Cross-VM Information Leaks in FPGA-Accelerated Cloud Environments

Ilias Giechaskiel*, Shanquan Tian*, and Jakub Szefer

14th IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Tysons Corner, Virginia, United States of America, December 2021

Cross-VM Information Leaks in FPGA-Accelerated Cloud Environments

Ilias Giechaskiel[†]
Independent Researcher
London, United Kingdom
ilias@giechaskiel.com

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

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

***Abstract*—The availability of FPGAs in cloud data centers offers rapid, on-demand access to hardware compute resources that users can configure to their own needs. However, the low-level access to the hardware FPGA and associated resources such as PCIe, SSD, or DRAM also opens up threats of malicious attackers uploading designs that are able to infer information about other users or about the cloud infrastructure itself. In particular, this work presents a new, fast PCIe-contention-based channel that is able to transmit data between different FPGA-accelerated virtual machines with bandwidths reaching 2 kbps with 97% accuracy. This paper further demonstrates that the PCIe receiver circuits are able to not just receive covert transmissions, but can also perform fine-grained monitoring of the PCIe bus or detect different types of activities from other users' FPGA-accelerated virtual machines based on their PCIe traffic signatures. Beyond leaking information across different virtual machines, the ability to monitor the PCIe bandwidth over hours or days can be used to estimate the data center utilization and map the behavior of the other users. The paper also introduces further novel threats in FPGA-accelerated instances, including contention due to shared NVMe SSDs as well as thermal monitoring to identify FPGA co-location using the DRAM modules attached to the FPGA boards. This is the first work to demonstrate that it is possible to break the separation of privilege in FPGA-accelerated cloud environments, and highlights that defenses for public clouds using FPGAs need to consider PCIe, SSD, and DRAM resources as part of the attack surface that should be protected.**

I. Introduction

Public cloud infrastructures with FPGA-accelerated virtual machine (VM) instances allow for easy, on-demand access to reconfigurable hardware that users can program with their own designs. The FPGA-accelerated instances can be used to accelerate machine learning, image and video manipulation, or genomic applications, for example [5]. The potential benefits of the instances with FPGAs have resulted in numerous cloud providers including Amazon Web Services (AWS) [9], Alibaba [3], Baidu [16], Huawei [27], Nimbix [35], and Tencent [39], giving public users direct access to FPGAs. However, allowing users low-level access to upload their own hardware designs has resulted in serious implications for the security of the cloud users and the cloud infrastructure itself.

Several recent works have considered the security implications of shared ("multi-tenant") FPGAs in the cloud, and have demonstrated covert-channel [23] and side-channel [25] attacks. However, today's cloud providers, such as AWS with their F1 instances, only offer "single-tenant" access to FPGAs. In the single-tenant setting, each FPGA is fully dedicated to the one user who rents it, while many other users may be in parallel using their separate, dedicated FPGAs which are within the same server. Once an FPGA is released by a user, it can then be assigned to the next user who rents it. This can lead to temporal thermal covert channels [41], where heat generated by one circuit can be later observed by other circuits that are loaded onto the same FPGA. Such channels are slow (less than 1 bps), and are only suitable for covert communication, since they require the two parties to coordinate and keep being scheduled on the same physical hardware one after the other.

Other means of covert communication in the single-tenant setting do not require being assigned to the same FPGA chip. For example, multiple FPGA boards in servers share the same power supply, and prior work has shown the potential for such shared power supplies to leak information between FPGA boards [24]. However, the resulting covert channel was slow (less than 10 bps) and was only demonstrated in a lab setup.

Another single-tenant security topic that has been previously explored is that of fingerprinting FPGA instances using Physical Unclonable Functions (PUFs) [40], [42]. Fingerprinting allows users to partially map the infrastructures and get some insights about the allocation of FPGAs (e.g., how likely a user is to be re-assigned to the same physical FPGA they used before), but fingerprinting by itself does not lead to information leaks. A more recent fingerprinting-related work explored mapping FPGA infrastructures using PCIe contention to find which FPGAs are co-located in the same Non-Uniform Memory Access (NUMA) node within a server [43]. However, no prior work has successfully launched a cross-VM covert- or side-channel attack in a real cloud FPGA setting.

By contrast, this work shows for the first time that shared resources can be used to leak information across separate virtual machines running on the FPGA-accelerated F1 instances in AWS data centers. In particular, we use the contention of the PCIe bus to not only demonstrate a new, fast covert channel (reaching up to 2 kbps), but also to identify patterns of activity based on the PCIe signatures of different Amazon FPGA Images (AFIs) used by other users. Our attacks do not require special privileges or potentially malicious circuits such as Ring Oscillators (ROs) or Time-to-Digital Converters (TDCs),

[†] The first two authors contributed equally to this work. This work was supported in part by NSF grant 1901901.

and can thus not easily be detected through static analysis or Design Rule Checks (DRCs) that cloud providers may perform. We further introduce two new methods of finding co-located instances that are in the same physical server: (a) through resource contention of the Non-Volatile Memory Express (NVMe) SSDs that are accessible from each F1 instance, and (b) through the common thermal signatures obtained from the decay rates of each FPGA's DRAM modules. Our work therefore shows that single-tenant attacks in real FPGA-accelerated cloud environments are practical, and demonstrates several ways to infer information about the operations of other cloud users and their FPGA-accelerated virtual machines.

A. Contributions

In summary, the contributions of this work are:

1) The demonstration of the first covert channel between separate F1 instance virtual machines, reaching 2 kbps with 97% accuracy.
2) Cross-VM side-channel leaks that reveal information about the behavior of different users through the PCIe signatures of their Amazon FPGA Images (AFIs).
3) A long-term monitoring approach of data center activity through PCIe contention.
4) The identification of an alternative, SSD-based co-location mechanism and possible covert channel between separate F1 users.
5) A new DRAM-based co-location mechanism to identify F1 instances which are on separate NUMA nodes, but share the same server.

B. Responsible Disclosure

Our findings and a copy of this paper have been shared with the AWS security team.

C. Paper Organization

The remainder of the paper is organized as follows. Section II provides the background on today's deployments of FPGAs in public cloud data centers. Section III discusses typical FPGA-accelerated cloud servers and PCIe contention that can occur among the FPGAs, while Section IV evaluates our fast, PCIe-based, cross-VM channel. Using the ideas from the covert channel, Section V investigates how to infer information about other VMs through their PCIe traffic patterns, with Section VI revolving around long-term PCIe monitoring of data center activity. Section VII then presents an alternative contention mechanism due to shared SSDs among FPGA-accelerated instances, while Section VIII introduces a new, thermal-based co-location mechanism. The paper ends with a list of related work in Section IX, and concludes in Section X.

II. Background

This section provides a brief background on FPGAs available in public cloud computing data centers, with a focus on the F1 instances from Amazon Web Services (AWS) [9] that are evaluated in this work.

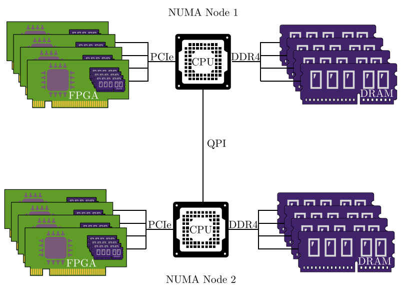


Fig. 1: Prior work suggested that AWS servers contain 8 FPGAs, equally divided between two NUMA nodes [43].

A. AWS F1 Instance Architecture

AWS has offered FPGA-accelerated virtual machine instances to users since late 2016 [4]. These so-called F1 instances are available in three sizes: `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge`, where the instance name represents twice the number of FPGAs it contains (so `f1.2xlarge` has 1 FPGA, while `f1.4xlarge` has 2, and `f1.16xlarge` has 8 FPGAs). Each instance is allocated 8 virtual CPUs (vCPUs), 122 GiB of DRAM, and 470 GB of NVMe SSD storage per FPGA. For example, `f1.4xlarge` instances have 16 vCPUS, 244 GiB of DRAM, and 940 GB of SSD space [9], since they contain 2 FPGAs.

Each FPGA board is attached to the server over a x16 PCIe Gen 3 bus. In addition, each FPGA board contains four DDR4 DRAM chips, totaling 64 GiB of memory per FPGA board [9]. These memories are separate from the server's DRAM and are directly accessible by each FPGA. The F1 instances use Virtex UltraScale+ XCVU9P chips [9], which contain over 1.1 million lookup tables (LUTs), 2.3 million flip-flops (FFs), and 6.8 thousand Digital Signal Processing (DSP) blocks [47].

As has recently been shown, each server contains 8 FPGA boards, which are evenly split between two Non-Uniform Memory Access (NUMA) nodes [43]. The AWS server architecture, as deduced by Tian et al. [43], is shown in Figure 1. Due to this architecture, an `f1.2xlarge` instance may be on the same NUMA node as up to three other `f1.2xlarge` instances, or on the same NUMA node as one other `f1.2xlarge` instance and one `f1.4xlarge` instance (which uses 2 FPGAs). Conversely, an `f1.4xlarge` instance may share the same NUMA node with up to two `f1.2xlarge` instances, or one `f1.4xlarge` instance. Finally, as `f1.16xlarge` instances use up all 8 FPGAs in the server, they do not share any resources with other instances, since both NUMA nodes of the server are fully occupied.

B. Programming AWS F1 Instances

Users utilizing F1 instances do not retain entirely unrestricted control to the underlying hardware, but instead need to adapt their hardware designs to fit within a predefined architecture. In particular, user designs are defined as "Custom Logic (CL)" modules that interact with external interfaces through the cloud-provided "Shell", which hides physical aspects such as clocking logic and I/O pinouts (including for

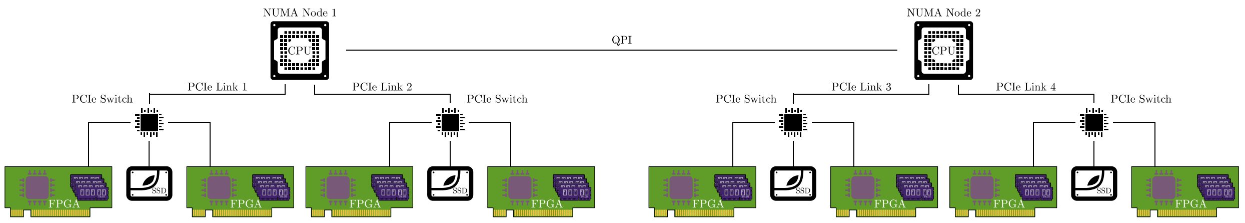


Fig. 2: Our deduced PCIe configuration for F1 servers based on the experiments in this work: each CPU has two PCIe links, each of which provides connectivity to two FPGAs and an NVMe SSD through a PCIe switch.

PCIe and DRAM) [23], [42]. This restrictive Shell interface further prevents users from accessing identifier resources, such as eFUSE and Device DNA primitives, which could be used to distinguish between different FPGA boards [23], [42]. Finally, users cannot directly upload bitstreams to the FPGAs. Instead, they generate a Design Checkpoint (DCP) file using Xilinx's tools and then provide it to Amazon to create the final bitstream (Amazon FPGA Image, or AFI), after it has passed a number of Design Rule Checks (DRCs). The checks, for example, include prohibiting combinatorial loops such as Ring Oscillators (ROs) as a way of protecting the underlying hardware [22], [23], though alternative designs bypassing these restrictions have been proposed [23], [38].

III. PCIe Contention in Cloud FPGAs

The user's Custom Logic running on the FPGA instances can use the Shell to communicate with the server through the PCIe bus. Users cannot directly control the PCIe transactions, but instead perform simple reads and writes to predefined address ranges through the Shell. These memory accesses get translated into PCIe commands and PCIe data transfers between the server and the FPGA. The users may also set up Direct Memory Access (DMA) transfers between the FPGA and the server. By designing hardware modules with low logic overhead, users can generate transfers fast enough to saturate the PCIe bandwidth. In fact, because of the shared PCIe bus within each Non-Uniform Memory Access (NUMA) node, these transfers can create interference and bus contention that affects the PCIe bandwidth of other users. The resulting performance degradation can be used for detecting co-location [43] or, as we show in this work, for fast covert- and side-channel attacks, in effect breaking separation of privilege between otherwise logically and physically separate VM instances.

Figure 1 shows the assumed AWS server configuration with 8 FPGAs per server, based on prior work by Tian et al. [43] and publicly available information on AWS F1 instances [9], [13]. AWS servers containing FPGAs have two Intel Xeon E5-2686 v4 (Broadwell) processors, connected through an Intel QuickPath Interconnect (QPI) link. Each processor forms its own NUMA node with its associated DRAM and four FPGAs attached as PCIe devices.

In our covert-channel analysis (Section IV), we show that the communication bandwidth is not identical for all pairs of FPGAs in a NUMA node. In particular, this suggests that the 4 PCIe devices are not directly connected to each CPU, but instead likely go through two separate switches, forming the hierarchy shown in Figure 2. Although not publicly confirmed by AWS, this topology is similar to the one described for P4d instances, which contain 8 GPUs [7]. As a result, although all 4 FPGAs in a NUMA node contend with each other, the covert-channel bandwidth is highest amongst those sharing a PCIe switch, due to the bottleneck imposed by the shared link.

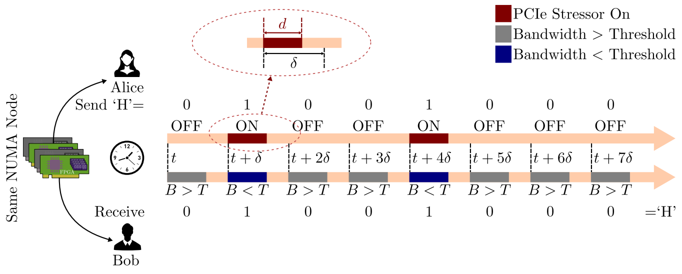

🟥 PCIe Stressor On
▨ Bandwidth > Threshold
🟦 Bandwidth < Threshold
d
δ
Alice
Send 'H'=
0 1 0 0 1 0 0 0
OFF ON OFF OFF ON OFF OFF OFF
Same NUMA Node
t $t+\delta$ $t+2\delta$ $t+3\delta$ $t+4\delta$ $t+5\delta$ $t+6\delta$ $t+7\delta$
$B>T$ $B<T$ $B>T$ $B>T$ $B<T$ $B>T$ $B>T$ $B>T$
Receive
0 1 0 0 1 0 0 0 ='H'
Bob

Fig. 3: Example cross-VM covert communication: The transmitter (Alice) sends the ASCII byte 'H', represented as 01001000 in binary, to the receiver (Bob) in 8 measurement intervals by stressing her PCIe bandwidth to transmit a 1 and remaining idle to transmit a 0. If Bob's FPGA bandwidth B drops below a threshold T, he detects a 1, otherwise a 0 is detected. To ensure no residual effects after each transmission, the time difference δ between successive measurements is slightly larger than the transmission duration d.

IV. PCIe-Based Cross-VM Covert Channel

In this section we describe our implementation for the first cross-VM covert-channel on public cloud FPGAs using PCIe contention (Section IV-A), and discuss our experimental setup (Section IV-B). We then analyze bandwidth vs. accuracy trade-offs (Section IV-C), before investigating the impact of receiver and transmitter transfer sizes on the covert-channel accuracy for a given covert-channel bandwidth (Section IV-D). Side channels and information leaks based on PCIe contention from other VMs are discussed in Section V, while Section VI shows long-term PCIe-based monitoring of data center activity.

A. *Covert-Channel Implementation*

Our covert-channel is based on saturating the PCIe link between the FPGA and the server, so, at their core, both the transmitter and the receiver consist of a) an FPGA image that responds to PCIe requests with minimal latency, and b) software that attaches to the FPGA and repeatedly writes to the mapped Base Address Register (BAR). The transmitter stresses its PCIe link to transmit a 1 but remains idle to transmit a

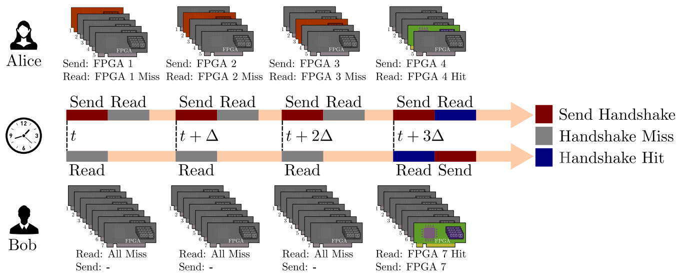


Fig. 4: The process to find a pair of co-located `f1.2xlarge` instances using PCIe contention uses the covert-channel mechanism to check for pre-agreed handshake messages: Alice transmits the handshake message with her first FPGA, and waits to see if Bob acknowledges the handshake message. In parallel, Bob measures the bandwidths of all his FPGAs. In this example, Bob detects the contention in his seventh FPGA during the fourth handshake attempt. Note that Alice and Bob can rent any number of FPGAs for finding co-location, with five and seven shown in this figure as an example.

bit 0, while the receiver keeps measuring its own bandwidth during the transmission period (the receiver is thus identical to a transmitter that sends a 1 during every measurement period). The receiver then classifies the received bit as a 1 if the bandwidth has dropped below a threshold, and as 0 otherwise.

The two communicating parties need to have agreed upon some minimal information prior to the transmissions: the specific data center to use (region and availability zone, e.g., `us-east-1e`), the time t to start communications, and the initial measurement period, expressed as the time difference between successive transmissions δ. All other aspects of the communication can be handled within the channel itself, including detecting that the two parties are on the same NUMA node, or increasing the bandwidth by decreasing δ. To ensure that the PCIe link returns to idle between successive measurements, transmissions stop before the end of the measurement interval, i.e., the transmission duration d satisfies $d < \delta$. Note that synchronization is implicit due to the receiver and transmitter having access to a shared wall clock time, e.g., via the Network Time Protocol (NTP). Figure 3 provides a high-level overview of our covert-channel mechanism.

Before they can communicate, the two parties (Alice and Bob in the example of Figure 3) first need to ensure that they are co-located on the same NUMA node within the server. To do so, they can launch multiple instances and attempt to detect whether any of their instances are co-located by sending handshake messages and expecting a handshake response, using the same setup information as for the covert channel itself (i.e., the time t to start the communication, the measurement duration δ, and setup information such as the data center region and availability zone). They additionally need to have agreed on the handshake message, which determines the per-handshake measurement duration Δ. This co-location process is summarized in Figure 4.

B. Experimental Setup

For our experiments, we use VMs with AWS FPGA Developer Amazon Machine Image (AMI) [14] version 1.8.1, which runs CentOS 7.7.1908, and develop our code with the Hardware and Software Development Kit (HDK/SDK) version 1.4.15 [8]. For our FPGA bitstream, we use the unmodified `CL_DRAM_DMA` image provided by AWS (`agfi-0d132ece5c8010bf7`) [12] for both the transmitter and the receiver designs. Our custom-written software maps the FPGA DRAM modules via PCIe Application Physical Function (AppPF) BAR4, a 64-bit prefetchable Base Address Register (BAR) [11]. To support write-combining for higher performance, we use the `BURST_CAPABLE` flag, and implement the data transfer using `memcpy`, getting similar performance to the AWS benchmarks [6].

Unless otherwise noted, we perform experiments with "spot" instances in the `us-east-1` (North Virginia) region in availability zone `d`, though prior work has shown that PCIe contention is present in all regions, with both spot and on-demand instances [43]. Although the results presented are for instances launched by a single user, it should also be noted that we have successfully created a cross-VM covert channel between instances launched by two different users.

C. Bandwidth vs. Accuracy Trade-Offs

Using our co-location mechanism, we found 4 distinct `f1.2xlarge` instances that are all in the same NUMA node,[1] and then measured the covert-channel accuracy for different bandwidths, i.e., different measurement parameters d and δ. Specifically, we tested (d, δ) from $(0.1\,\text{ms}, 0.2\,\text{ms})$ to $(9\,\text{ms}, 10\,\text{ms})$, corresponding to transmission rates between $5\,\text{kbps}$ and $100\,\text{bps}$. For these experiments, the receiver keeps transferring $2\,\text{kB}$ chunks of data from the host, while the

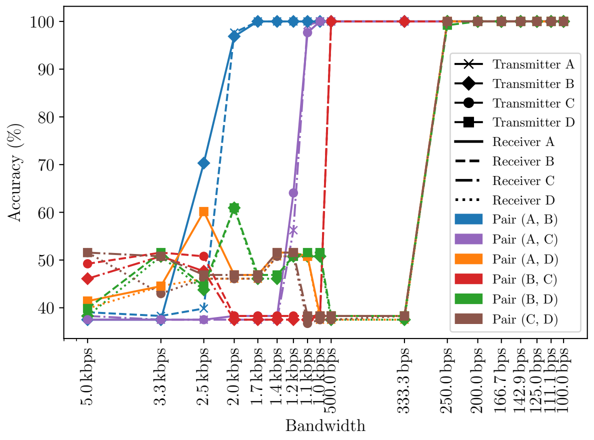


Fig. 5: Bandwidth and accuracy for covert-channel transmissions between any pair of FPGAs, among the four FPGAs in the same NUMA node. Each FPGA pair is color-coded, with transmitters indicated through different markers, and receivers through different line styles. For any given pair, the bandwidth is approximately the same in each direction, i.e., the bandwidth from FPGA X to FPGA Y is approximately the same as the bandwidth from Y to X. Communication is possible between any two FPGAs in the NUMA node, but the bandwidths for different pairs diverge.

[1]Similar to [43], we often found full NUMA nodes within a few minutes when launching ten `f1.2xlarge` instances.

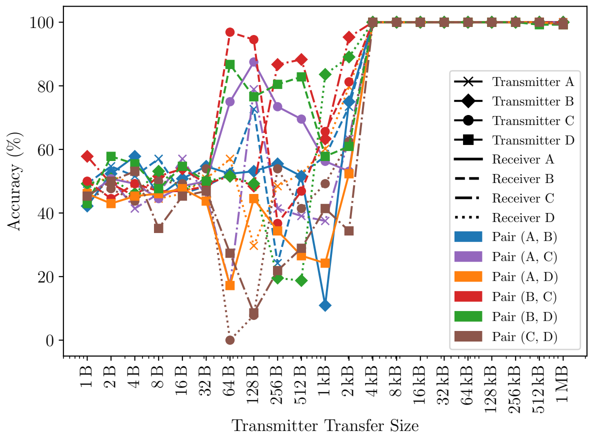


Fig. 6: Covert-channel accuracy for different transmitter transfer sizes. Each chunk transmitted over PCIe needs to be at least 4 kB to an ensure an accuracy of 100% at 200 bps between any two FPGAs in the NUMA node.

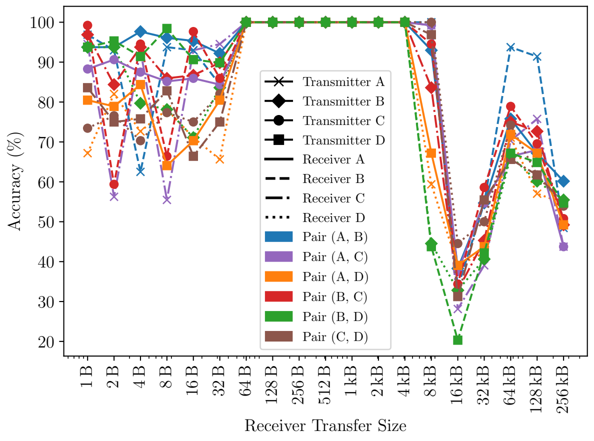


Fig. 7: Covert-channel accuracy for different receiver transfer sizes. Chunks between 64 B and 4 kB are suitable for 100% accuracies, but sizes outside this range result in a drop in accuracy for at least one pair of FPGAs in the NUMA node.

transmitter repeatedly sends 64 kB of data in each transmission period (i.e., until the end of the interval d). These parameters are explored separately in Section IV-D below.

The results of our experiments, shown in Figure 5, indicate that we can create a fast covert channel between any two FPGAs in either direction: at 200 bps and below, the accuracy of the covert channel is 100%, with the accuracy at 250 bps dropping to 99% for just one pair. At 500 bps, three of the six possible pairs can communicate at 100% accuracy, while one pair can communicate with 97% accuracy at 2 kbps. It should be noted that, as expected, the bandwidth within any given pair is symmetric, i.e., it remains the same when the roles of the transmitter and the receiver are reversed. As the VMs occupy a full NUMA node, there should not be any impact from other users' traffic. The variable bandwidth between different pairs is therefore likely due to the PCIe topology.

D. Transfer Sizes

In this set of experiments, we fix $d = 4\,\text{ms}$, $\delta = 5\,\text{ms}$ (i.e., a covert-channel bandwidth of 200 bps), and vary the transmitter and receiver transfer sizes. Figure 6 first shows the per-pair channel accuracy for different transmitter sizes. The results show that at 4 kB and above, the covert-channel accuracy is 100%, while it remains much lower at smaller transfer sizes. This is because sending smaller chunks of data over PCIe results in lower bandwidth due to the associated PCIe overhead of each transaction. For example, in one 4 ms transmission, the transmitter completes 140,301 transfers of 1 B each, corresponding to a bandwidth of only $1\,\text{B} \times 140{,}301/4\,\text{ms} = 33.5\,\text{MBps}$. However, in the same time, a transmitter can complete 1,890 transfers of 4 kB, for a bandwidth of $4\,\text{kB} \times 1{,}890/4\,\text{ms} = 1.8\,\text{GBps}$.[2]

The results of the corresponding experiments for receiver transfer sizes are shown in Figure 7. Similar to the transmitter experiments, very small transfer sizes are unsuitable for the covert channel due to the low resulting bandwidth. However, unlike in the transmitter case, large receiver transfer sizes are also problematic, as the number of transfers completed within each measurement interval is too small to distinguish between external transmissions and the inherent measurement noise.

[2]The maximum transfer size used of 1 MB was chosen to ensure that multiple transfers were possible within each transfer interval without ever interfering with the next measurement interval.

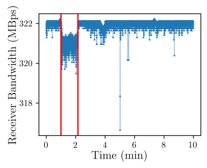
V. Side-Channel Leaks in Marketplace AMIs

The F1 instances studied in this paper can be leveraged to accelerate various types of computations. To help users in that regard, the AWS Marketplace lists numerous virtual machine images created and sold by independent software vendors [10]. Users can purchase instances with pre-loaded software and hardware FPGA designs for data analytics, machine learning, and other applications, and deploy them directly on the AWS Elastic Cloud Compute (EC2) platform. AWS Marketplace products are usually delivered as Amazon Machine Images (AMIs), each of which provides the virtual machine setup, system environment settings, and all the required programs for the application that is being sold. AWS Marketplace instances which use FPGAs naturally use PCIe to communicate between the software and the hardware of the purchased instance.

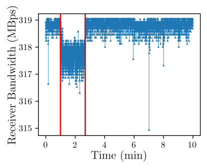
As we demonstrate for the first time, PCIe activity patterns can be detected across different, logically isolated VMs, leading to a new side channel in the FPGA-accelerated instances, and giving us insights into the AMIs running in co-located VMs. To show the new side channel between different VMs, we first introduce an AMI we purchased to test as the victim software and hardware design (Section V-A), and then discuss the recovery of potentially private information from the victim AMI's activity by running a co-located receiver VM that monitors the victim's PCIe activity (Section V-B).

A. Experimental Setup

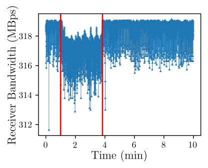
Among the different hardware accelerator solutions for cloud FPGAs, in this section we target video processing using the DeepField AMI, which leverages FPGAs to accelerate



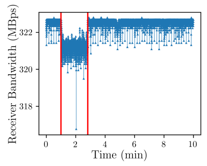
(a) 360p, 15 FPS



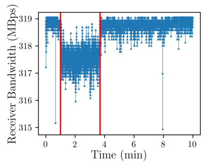
(b) 480p, 15 FPS



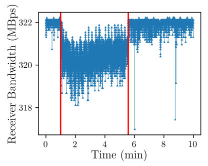
(c) 720p, 15 FPS



(d) 360p, 30 FPS



(e) 480p, 30 FPS



(f) 720p, 30 FPS

Fig. 8: PCIe bandwidth traces collected by the attacker while the victim runs the DeepField AMI to perform VSR conversions with input videos of different resolutions and frame rates. Within each subfigure, the red lines label the start and end of the VSR conversion on the FPGA. Unlike the other experiments in this paper, the receiver's absolute bandwidth varies between 310–330 MBps, as it uses a different setup (Amazon Linux release 2 rather than CentOS).

the Video Super-Resolution (VSR) algorithm to convert low-resolution videos to high-resolution ones [18]. The DeepField AMI is based on Amazon Linux release 2 (Karoo), and sets up the system environment to make use of the proprietary, pre-trained neural network models [18]. To use the AMI, the virtual machine software first loads the Amazon FPGA Image (AFI) onto the associated FPGA using the `load_afi` command to set up the FPGA board on the F1 instance. The `ffmpeg` program, which is customized for the FPGA platform, is called to convert an input video of no more than 1280×720 in resolution to a high-resolution video with a maximum output resolution of 3840×2160. As discussed above, the DeepField AMI handles all of the software and provides the FPGA image for the acceleration of the VSR algorithm. Users do not know how the FPGA logic operates, as it is provided as a pre-compiled AFI. However, PCIe contention allows us to reveal potentially private information from such example AMIs by running our attacker VM to measure the PCIe activity of the victim. In particular, this type of high-performance computing for image and video processing inevitably requires massive data transfers between the FPGA and the host processor through PCIe. These AMI behaviors are reflected in the PCIe bandwidth trace.

For our experiments, we first launch a group of `f1.2xlarge` instances running the DeepField AMI to find a co-located F1 instance pair using our PCIe contention approach of Section IV. After verifying that the attacker and the victim are co-located, we set up the attacker VM in monitoring mode, which continuously measures the PCIe bandwidth, similar to the receiver in the covert-channel setup. The monitoring program has been configured to measure bandwidth with a measurement duration of $\delta = 20\,\text{ms}$ and a data transfer duration of $d = 18\,\text{ms}$.

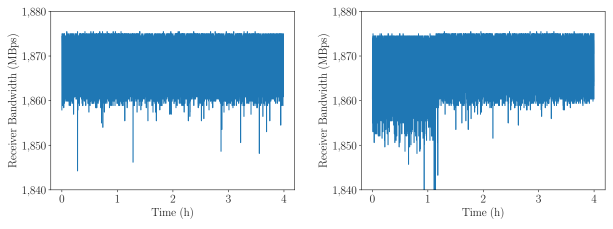
The victim VM then runs the unmodified DeepField AMI to convert different lower-resolution videos to higher-resolution ones using the `ffmpeg` program. In our experiments, each run of the DeepField AMI takes approximately $5\,\text{min}$, and each bandwidth trace in the attacker VM lasts for $10\,\text{min}$, thus covering both the conversion process, as well as periods of inactivity. As discussed in Section V-B, by comparing the bandwidth traces among the different experiments, we observe that we can infer information about a) whether the victim is actively in the process of converting a video, and b) deduce certain parameters of the videos.

B. Private Information Leakage from AMIs

We now show that private information regarding the activities of co-located instances can be revealed through the PCIe bandwidth traces. Figure 8 shows the PCIe bandwidth measured by the attacker while the victim is running the DeepField AMI on an `f1.2xlarge` instance. We test different input video files, with three different resolutions (360p, 480p, and 720p) and two frame rates of 15 and 30 frames-per-second (FPS). All videos have a 16:9 aspect ratio, and, except for the resolutions and frame rates, the contents of the input video files are otherwise identical. The output video produced for each conversion always has a resolution of 3840×2160, but maintains the same frame rate as the original input. The beginning and ending of the VSR conversion on the FPGA can be clearly seen in Figure 8, where vertical red lines delineating the start and end of the process have been added for clarity. We observe that the PCIe bandwidth drops during the conversion, and that runtime is reduced as the input resolution or the input frame rate decrease. For example, the runtime for a 720p, 30 FPS video (Figure 8f) is approximately twice as long as for a 15 FPS one (Figure 8c).

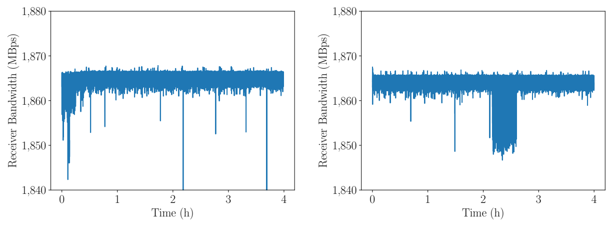
VI. Long-Term PCIe Monitoring

In this section, we present the results of measuring the PCIe bandwidth for two on-demand `f1.2xlarge` instances in the `us-east-1` region (availability zone `e`). These experiments took place between 5pm on April 25, 2021 and 2am on April 26 (Eastern Time, as `us-east-1` is located in North Virginia). For both sets of four-hour measurements, the first `f1.2xlarge` instance (Figure 9) is measuring with a transmission duration of $d = 4\,\text{ms}$ and a measurement duration of $\delta = 5\,\text{ms}$, while the second instance (Figure 10) has $d = 18\,\text{ms}$ and $\delta = 20\,\text{ms}$. For the first instance, the PCIe link remains mostly idle during the evening (Figure 9a), but experiences contention in the first night hour (Figure 9b). The second instance instead appears to be co-located with other FPGAs that make heavier use of their PCIe bandwidth. During the evening measurements (Figure 10a), the PCIe bandwidth drops momentarily below $1{,}200\,\text{MBps}$ during the third hour and below $800\,\text{MBps}$ during the fourth hour. It also experiences sustained contention in the third hour of the night measurement (Figure 10b). Although the bandwidth in the two instances is comparable, the $5\,\text{ms}$ measurements are noisier compared to the $20\,\text{ms}$ ones. Finally note that, generally, our covert-channel code results in bandwidth drops of over $800\,\text{MBps}$,



(a) April 25, after 5pm (b) April 25, after 9pm

Fig. 9: Long-term PCIe-based data center monitoring between the evening of April 25 and the early morning of April 26, with $d = 4\,\text{ms}$ and $\delta = 5\,\text{ms}$ on an `f1.2xlarge` on-demand instance.



(a) April 25, after 5pm (b) April 25, after 9pm

Fig. 10: Long-term PCIe-based data center monitoring on a different `f1.2xlarge` on-demand instance with $d = 18\,\text{ms}$ and $\delta = 20\,\text{ms}$.

while the activity of other users tends to cause drops of less than $50\,\text{MBps}$, suggesting that noise from external traffic has minimal impact on our channel.

VII. SSD-Based Co-Location

Another shared resource that can lead to contention is the SSD storage that F1 instances can access. The public specification of F1 instances notes that `f1.2xlarge` instances have access to $470\,\text{GB}$ of Non-Volatile Memory Express (NVMe) SSD storage, `f1.4xlarge` have $940\,\text{GB}$, and `f1.16xlarge` have $4 \times 940\,\text{GB}$ [9]. This suggests that F1 servers have four separate $940\,\text{GB}$ SSD drives, each of which can be shared between two `f1.2xlarge` instances. In this section, we confirm our hypothesis that one SSD drive can be shared between multiple instances, and explain how this fact can be exploited to reverse-engineer the PCIe topology and co-locate VM instances. The SSD contention we uncover can also be used for a slow, but reliable, covert channel, or to degrade the performance of other users, akin to a Denial-of-Service (DoS) attack. We also demonstrate the existence of FPGA-to-SSD contention, which is likely the result of the SSD going through the same PCIe switch, as shown in Figure 2. This topology remains consistent with the one publicly described for GPU-based P4d instances [7], which appear to be architecturally similar to F1 instances.

A. SSD-to-SSD Contention

SSD contention is tested by measuring the bandwidth of the SSD by using the `hdparm` command with its `-t` option, which performs disk reads without any data caching [33]. Measurements are averaged over repeated reads of $2\,\text{MB}$ chunks from the disk in a period of 3 seconds. When the server is otherwise idle, `hdparm` reports the SSD read bandwidth to be over $800\,\text{MBps}$. However, when the other `f1.2xlarge` instance that shares the same SSD stresses it using the `stress` command [46] with the `--io 4 --hdd 4` parameters, the bandwidth drops below $50\,\text{MBps}$. The `stress` command with the parameters above results in 4 threads calling `sync` (to stress the read buffers) and another 4 threads calling `write` and `unlink` (to stress write performance). The total number of threads is kept to 8, to match the number of vCPUs allocated to an `f1.2xlarge` instance, while all FPGAs remain idle during these experiments.

This non-uniform SSD behavior can be used for a robust covert channel with a bandwidth of $0.125\,\text{bps}$ with 100% accuracy. Specifically, for a transmission of bit 1, `stress` is called for 7 seconds, while for a transmission of bit 0, the transmitter remains idle. The receiver uses `hdparm` to measure its SSD's bandwidth, and can distinguish between contention and no-contention of the SSD resources (i.e., bits 1 and 0 respectively) using a simple threshold. The period of 8 seconds per bit also accounts for 1 second of inactivity in every transmission, allowing the disk usage to return to normal.

The same mechanism can be exploited to deteriorate the performance of other tenants for DoS attacks. It can further co-locate instances on an even more fine-grained level than was previously possible. To accomplish this, we rent several `f1.2xlarge` instances until we find four which form a full NUMA node through the PCIe-based co-location approach of Section IV. We then stress the SSD in one of the four instances, and measure the SSD performance in the remaining three. We discover two pairs of instances with mutual SSD contention, which supports our hypothesis, and is also consistent with the PCIe topology for other instance types [7].

The fact that SSD contention only exists between two `f1.2xlarge` instances can be beneficial for adversaries: when the covert-channel receiver and the transmitter are scheduled on two instances that share an SSD, they can communicate without interference from other tenants in the same NUMA node.[3]

B. FPGA-to-SSD Contention

To formalize the above observations, we use the methodology described in Section IV to find four co-located `f1.2xlarge` instances in the same NUMA node. Then, for each pair of instances, we repeatedly run `hdparm` in the "receiver" instance for a period of 3 minutes, and then in the transmitter instance, a) at the one minute mark run `stress` for $30\,\text{s}$, and b) at the two minute mark use our FPGA-based covert-channel code as a stressor which constantly transmits the bit 1 during each measurement period for another $30\,\text{s}$.

The results of these experiments are summarized in Figure 11. During idle periods, the SSD bandwidth is approxi-

[3]Assuming that slots within a server are assigned randomly, the probability of getting instances with shared SSDs given that they are already co-located in the same NUMA node is 33%: out of the three remaining slots in the same NUMA node, exactly one slot can be in an instance that shares the SSD.

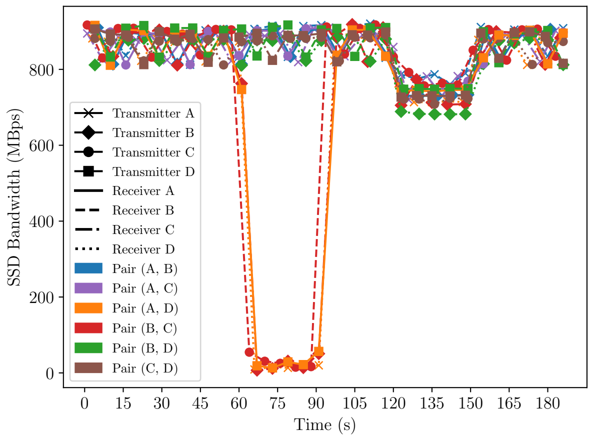


Fig. 11: NVMe SSD bandwidth for all transmitter and receiver pairs in a NUMA node, as measured by `hdparm`. Running `stress` between seconds 60 to 90 causes a bandwidth drop in exactly one other instance in the NUMA node, while running the FPGA-based PCIe stressor (between seconds 120 and 150) reduces the SSD bandwidth in all cases.

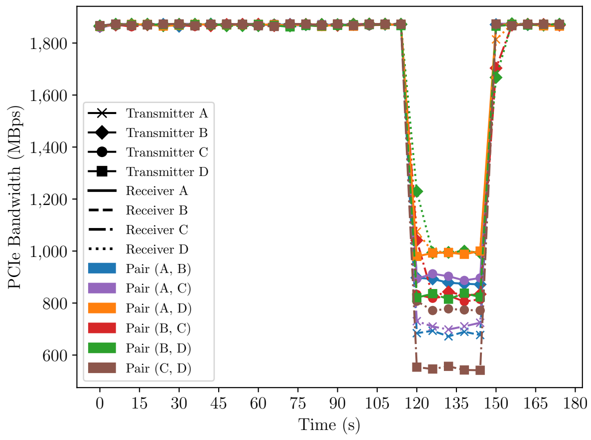


Fig. 12: FPGA PCIe bandwidth for all transmitter and receiver pairs in a NUMA node, as measured by our covert-channel receiver. Running `stress` between seconds 60 to 90 does not cause a bandwidth drop, but running the FPGA-based PCIe stressor (between seconds 120 and 150) reduces the bandwidth in all cases.

mately 800–900 MBps. However, for the two instances with SSD contention, i.e., pairs (A, D) and (B, C), the bandwidth drops to as low as 7 MBps while the `stress` command is running (the bandwidth for the other instance pairs remains unmodified). When the FPGA-based PCIe stressor is enabled, the SSD bandwidth reported by `hdparm` is reduced in a measurable way to approximately 700 MBps.

We further test for the opposite effect, i.e., whether stressing the SSD can cause a measurable difference to the FPGA-based PCIe performance. We again stress the SSD between 60–90 s, and stress the FPGA between 120–150 s. As the results of Figure 12 show, the PCIe bandwidth drops from almost 1.8 GBps to approximately 500–1,000 MBps when the FPGA-stressor is enabled, but there is no significant difference in performance when the SSD-based stressor is turned on. This is likely because the FPGA-based stressor can more effectively saturate the PCIe link, while the SSD-based stressor seems to be limited by the performance of the hard drive itself, whose bandwidth when idle (800 MBps) is much lower than that of the FPGA (1.8 GBps). In summary, using the FPGA as a PCIe stressor can cause the SSD bandwidth to drop, but the converse is not true, since there is no observable influence on the FPGA PCIe bandwidth as a result of SSD activity.

VIII. DRAM-Based Cross-NUMA Co-Location

DRAM decay is known to depend on the temperature of the DRAM chip and its environment [48], [49]. Since the FPGAs in cloud servers have direct access to the on-board DRAM, they can be used as sensors for detecting and estimating the temperature around the FPGA boards, supplementing PCIe-traffic-based measurements.

Figure 13 summarizes how the DRAM decay of on-board chips can be used to monitor thermal changes in the data center. When a DRAM module is being initialized with some data, the DRAM cells will become charged to store the values,

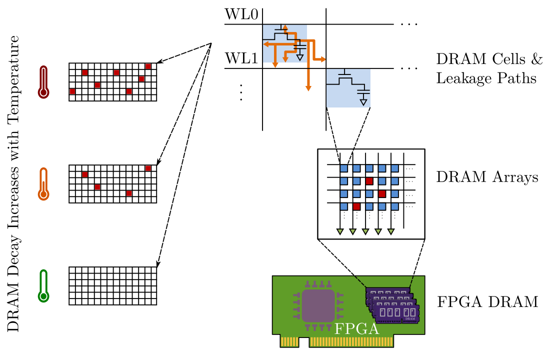
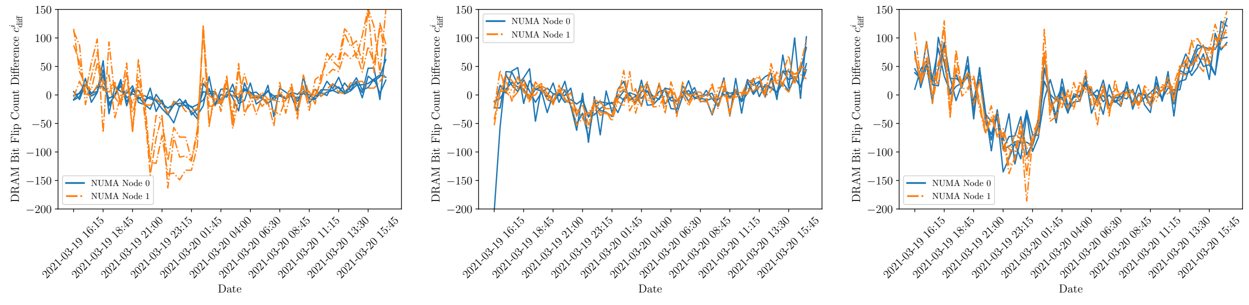

DRAM Decay Increases with Temperature
WL0
WL1
DRAM Cells & Leakage Paths
DRAM Arrays
FPGA DRAM
FPGA

Fig. 13: By alternating between AFIs that instantiate DRAM controllers or leave them unconnected, the decay rate of DRAM cells can be measured as a proxy for environmental temperature monitors [40].

with true cells storing logical 1s as charged capacitors, and anti-cells storing them as depleted capacitors. Typically, true and anti-cells are paired, so initializing the DRAM to all ones will ensure only half of the DRAM cells will be charged, even if the actual location of true and anti-cells is not known.

After the data has been written to the DRAM and the cells have been charged, the DRAM refresh is disabled. Disabling DRAM refresh in the server itself is not possible as the physical hardware on the server is controlled by the hypervisor, not the users. However, the FPGA boards have their own DRAMs. By programming the FPGAs with AFIs that do and do not have DRAM controllers, disabling of the DRAM refresh can be emulated, allowing the DRAM cells to decay [42]. Eventually, some of the cells will lose enough charge to "flip" their value (for example, data written as 1 becomes 0 for true cells, since the charge has dissipated).

DRAM data can then be read after a fixed time T_{decay}, which is called the decay time. The number of flipped cells during this time depends on the temperature of the DRAM and its environment [49], and can therefore produce coarse-grained DRAM-based temperature sensors of F1 instances.



(a) Server A

(b) Server B

(c) Server C

Fig. 14: DRAM decay traces from three `f1.16xlarge` instances (24 FPGAs in total) for a period of 24 hours, using the differences between successive measurements c^i_{diff} as the comparison metric, which results in the highest co-location accuracy of 96%. Within each server, measurements from slots in the same NUMA node have been drawn in the same style.

Prior work [43] and this paper have so far focused on information leaks due to shared resources within a NUMA node, but did not attempt to co-locate instances that are in the same physical server, but belong to different NUMA nodes. In this section, we propose such a methodology that uses the boards' thermal signatures, which are obtained from the decay rates of each FPGA's DRAM modules. To collect these signatures, we use the method and code provided by Tian et al. [42] to alternate between bitstreams that instantiate DRAM controllers and ones that leave them unconnected to initialize the memory and then disable its refresh rate. When two instances are in the same server, the temperatures of all 8 FPGAs in an `f1.16xlarge` instance (and by extension the DRAM thermal signatures) are highly correlated. However, when the instances come from different servers, the decay rates are different, and thus contain distinguishable patterns that can be used to classify the two instances separately. This insight can be used to find FPGA instances that are co-located in the same server, even if they span different NUMA nodes.

A. *Setup & Evaluation*

Our method for co-locating instances within a server has two aspects to it: first, we show that we can successfully identify two FPGA boards as being in the same server with high probability using their DRAM decay rates, and then we show that by using PCIe-based co-location we can build the full profile of a server, and identify all eight of its FPGA boards, even if they are in different NUMA nodes. More specifically, we use the open-source software by Tian et al. [42] to collect DRAM decay measurements for several FPGAs over a long period of time and then find which FPGAs' DRAM decay patterns are the "closest".

To validate our approach, we rent three `f1.16xlarge` instances (a total of 24 FPGAs) for a period of 24 hours, and measure how "close" each pair of FPGA traces is by calculating the total distance between their data points over the entire measurement period for three different metrics. The first metric compares the raw number of bit flips from the DRAM decay measurement c^i_{raw} directly. The second approach normalizes the data to fit in the $[-1, 1]$ range, i.e., $c^i_{\text{norm}} = (2c^i_{\text{raw}} - m - M)/(M - m)$, where $m = \min_i c^i_{\text{raw}}$ and $M = \max_i c^i_{\text{raw}}$. We finally also propose an alternative metric, shown in Figure 14 for all three servers, which takes the difference between successive raw measurements, i.e., $c^i_{\text{diff}} = c^i_{\text{raw}} - c^{i-1}_{\text{raw}}$.

The raw data metric has an accuracy of 75%, the normalized metric is 71% accurate, while the difference metric succeeds in correctly pairing all FPGAs except for one, for an accuracy of 96%.[4] Note that when using our metric, if FPGA A is the closest to FPGA B, then B is *not* necessarily the closest to A. However, if FPGA A is closest to B and B is closest to C, then A, B, and C are all in the same server.

In the experiments of Figure 14, this approach places slots 0–4 of server A together (along with, mistakenly, slot 0 of server B), slots 5–7 of server A as a second group, slots 1–7 of server B as one server, and slots 0–3 and 4–7 of server C as the two final groups. Consequently, our method successfully identifies the six NUMA nodes without making use of PCIe contention at all.

However, by using insights about the NUMA nodes that can be extracted through our PCIe-based experiments, the accuracy and reliability of this method can be further increased. For example, slot 0 of server B could already be placed in the same NUMA node as slots 1–3 using PCIe-based co-location. Leveraging the PCIe-based co-location method, if the "closest" FPGA is known to be in the same NUMA node due to PCIe contention, and the second-closest FPGA (not in the same NUMA node according to PCIe contention) is only farther by at most 1% compared to the closest FPGA, then this second-closest FPGA can be identified as belonging to the second NUMA node of the same server. In the experiment of Figure 14, this approach successfully groups all FPGAs in the three tested servers without errors.

IX. Related Work

Since the introduction of FPGA-accelerated cloud computing about five years ago, a number of researchers have been

[4]Shorter measurement periods still result in high accuracies. For example, using the DRAM data from the first 12 hours results in only one additional FPGA mis-identification, for an accuracy of 92%.

exploring the security aspects of FPGAs in the cloud. A key feature differentiating such research from prior work on FPGA security outside of cloud environments is the threat model, which assumes remote attackers without physical access to or modifications of the FPGA boards. This section summarizes selected work that is applicable to the cloud setting, leaving traditional FPGA security topics to existing books [28] or surveys [29], [34].

A. PCIe-Based Threats

The Peripheral Component Interconnect Express (PCIe) standard provides a high-bandwidth, point-to-point, full-duplex interface for connecting peripherals within servers. Existing work has shown that PCIe switches can cause bottlenecks in multi-GPU systems [17], [20], [21], [36], [37], leading to severe stalls due to their scheduling policy [30]. In terms of PCIe contention in FPGA-accelerated cloud environments, prior work has shown that different driver implementations result in different overheads [45], and that changes in PCIe bandwidth can be used to co-locate different instances on the same server [43]. However, no work had used PCIe contention for covert-channel attacks locally, or on FPGA-accelerated clouds. Moreover, by going beyond just PCIe, our work is able to deduce cross-NUMA-node co-location using the DRAM thermal fingerprinting approach.

B. Power-Based Threats

Computations that cause data-dependent power consumption can result in information leaks that can be detected even by adversaries without physical access to the device under attack. For example, it is known that a shared power supply in a server can be used to leak information between different FPGAs, where one FPGA modulates power consumption and the other measures the resulting voltage fluctuations [24]. However, such work results in low transmission rates (below $10\,\mathrm{bps}$), and has only been demonstrated in a lab environment.

Other work has shown that it is possible to develop stressor circuits which modulate the overall power consumption of an FPGA and generate a lot of heat, for instance by using ring oscillators or transient short circuits [1], [2], [26]. Such ideas could be used to prematurely age FPGAs, which has been shown to be a risk for FPGAs exposed to excessive heat for an extended period of time [15]. Our work has instead focused on information leaks and non-destructive reverse-engineering of the infrastructure.

C. Thermal-Based Threats

It is now well-known that it is possible to implement temperature sensors suitable for thermal monitoring on FPGAs using ring oscillators [19], whose frequency drifts in response to temperature variations [31], [32], [44], [50]. A receiver FPGA could thus use a ring oscillator to observe the ambient temperature of a data center. For example, existing work [41] has explored a new type of temporal thermal attack: heat generated by one circuit can be later observed by other circuits that are loaded onto the same FPGA. This type of attack is able to leak information between different users of an FPGA who are assigned to the same FPGA over time. However, the bandwidth of temporal attacks is low (less than $1\,\mathrm{bps}$), while our covert channels can reach a bandwidth of up to $2\,\mathrm{kbps}$.

D. DRAM-Based Threats

Recent work has shown that direct control of the DRAM connected to the FPGA boards can be used to fingerprint them [42]. This can be combined with existing work [43] to build a map of the cloud data centers where FPGAs are used. Such fingerprinting does not by itself, however, help with cross-VM covert channels, as it does not provide co-location information. By contrast, our PCIe, SSD, and DRAM approaches are able to co-locate instances in the same server and enable cross-VM covert channels and information leaks.

E. Multi-Tenant Security

This work has focused on the single-tenant setting, where each user gets full access to the FPGA, and thus reflects the current environment offered by cloud providers. However, there is also a large body of security research in the multi-tenant context, where a single FPGA is shared by multiple, logically (and potentially physically) isolated users. For example, recent work in this area has shown that crosstalk between wires inside the FPGA chips can be used to leak information [22], while power-based attacks can lead to covert-channel [23] and side-channel [25] attacks. Our work on PCIe, SSD, and DRAM threats is orthogonal to such work, but is directly applicable to current cloud FPGA deployments.

X. Conclusion

This paper demonstrated the first covert- and side-channel attacks between separate users in a public, FPGA-accelerated cloud computing setting. In addition to making use of contention of the PCIe bus, this work identified new, alternative SSD- and DRAM-based co-location mechanisms, which can be used to detect FPGAs which are in the same server, even if they are on separate NUMA nodes. More generally, this work demonstrated that malicious adversaries can use PCIe monitoring to observe the data center server activity, breaking the separation of privilege that isolated VM instances are supposed to provide. With more types of accelerators becoming available on the cloud, including FPGAs, GPUs, and TPUs, PCIe-based threats are bound to become a key aspect of cross-user attacks. Overall, our insights showed that low-level, direct hardware access to PCIe, SSD, and DRAM hardware creates new attack vectors that need to be considered by both users and cloud providers alike when deciding how to trade off performance, cost, and security for their designs: even if the endpoints of computations are assumed to be secure, the shared nature of cloud infrastructures poses new challenges that need to be addressed.

References

[1] A. Agne, H. Hangmann, M. Happe, M. Platzner, and C. Plessl, "Seven recipes for setting your FPGA on fire—A cookbook on heat generators," *Microprocessors and Microsystems*, vol. 38, no. 8, pp. 911–919, Nov. 2014.

[2] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2019.

[3] Alibaba Cloud, "Instance families," https://www.alibabacloud.com/help/doc-detail/25378.htm#f3, 2021, Accessed: 2021-05-08.

[4] Amazon Web Services, "Developer preview—EC2 instances (F1) with programmable hardware," https://aws.amazon.com/blogs/aws/developer-preview-ec2-instances-f1-with-programmable-hardware/, 2016, Accessed: 2021-05-08.

[5] ——, "The agility of F1: Accelerate your applications with custom compute power," https://d1.awsstatic.com/Amazon_EC2_F1_Infographic.pdf, 2018, Accessed: 2021-05-08.

[6] ——, "F1 FPGA application note: How to use write combining to improve PCIe bus performance," https://github.com/awslabs/aws-fpga-app-notes/tree/master/Using-PCIe-Write-Combining, 2019, Accessed: 2021-05-08.

[7] ——, "Amazon EC2 P4d instances deep dive," https://aws.amazon.com/blogs/compute/amazon-ec2-p4d-instances-deep-dive/, 2020, Accessed: 2021-05-08.

[8] ——, "Official repository of the AWS EC2 FPGA hardware and software development kit," https://github.com/aws/aws-fpga/tree/v1.4.15, 2020, Accessed: 2021-05-08.

[9] ——, "Amazon EC2 instance types," https://aws.amazon.com/ec2/instance-types/, 2021, Accessed: 2021-05-08.

[10] ——, "AWS marketplace," https://aws.amazon.com/marketplace, 2021, Accessed: 2021-05-08.

[11] ——, "AWS shell interface specification," https://github.com/aws/aws-fpga/blob/master/hdk/docs/AWS_Shell_Interface_Specification.md, 2021, Accessed: 2021-05-08.

[12] ——, "CL_DRAM_DMA custom logic example," https://github.com/aws/aws-fpga/tree/master/hdk/cl/examples/cl_dram_dma, 2021, Accessed: 2021-05-08.

[13] ——, "F1 FPGA application note: How to use the PCIe peer-2-peer version 1.0," https://github.com/awslabs/aws-fpga-app-notes/tree/master/Using-PCIe-Peer2Peer, 2021, Accessed: 2021-05-08.

[14] ——, "FPGA developer AMI," https://aws.amazon.com/marketplace/pp/B06VVYBLZZ, 2021, Accessed: 2021-05-08.

[15] A. Amouri, F. Bruguier, S. Kiamehr, P. Benoit, L. Torres, and M. Tahoori, "Aging effects in FPGAs: An experimental analysis," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[16] Baidu Cloud, "FPGA cloud compute," https://cloud.baidu.com/product/fpga.html, 2021, Accessed: 2021-05-08.

[17] G. Baker and C. Lupo, "TARUC: A topology-aware resource usability and contention benchmark," in *ACM/SPEC International Conference on Performance Engineering (ICPE)*, 2017.

[18] BLUEDOT, "DeepField-SR video super resolution hardware accelerator," https://www.xilinx.com/products/acceleration-solutions/deepField-sr.html, 2021, Accessed: 2021-05-08.

[19] E. Boemo and S. López-Buedo, "Thermal monitoring on FPGAs using ring-oscillators," in *International Workshop on Field-Programmable Logic and Applications (FPL)*, 1997.

[20] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (SHOC) benchmark suite," in *Workshop on General-Purpose Processing on Graphics Processing Units (GPGPU)*, 2010.

[21] I. Faraji, S. H. Mirsadeghi, and A. Afsahi, "Topology-aware GPU selection on multi-GPU nodes," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016.

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

[23] ——, "Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs," in *IEEE International Conference on Computer Design (ICCD)*, 2019.

[24] ——, "C^{3}APSULe: Cross-FPGA covert-channel attacks through power supply unit leakage," in *IEEE Symposium on Security and Privacy (S&P)*, 2020.

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

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

[27] Huawei Cloud, "FPGA accelerated cloud server," https://www.huaweicloud.com/en-us/product/fcs.html, 2021, Accessed: 2021-05-08.

[28] T. Huffmire, C. Irvine, T. D. Nguyen, T. Levin, R. Kastner, and T. Sherwood, *Handbook of FPGA Design Security*, 1st ed. Springer, 2010.

[29] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud FPGAs: A survey," arxiv.org/abs/2005.04867, 2020, Accessed: 2021-05-08.

[30] C. Li, Y. Sun, L. Jin, L. Xu, Z. Cao, P. Fan, D. Kaeli, S. Ma, Y. Guo, and J. Yang, "Priority-based PCIe scheduling for multi-tenant multi-GPU systems," *IEEE Computer Architecture Letters (LCA)*, vol. 18, no. 2, pp. 157–160, Jul. 2019.

[31] S. López-Buedo, J. Garrido, and E. Boemo, "Thermal testing on reconfigurable computers," *IEEE Design & Test of Computers (D&T)*, vol. 17, no. 1, pp. 84–91, Jan. 2000.

[32] ——, "Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems," *IEEE Transactions on Components and Packaging Technologies (TCAPT)*, vol. 25, no. 4, pp. 561–566, Dec. 2002.

[33] M. Lord, "hdparm," https://sourceforge.net/projects/hdparm/, 2021, Accessed: 2021-05-08.

[34] S. S. Mirzargar and M. Stojilović, "Physical side-channel attacks and covert communication on FPGAs: A survey," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2019.

[35] Nimbix, Inc., "Xilinx Alveo accelerator cards," https://www.nimbix.net/alveo, 2021, Accessed: 2021-05-08.

[36] D. Schaa and D. Kaeli, "Exploring the multiple-GPU design space," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2009.

[37] K. Spafford, J. S. Meredith, and J. S. Vetter, "Quantifying NUMA and contention effects in multi-GPU systems," in *Workshop on General-Purpose Processing on Graphics Processing Units (GPGPU)*, 2011.

[38] T. Sugawara, K. Sakiyama, S. Nashimoto, D. Suzuki, and T. Nagatsuka, "Oscillator without a combinatorial loop and its threat to FPGA in data centre," *Electronics Letters*, vol. 15, no. 11, pp. 640–642, May 2019.

[39] Tencent Cloud, "FPGA cloud server," https://cloud.tencent.com/product/fpga, 2021, Accessed: 2021-05-08.

[40] S. Tian, A. Krzywosz, I. Giechaskiel, and J. Szefer, "Cloud FPGA security with RO-based primitives," in *International Conference on Field-Programmable Technology (FPT)*, 2020.

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

[42] S. Tian, W. Xiong, I. Giechaskiel, K. B. Rasmussen, and J. Szefer, "Fingerprinting cloud FPGA infrastructures," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020.

[43] S. Tian, W. Xiong, I. Giechaskiel, and J. Szefer, "Cloud FPGA cartography using PCIe contention," in *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2021.

[44] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in FPGAs," in *IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2008.

[45] X. Wang, Y. Niu, F. Liu, and Z. Xu, "When FPGA meets cloud: A first look at performance," *IEEE Transactions on Cloud Computing (TCC)*, 2020.

[46] A. P. Waterland, "stress," https://web.archive.org/web/20190502/https://people.seas.harvard.edu/~apw/stress/, 2014, Accessed: 2021-05-08.

[47] Xilinx, Inc., "UltraScale+ FPGAs: Product tables and product selection guides," https://www.xilinx.com/support/documentation/selection-guides/ultrascale-plus-fpga-product-selection-guide.pdf, 2021, Accessed: 2021-05-08.

[48] W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser, and J. Szefer, "Spying on temperature using DRAM," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[49] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Run-time accessible DRAM PUFs in commodity devices," in *Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2016.

[50] C.-E. Yin and G. Qu, "Temperature-aware cooperative ring oscillator PUF," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2009.