AWS EC2 F1 FPGA instance.

Our attack is based on PCIe communication side-channels. There has been attack in FPGA cloud using PCIe contention. In [52], the authors utilize the generation of PCIe contention to perform infrastructure cartography. They use PCIe stressors to generate PCIe contention and reveal information regarding cloud servers in AWS cloud. However, their attack targets multiple FPGAs and aim at revealing infrastructure information instead of revealing information about applications on the same FPGA. In [22, 23], the authors build a covert communication channel based on PCIe contention and consider information leakage in the PCIe contention side-channel. Similar to our work, PCIe traffic is monitored, and information like execution timing traces of victim applications can be obtained. The difference is that we consider multi-tenancy FPGAs (accelerators from multiple users residing on the same FPGA hardware), whereas they consider the scenario where accelerators from different users are distributed to multiple FPGA boards connected to the same server.

The most similar works we find in literature are [15, 25]. To the best of our knowledge, they are also the only works about multi-tenancy FPGA accelerator fingerprinting. In these papers, to achieve a similar goal, the authors propose to use power side-channel for fingerprinting co-located FPGA circuits. Their measurement targets lower-level side-channel leakage and they focus on classifying cryptographic cores, whereas our method is more coarse-grained and we focus on identifying general accelerator workloads.

Our proposed method is more closely related and will be beneficial to side-channel attacks in FPGA cloud, which rely on co-locating with target victims and information about co-located victim circuits. These attacks include attacks targeting cross-talk information leakage [21], power analysis attacks [40, 47] that collects power side-channel information using co-located malicious circuits and reveal secret information from collected data, fault attacks [5] that actively induces faults like voltage drops to victim circuits, etc.

8   CONCLUSION

In this paper, we propose a novel attack targeting multi-tenancy FPGA clouds, where attackers can obtain knowledge about co-located accelerators. By implementing a PoC attack accelerator as well as its corresponding host program, we test accelerators from several application scenarios like signal processing, numerical simulation acceleration, etc. Our results show that communication

Gotcha! I Know What You Are Doing on the FPGA Cloud

CCS '23, November 26–30, 2023, Copenhagen, Denmark

links like PCIe can serve as a new source of side-channel and can be exploited by fingerprinting attacks targeting co-located FPGA accelerators. Our proposed attack method will be beneficial for cloud FPGA side-channel attacks, since successfully recognizing target co-located victims is a prerequisite and can significantly reduce the costs of attacks. As far as we know, this is the first work targeting fingerprinting co-located FPGA accelerators using communication side-channels. Future work will be dedicated to security-enhanced FPGA interface development and another version of this research under an open-world setting.

ACKNOWLEDGMENT

REFERENCES

[1] 2021. Intel Devcloud. [Online; accessed 10 March 2022].
[2] 2022. Intel Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual. [Online; accessed 10 July 2022].
[3] 2022. intel/FPGA-Devcloud. [Online; accessed 10 March 2022].
[4] Martín Abadi. 2016. TensorFlow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. 1–1.
[5] Md Mahbub Alam, Shahin Tajik, Fatemeh Ganji, Mark Tehranipoor, and Domenic Forte. 2019. RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 48–55.
[6] Alibaba Cloud. 2022. Elastic Compute Service: Instance Family. [Online; accessed 17 July 2022].
[7] Amazon. 2022. Amazon EC2 F1 Instances. [Online; accessed 10 March 2022].
[8] Amazon Web Service. 2016. Developer Preview – EC2 Instances (F1) with Programmable Hardware. [Online; accessed 17 July 2022].
[9] Tiago Carneiro, Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C De Albuquerque, and Pedro Pedrosa Reboucas Filho. 2018. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access* 6 (2018), 61677–61685.
[10] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee, 44–54.
[11] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits* 52, 1 (2016), 127–138.
[12] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.
[13] Philippe Coussy and Adam Morawiec. 2010. *High-level synthesis*. Vol. 1. Springer.
[14] Ghada Dessouky, Ahmad-Reza Sadeghi, and Shaza Zeitouni. 2021. SoK: Secure FPGA multi-tenancy in the cloud: Challenges and opportunities. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 487–506.
[15] Colin Drewes, Olivia Weng, Keegan Ryan, Bill Hunter, Christopher McCarty, Ryan Kastner, and Dustin Richmond. 2023. Turn on, Tune in, Listen up: Maximizing Side-Channel Recovery in Time-to-Digital Converters. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 111–122.
[16] Chongzhou Fang, Najmeh Nazari, Behnam Omidi, Han Wang, Aditya Puri, Manish Arora, Setareh Rafatirad, Houman Homayoun, and Khaled N Khasawneh. 2023. HeteroScore: Evaluating and Mitigating Cloud Security Threats Brought by Heterogeneity. (2023).
[17] Chongzhou Fang, Han Wang, Najmeh Nazari, Behnam Omidi, Avesta Sasan, Khaled N Khasawneh, Setareh Rafatirad, and Houman Homayoun. 2022. Repttack: Exploiting Cloud Schedulers to Guide Co-Location Attacks. In *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*.
[18] Stephen I Gallant et al. 1990. Perceptron-based learning algorithms. *IEEE Transactions on neural networks* 1, 2 (1990), 179–191.
[19] Ilias Giechaskiel, Kasper B. Rasmussen, and Ken Eguro. 2018. Leaky Wires: Information Leakage and Covert Communication Between FPGA Long Wires. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (Incheon, Republic of Korea) *(ASIACCS '18)*. Association for Computing Machinery, 15–27. https://doi.org/10.1145/3196494.3196518

[20] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. 2019. Measuring long wire leakage with ring oscillators in cloud FPGAs. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 45–50.
[21] Ilias Giechaskiel and Jakub Szefer. 2020. Information leakage from FPGA routing and logic elements. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.
[22] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. 2021. Cross-VM information leaks in FPGA-accelerated cloud environments. In *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 91–101.
[23] Ilias Giechaskiel, Shanquan Tian, and Jakub Szefer. 2022. Cross-vm covert-and side-channel attacks in cloud fpgas. *ACM Transactions on Reconfigurable Technology and Systems* 16, 1 (2022), 1–29.
[24] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. 2020. Are cloud FPGAs really vulnerable to power analysis attacks?. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1007–1010.
[25] Mustafa Gobulukoglu, Colin Drewes, William Hunter, Ryan Kastner, and Dustin Richmond. 2021. Classifying Computations on Multi-Tenant FPGAs. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1261–1266.
[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[27] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet-Moundi. 2019. High-Speed Ring Oscillator based Sensors for Remote Side-Channel Attacks on FPGAs. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. 1–8. https://doi.org/10.1109/ReConFig48160.2019.8994789
[28] Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Vol. 1. IEEE, 278–282.
[29] Intel. 2017. Open Programmable Acceleration Engine - Documentation. [Online; accessed 10 March 2022].
[30] Intel. 2022. Intel FPGA SDK for opencl pro edition: Programming guide. [Online; accessed 10 March 2022].
[31] Intel. 2022. oneAPI Programming Model. [Online; accessed 10 March 2022].
[32] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, et al. 2018. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 29–42.
[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[34] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 2021. 1D convolutional neural networks and applications: A survey. *Mechanical systems and signal processing* 151 (2021), 107398.
[35] Oliver Knodel, Paul R Genssler, Fredo Erxleben, and Rainer G Spallek. 2018. FPGAs and the cloud–An endless tale of virtualization, elasticity and efficiency. *International Journal on Advances in Systems and Measurements* 11, 3-4 (2018), 230–249.
[36] Jonas Krautter, Dennis RE Gnad, and Mehdi B Tahoori. 2022. Remote Fault Attacks in Multi-Tenant Cloud FPGAs. *IEEE Design & Test* (2022).
[37] Christian Krieg, Clifford Wolf, and Axel Jantsch. 2016. Malicious LUT: a stealthy FPGA trojan injected and triggered by the design flow. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.
[38] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).
[39] Microsoft Research. 2017. Microsoft unveils Project Brainwave for real-time AI. [Online; accessed 17 July 2022].
[40] Shayan Moini, Shanquan Tian, Daniel Holcomb, Jakub Szefer, and Russell Tessier. 2021. Remote power side-channel attacks on BNN accelerators in FPGAs. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1639–1644.
[41] Aaftab Munshi. 2009. The opencl specification. In *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 1–314.
[42] Najmeh Nazari, Hosein Mohammadi Makrani, Chongzhou Fang, Behnam Omidi, Setareh Rafatirad, Hossein Sayadi, Khaled N Khasawneh, and Houman Homayoun. 2023. Adversarial Attacks against Machine Learning-based Resource Provisioning Systems. *IEEE Micro* (2023).
[43] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W Moore. 2018. Understanding PCIe performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 327–341.
[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
[45] Kartik Patwari, Syed Mahbub Hafiz, Han Wang, Houman Homayoun, Zubair Shafiq, and Chen-Nee Chuah. 2022. DNN Model Architecture Fingerprinting Attack on CPU-GPU Edge Devices. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 337–355.

[46] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[47] George Provelengios, Daniel Holcomb, and Russell Tessier. 2019. Characterizing power distribution attacks in multi-user FPGA environments. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 194–201.

[48] Chethan Ramesh, Shivukumar B. Patil, Siva Nishok Dhanuskodi, George Provelengios, Sebastien Pillement, Daniel Holcomb, and Russell Tessier. 2018. FPGA Side Channel Attacks without Physical Access. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 45–52. https://doi.org/10.1109/FCCM.2018.00016

[49] Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[50] Pawel Swierczynski, Georg T Becker, Amir Moradi, and Christof Paar. 2017. Bitstream fault injections (BiFI)–Automated fault attacks against SRAM-based FPGAs. *IEEE Trans. Comput.* 67, 3 (2017), 348–360.

[51] Mingtian Tan, Junpeng Wan, Zhe Zhou, and Zhou Li. 2021. Invisible probe: Timing attacks with pcie congestion side-channel. In *2021 IEEE Symposium on*

[52] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. 2021. Cloud FPGA cartography using PCIe contention. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 224–232.

[53] Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. 2020. Fingerprinting cloud FPGA infrastructures. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 58–64.

[54] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[55] Zane Weissman, Thore Tiemann, Daniel Moghimi, Evan Custodio, Thomas Eisenbarth, and Berk Sunar. 2019. Jackhammer: Efficient rowhammer on heterogeneous fpga-cpu platforms. *arXiv preprint arXiv:1912.11523* (2019).

[56] Xilinx. 2020. Vitis Accel Examples' Repository. [Online; accessed 10 March 2022].

[57] Mark Zhao and G. Edward Suh. 2018. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. 229–244. https://doi.org/10.1109/SP.2018.00049

Security and Privacy (SP). IEEE, 322–338.