Cloud FPGA Cartography Using PCIe Contention

Shanquan Tian*, Ilias Giechaskiel*, Wenjie Xiong, and Jakub Szefer

29th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), Orlando, Florida, United States of America, May 2021

Cloud FPGA Cartography using PCIe Contention

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

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

Wenjie Xiong
Yale University
New Haven, CT, USA
wenjie.xiong@yale.edu

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

***Abstract*—Public cloud infrastructures allow for easy, on-demand access to FPGA resources. However, the low-level, direct access to the FPGA hardware exposes the infrastructure providers to new types of attacks. Prior work has shown that it is possible to uniquely identify the underlying hardware by creating fingerprints of the different FPGA instances that users rent from a cloud provider, but such work was not able to actually map the cloud FPGA infrastructure itself. Meanwhile, this paper demonstrates that it is possible to reverse-engineer the co-location of FPGA boards inside a cloud FPGA server using PCIe contention. Specifically, this work deduces the Non-Uniform Memory Access (NUMA) locality of FPGA boards within a server by analyzing their mutual PCIe contention during simultaneous use of the PCIe bus. In addition, experiments conducted in data centers located in several geographic regions and repeated at different times are used to calculate the probability that cloud providers allocate FPGA boards co-located in the same server to a user. This paper thus shows that it is possible to map cloud FPGA infrastructures, and learn how FPGA instances are physically co-located within a server. Consequently, this paper also highlights the importance of mitigating these novel avenues for reverse-engineering and mapping of cloud FPGA setups, as they can reveal insights about the cloud infrastructure itself, or assist other single- and multi-tenant attacks.**

***Index Terms*—Cloud FPGAs, FPGA Security, PCIe Contention, FPGA Fingerprinting, Cloud Cartography**

I. Introduction

Over the past 5 years, several cloud providers have made Intel and Xilinx FPGAs available in their infrastructures, with Amazon Web Services (AWS) F1 instances being the most widely accessible [4]. More recently, FPGAs are being offered from other cloud providers such as Alibaba [1], Baidu [6], Huawei [16], Nimbix [27], or Tencent [38]. This widespread adoption of public cloud FPGAs has made the on-demand hardware acceleration of analytics in financial, big data, and anomaly detection applications possible [3]. However, it has also resulted in serious implications for the security of the cloud infrastructure itself.

Some cloud providers, including AWS, have taken steps to protect their infrastructure by blocking known potential threats such as Ring Oscillators (ROs), which can be used to waste power [28], cause hardware faults [18], detect voltage [14] and temperature [40] variations, or produce unique fingerprints of the underlying hardware in the form of Physical Unclonable Functions (PUFs) [39]. However, recent research has shown these countermeasures to be insufficient, not only because of alternative RO structures that can bypass these defense mechanisms [13], [37], but also due to DRAM-based [41] PUFs that can be used to fingerprint the FPGA instances without the use of ROs. All the existing attacks and defenses have not considered how attackers could gain information about the location of the FPGAs inside the data center, which is the focus of this work.

In particular, this paper presents a new approach for mapping cloud FPGA infrastructures, based on PCIe contention. The main insight behind our research is that memory accesses between the host computer and an FPGA board become a bottleneck when two or more FPGAs from the same Non-Uniform Memory Access (NUMA) node within a server are accessing memory simultaneously. The accesses are done over PCIe, and we show that it is possible to influence the PCIe bandwidth by running an FPGA memory stressor, and to observe this change in bandwidth by running a separate FPGA memory tester on a different FPGA board in the server. Using this approach, we can determine which FPGA slots within a server can mutually interfere due to their NUMA node locality. This can be done even without the use of Physical Unclonable Functions (PUFs) or banned circuits such as ring oscillators, and without any knowledge of the data center architecture.

We show that in `f1.16xlarge` AWS instances, FPGAs in slots 0–3 or 4–7 interfere with each other, and thus conclude that they form separate NUMA nodes. We then use data from dozens of `f1.2xlarge` instances that have been rented one after the other and determine that successive instances often (but not always) belong to the same NUMA locality. The findings are confirmed for `f1.4xlarge` instances and across different data center regions for both instance types. We perform additional experiments with `f1.2xlarge` instances to calculate the probability of renting the same FPGA or FPGAs within the same NUMA locality across time, therefore fingerprinting and mapping the cloud infrastructure on a very fine-grained level, which was previously impossible. In fact, we find that a given FPGA board can be reused between the three F1 instance types available in AWS. This corrects prior work claiming no such overlap [41].

As our work exposes a fundamental infrastructure issue, it is critical to understand these threats, so that they can be mitigated. Simply focusing on the security of the FPGA chip itself, but ignoring other infrastructure components such as the PCIe bus, leaves cloud FPGAs open to new vulnerabilities.

[†] The first two authors contributed equally to this work.

* This work was supported in part by NSF grant 1901901.

To help address these issues, we disclosed our findings to the Amazon Web Services security and FPGA teams prior to publication.

A. Contributions

In summary, the contributions of this paper are as follows:

1) We identify that there is contention between FPGA slots in cloud servers by analyzing the impact of simultaneous memory accesses over PCIe in `f1.16xlarge` AWS instances. Our analysis exposes new details about the AWS servers and their NUMA localities.
2) Having determined that contention exists, we jointly experiment with PCIe contention and DRAM PUF fingerprinting on `f1.2xlarge` and `f1.4xlarge` instances across different AWS data center regions to reveal insights about the FPGA instance allocation algorithm used by the AWS cloud and overlaps between the physical hardware used for the different instance types.

B. Paper Organization

After summarizing the relevant background on the FPGA architecture of Amazon F1 instances, cloud FPGA security, and PCIe-related concepts (Section II), we present the threat model (Section III) and our experimental setup, which allows us to identify co-located FPGA instances within a server (Section IV). Next, the evaluation covers (a) `f1.16xlarge` experiments leading to the discovery of PCIe contention and NUMA localities, and (b) tests with `f1.2xlarge` and `f1.4xlarge` instances to calculate the probability that the allocation algorithm places different instances in the same server (Section V). We then discuss this work in the context of related research (Section VI), before concluding (Section VII).

II. Background

This section introduces the architecture of FPGAs on Amazon's cloud (Section II-A), and PCIe-related concepts (Section II-B) to contextualize the rest of the paper. An in-depth discussion of related work can be found in Section VI.

A. AWS F1 Instances

Amazon Web Services (AWS) offers FPGAs as part of its public cloud infrastructure in several geographical regions. These instances, or Virtual Machines (VMs), come in three flavors, with 1, 2, or 8 dedicated FPGAs per VM instance, called `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge` (the instance name is twice the number of FPGAs, so `f1.2xlarge` has 1 FPGA, while `f1.4xlarge` has 2, etc.). The total amount of resources allocated per VM increases proportionally with the number of attached FPGAs, providing 8 virtual CPUs (vCPUs) from Intel Xeon E5-2686 v4 (Broadwell) processors, 122 GiB of RAM, and 470 GB of NVMe SSD per FPGA [4]. Thus, e.g., `f1.16xlarge` instances have 64 vCPUs, 976 GiB of RAM and 3.7 TB of disk space.

Each FPGA board can communicate with the server over x16 PCIe Gen 3. In addition, each FPGA can access (via the programmable logic) four DDR4 DRAM chips on the FPGA board itself, which are separate from the server's DRAM. The FPGA DRAM comes with Error Correcting Codes (ECC), and a total of 16 GB of memory [4] for each FPGA. AWS F1 instances use 16 nm Virtex UltraScale+ XCVU9P chips [4], which internally contain over 1.1 million lookup tables (LUTs), 2.3 million flip-flops (FFs), and 6.8 thousand Digital Signal Processing (DSP) blocks [43]. It should be noted that `f1.16xlarge` instances use a dedicated PCIe fabric, which "lets the FPGAs share the same memory space and communicate with each other across the fabric at up to 12 Gbps in each direction" [4], suggesting that a server can consist of at most two CPUs and eight attached PCIe cards (FPGAs). As we show in this paper, servers indeed seem to have two Non-Uniform Memory Access (NUMA) locality nodes, each encompassing one CPU and four FPGAs. This is consistent with known server and PCIe designs, but is not publicly specified by Amazon.

AWS has taken a number of measures to ensure the security of its infrastructure. First, many details of the server architecture, such as the concrete FPGA board design, or the physical server internals, are not disclosed publicly (other than the information discussed above). Moreover, designs must interact with external interfaces through the cloud-provided "shell", which hides physical aspects such as clocking logic and I/O pinouts (including for PCIe and DRAM) [14], [41]. This restrictive shell interface further prevents users from accessing identifier resources (e.g., eFUSE and Device DNA primitives) that could be used to distinguish between different FPGA boards [14], [41]. 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 [13], [14].

B. PCIe Contention & NUMA Localities

The Peripheral Component Interconnect Express (PCIe) standard provides a high-bandwidth, serial, full-duplex interface. Unlike its parallel, bus-based PCI predecessor, each PCIe slot provides a point-to-point communication mechanism (link) that connects devices in a tree topology to the host CPU via the root complex and possibly through intermediate switches. A link is composed of 1, 2, 4, 8, 12, 16, or 32 pairs of RX and TX differential signals (lanes) that allow for higher throughput by interleaving transmissions (approximately 1 GBps per lane for PCIe 3.0) [35].

PCIe implements a credit-based flow-control protocol between link partners (e.g., the card and a switch, or a switch and the root complex) by sequencing the outgoing Transaction Layer Packets (TLPs) [35]. The credit tokens are therefore used to arbitrate and distribute bandwidth among competing incoming or outgoing link connections. As credits are point-to-point-based, it is possible for end-to-end traffic to be passing through a congested link (due to an unrelated flow), even when the individual endpoints can deal with higher bandwidth [23]. The slowdown that can be experienced by applications therefore not only depends on which devices are communicating, but also the PCIe topology of the given system [11], [23].

Although different PCIe slots on the motherboard might advertise the same number of lanes and identical performance, where they lie on the PCIe topology graph might be different, leading to non-uniform latency and bandwidth. For example, even desktop computers can have multiple PCIe root complexes (one in the chipset and one in the CPU integrated I/O Hub) that are linked over lower-bandwidth interfaces [22]. These interfaces, such as Intel's Quick Path Interconnect (QPI), can also be used for cache coherency protocols between different CPU sockets [25], which represent different root complexes and only have direct access to a subset of PCIe devices or DRAM chips.

This leads to Non-Uniform Memory Access (NUMA) latencies from a CPU to different PCIe or memory slots (or vice versa), and potentially increases contention of the interconnect resources [20], [24]. Memory and PCIe devices that have "the same access characteristics for a particular processor" are called NUMA nodes or localities [20]. Although for a given program it is possible to reduce the effects of NUMA problems, for instance by pinning threads to cores [24], PCIe contention can remain pronounced even when performing only "local" accesses within a NUMA node. In some systems, Translation Lookaside Buffer (TLB) misses in the Input-Output Memory Management Unit (IOMMU) may also increase latency and decrease throughput when issuing Direct Memory Access (DMA) requests [26].

PCIe congestion has in the past primarily been studied for multi-GPU systems, with PCIe switches becoming a bottleneck when handling traffic from multiple GPUs [7], [9], [11], [34], [36], in part due to a Round-Robin scheduling policy that can lead to severe stalls [21].

Despite the existing characterization of PCIe contention in setups with more than one GPU, no works have used PCIe contention to reverse-engineer cloud FPGA infrastructures. The closest research is a recent work by Wang et al. [42], who investigated the PCIe overhead of accessing FPGAs in cloud environments for different driver implementations. In our work, we instead more extensively study PCIe contention effects due to simultaneous memory accesses over multiple FPGAs at once, and use our results for architectural insights into the AWS server infrastructure.

III. Threat Model

This paper is concerned with reverse-engineering the physical server setup in cloud FPGAs. We assume the adversary can rent cloud FPGA instances in an attempt to get information about the hardware. The adversarial user is free to place and route their potentially malicious logic within the confines of their dedicated region on the FPGA chip, but their custom logic must obey the Design Rule Checks (DRCs) imposed by the cloud providers. For example, no combinatorial loops are allowed on AWS FPGAs, and users do not have direct

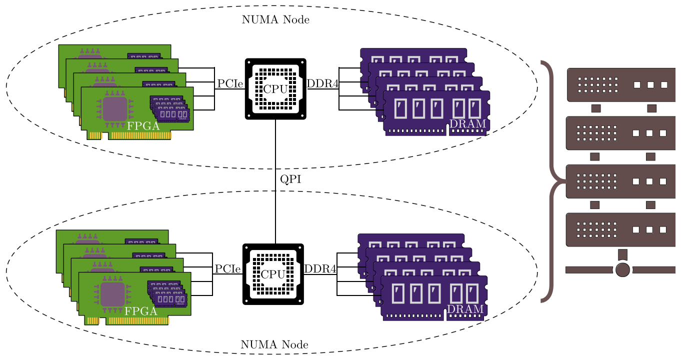


Fig. 1. Diagram of the deduced AWS server configuration, with 8 FPGAs sharing the same server across two NUMA nodes.

access to I/O pins, but are instead forced to interact with external resources through the cloud-provided shell. Users further do not have access to resources which are inaccessible in cloud environments, including identifiers such as eFUSE and Device DNA primitives, or voltage and temperature monitors. Adversaries can instead try to infer or influence such information indirectly (e.g., by using PUFs or other alternative RO constructions), but they do not have physical access to the underlying FPGA boards or server racks themselves. We also do not assume vulnerabilities in the virtualization mechanisms that would allow users to (directly) snoop on the memory or PCIe transactions of other VM instances. Attacks on the cloud-provided logic, FPGA software tools, or the bitstream itself are similarly out-of-scope.

We instead focus on detecting contention that is the result of changes in the PCIe bandwidth when different FPGAs in the same server attempt simultaneous memory accesses. A key contribution of this work is therefore the ability to detect such contention and correlate it to co-located FPGAs, without external equipment, access to privileged FPGA logic, or vulnerabilities in the FPGA software itself.

IV. Experimental Setup

The experiments in this paper are performed on FPGA F1 instances publicly available through the AWS Elastic Cloud Compute (EC2) platform. Figure 1 shows the likely AWS server configuration based on public information and the results of our experiments. In particular, a server can contain up to 8 FPGAs, split over two NUMA nodes across two CPU sockets. Each server can accommodate eight separate `f1.2xlarge` users (recall that each `f1.2xlarge` instance only uses 1 FPGA), four `f1.4xlarge` instances with 2 FPGAs each, a combination of `f1.2xlarge` and `f1.4xlarge` instances using at most 8 FPGAs in total, or a single `f1.16xlarge` customer using all 8 FPGAs. Based on our assumption that only 4 FPGAs share the same NUMA node, we expect that only interference within a NUMA node will be measurable. This translates into up to four `f1.2xlarge`, two `f1.4xlarge`, or one `f1.4xlarge` and two `f1.2xlarge` instances interfering with each other.

To confirm the above server setup configuration, i.e., to show that exactly four FPGAs can lead to mutual PCIe contention, we experiment with all three types of F1 instances. Unless otherwise specified, experiments are primarily conducted in the `us-east-1` region, but Section V-D reproduces the experimental results in all four AWS data center regions that offer FPGA instances, namely `ap-southeast-2` (Sydney), `eu-west-1` (Ireland), `us-east-1` (North Virginia), and `us-west-2` (Oregon). As explained in Section V-D, our approach works with both *on-demand* and *spot* instances.

Our setup involves testing with two FPGAs at a time, a *memory stressor* and a *memory tester*. The tester repeatedly measures its PCIe bandwidth by writing from the host VM to the FPGA DRAM, while the stressor similarly attempts to interfere with the tester's bandwidth by stressing its own PCIe connection. More precisely, we use the (unmodified) `CL_DRAM_DMA` example FPGA image provided by the AWS FPGA development kit [5] as the basis for both the PCIe tester and the PCIe stressor designs.

For the stressor, approximately $8.4\,\mathrm{MB}$ of data are moved between the CPU and FPGA for each transfer, and the transfers are repeated 100 times, giving a total transfer size of $840\,\mathrm{MB}$. This transfer takes less than a second to complete. The tester runs the same bandwidth-measurement program, but transfers $3.9\,\mathrm{kB}$ per test, or $394\,\mathrm{kB}$ total.

The stressor and tester are run in parallel on separate instances in order to observe if the tester bandwidth is affected (reduced) due to the stressor's activity or not. If it is, we conclude that the stressor and tester instances are on the same server and share the same NUMA node. The tests are repeated multiple times with the same instances to prevent false positives or negatives, e.g., due to memory activity from other, unrelated users' instances being on the same server.

Finally, in addition to measuring the bandwidth, we use the open-source code by Tian et al. [41] to collect Physical Unclonable Function (PUF) fingerprints of the FPGA instances and more concretely map the cloud infrastructure.

V. Evaluation

In this section we perform a thorough evaluation of the cross-FPGA PCIe contention in several geographical regions and VM instance types. Specifically, we first analyze the impact of simultaneous memory accesses over PCIe in `f1.16xlarge` instances, showing that FPGAs in slots 0–3 and 4–7 interfere with each other, and conclude that they form separate NUMA nodes (Section V-A). We then experiment with dozens of `f1.2xlarge` (Section V-B) and `f1.4xlarge` (Section V-C) instances across different data center regions (Section V-D) to determine that successive instances often (but not always) belong to the same NUMA locality, for both spot and on-demand instance types. We use our results to calculate the probability of renting FPGAs in the same server (Section V-E) and fingerprint the FPGAs (Section V-F), showing overlap among instance types, in contrast to prior work. We finally discuss some practical aspects of our reverse-engineering approach (Section V-G).

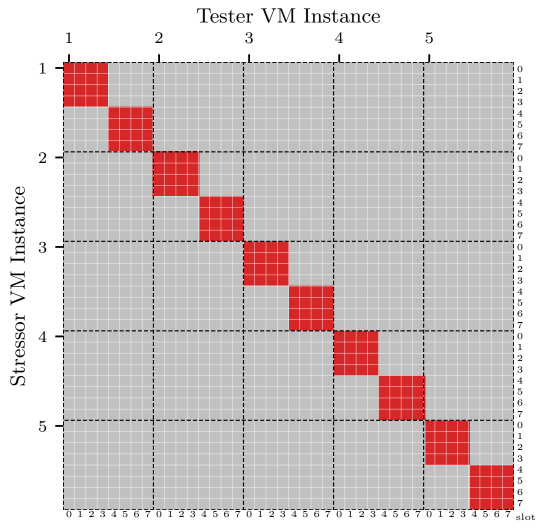
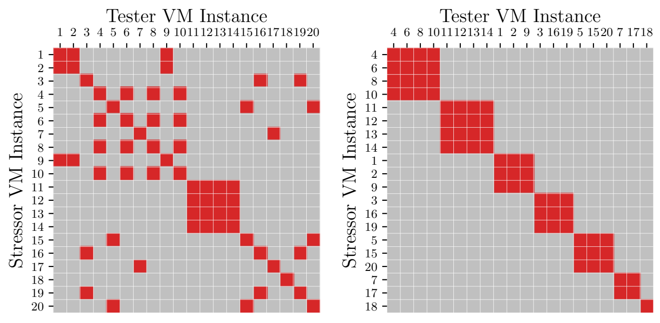


Fig. 2. PCIe contention analysis for all eight FPGA slots in five `f1.16xlarge` instances rented concurrently in the `us-east-1` region. A red square denotes that the stressor (y-axis) interferes with the tester (x-axis), as determined by observing that the PCIe bandwidth is reduced below a target threshold. As can be seen, contention exists among groups of exactly 4 consecutive slots within a single VM instance.

A. Determining NUMA localities

Based on the background presented in Section II-A, we expect each F1 server to contain 2 CPUs, each of which has 4 FPGAs in its locality, for a maximum of 8 FPGAs per server. We verify this in several ways. First, by running the `numactl --hardware` command [20], we determine that `f1.16xlarge` instances produce `2 nodes` compared to smaller instances, which show `1 nodes`. The results of `lscpu` are similar, and commands like `hwloc-info` or `lstopo -p` also only reference `2 NUMANodes` in the `f1.16xlarge` case. Even though information about the physical hardware in VMs is not always reliable, (e.g, `/sys/bus/pci/devices/<id>/numa_node` returns `-1` for the various PCIe slot identifiers), an application note by AWS also confirms that FPGAs in slots 0–3 and 4–7 are separate "groups" (i.e, NUMA nodes), with FPGAs within a group being able to "directly access other FPGAs within the same group", while an access "between groups is not direct and not optimal (higher latency, lower bandwidth)" [2].

With the above information supporting our initial assumptions, we rent five `f1.16xlarge` spot instances, containing a total of 40 FPGAs. We use each of those FPGAs individually as a stressor, and consecutively (one by one) test the effect on all FPGAs (including the stressor) as testers, for a total of $40 \times 40 = 1{,}600$ data points. We repeat measurements twice, with the results of both repetitions being identical: as shown in Figure 2, we only find contention between $(instance, slot)$ FPGA pairs (i_1, s_1) and (i_2, s_2) if and only if $i_1 = i_2$ and $0 \leq s_1, s_2 \leq 3$ or $4 \leq s_1, s_2 \leq 7$. In other words, there is no overlap between separate `f1.16xlarge` instances or cross-



(a) Original (b) Re-ordered

Fig. 3. Cross-VM contention between `f1.2xlarge` instances in the `us-east-1` region: (a) presents the instances in the order in which they are launched, while (b) re-orders them to more clearly show pairs with PCIe contention.

NUMA effects, and the graph is symmetric, i.e., the roles of stressors and testers are interchangeable.

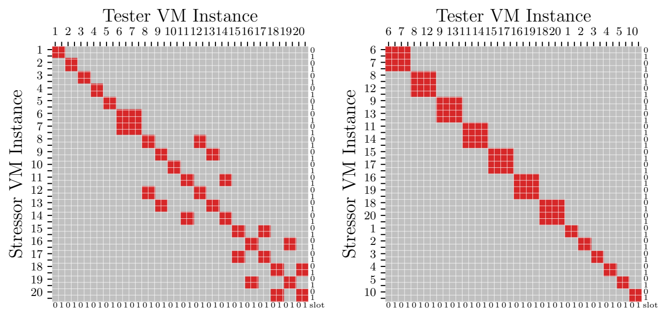
B. Cross-VM PCIe Contention

In this section we rent 20 `f1.2xlarge` spot instances in the `us-east-1` region, collecting $20 \times 20 = 400$ bandwidth measurements for each pair of possible stressor and tester combinations. We repeat experiments five times, with identical results for each repetition, again proving that our bandwidth-based metric is a robust way of detecting cross-instance contention. The results are plotted in Figure 3 in two ways. First, Figure 3a shows the results of our measurements, with instances numbered in the chronological order in which they are launched. Second, Figure 3b re-orders the instances so that FPGA pairs with contention are plotted adjacent to each other.

As in the `f1.16xlarge` case, the resulting matrix is symmetric, since the contention is bi-directional. More importantly, otherwise-independent VM instances affect each other in a measurable way. Specifically, within the 20 instances launched, we find two groups of full NUMA nodes (i.e., 4 FPGAs), three groups of 3 FPGAs, one of 2 FPGAs, and only one FPGA without any contention, likely because it only has 2 instances launched after it. FPGAs within the same NUMA node are occasionally returned one after the other (e.g., instances 11–15), but are also sometimes interspersed with other NUMA nodes (e.g., instances $\{4, 6, 8, 10\}$). We calculate the probability of renting another instance in the same NUMA node in Section V-E.

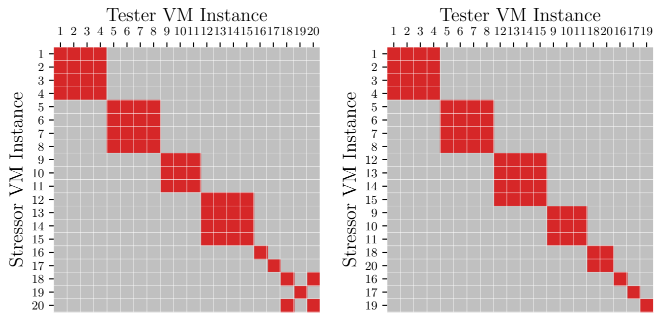
C. Contention between `f1.4xlarge` Instances

To evaluate whether our observations also hold for `f1.4xlarge` instances, we rent 20 `f1.4xlarge` spot instances in `us-east-1`, for a total of $40 \times 40 = 1{,}600$ stressor and tester pairs, repeating measurements three times. The results, which are shown in Figure 4, indicate that contention is still possible both within and between different `f1.4xlarge` instances: we find 7 pairs of distinct instances that form complete NUMA nodes with 4 FPGAs. We again notice that co-located instances tend to not be fully consecutive, but are



(a) Original (b) Re-ordered

Fig. 4. Cross-VM contention between `f1.4xlarge` instances with 2 FPGAs each in the `us-east-1` region. In this test, 14 of the 20 launched instances are co-located in 7 NUMA nodes while the remaining 6 are not co-located with any of the other instances. Recall that `f1.4xlarge` VMs contain two FPGAs, so the two slots within an instance always interfere with each other, explaining why there are always at least two red squares per row and column in the figure.



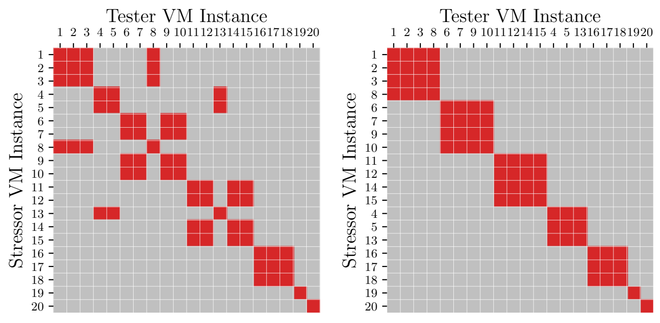
(a) Original (b) Re-ordered

Fig. 5. Example test results for cross-VM contention between `f1.2xlarge` spot instances in the `us-west-2` region.

instead interspersed with other instances. However, unlike the results of Section V-B, where the lone FPGA was rented near the end of the 20 instances, all six lone `f1.4xlarge` instances are among the first 10 instances launched, with the first five corresponding to instances 1–5. In other words, in our experiments, it was almost always possible to find contention in `f1.2xlarge` instances provided enough instances were launched after them. However, the first few `f1.4xlarge` instances were not co-located with any other instances, while later VMs were more likely to be co-located in the same server.

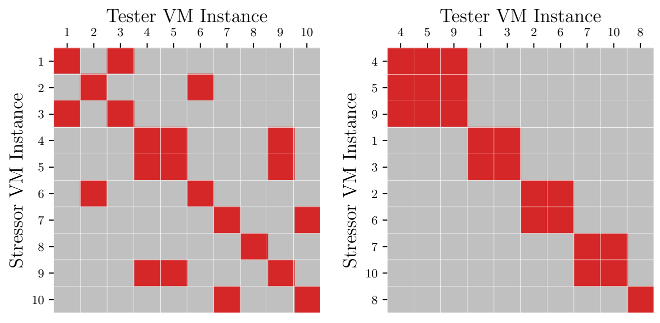
D. Data Center Regions and On-Demand Instances

As the previous experiments were conducted with spot instances in the `us-east-1` region, we perform measurements with spot instances in the `us-west-2` (Figure 5) and `eu-west-1` (Figure 6) regions, as well as with on-demand instances in the `ap-southeast-2` (Figure 7) and `us-east-1` (Figure 8) regions, with experiments repeated once with 20 and 10 FPGAs respectively. It should be noted that we could not find availability for spot instances in the `ap-southeast-2` region. In addition, the pre-synthesized



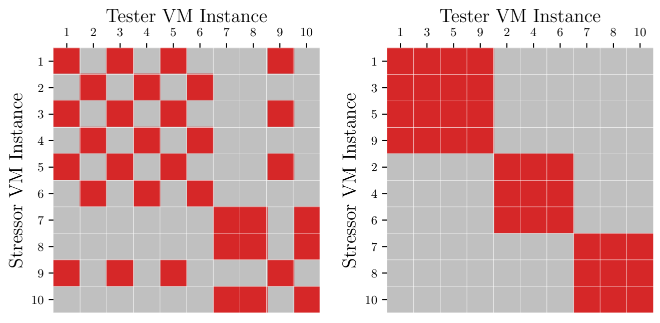
(a) Original (b) Re-ordered

Fig. 6. Example test results for cross-VM contention between `f1.2xlarge` spot instances in the `eu-west-1` region.



(a) Original (b) Re-ordered

Fig. 8. Cross-VM contention between `f1.2xlarge` on-demand instances in the `us-east-1` region.



(a) Original (b) Re-ordered

Fig. 7. Cross-VM contention between `f1.2xlarge` on-demand instances in the `ap-southeast-2` region.

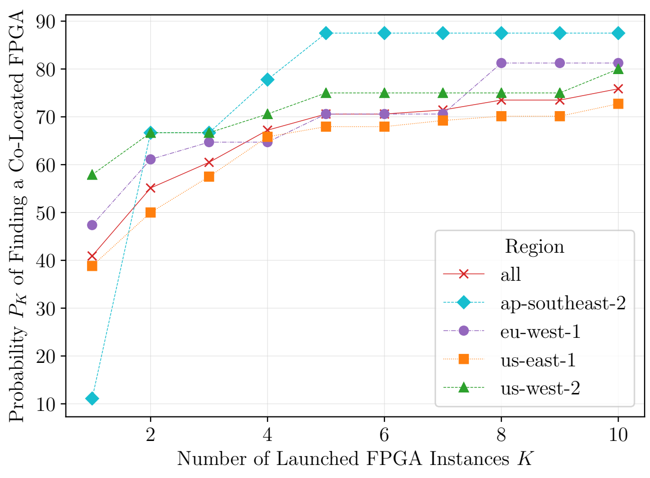


Fig. 9. Probability P_K of finding another `f1.2xlarge` instance in the same NUMA node (overall and for individual AWS data centers), as a function of the number of FPGA instances launched K.

`CL_DRAM_DMA` AFI was not available in this region, so we synthesized it using its publicly-available source code.

The results are broadly similar to the experiments of Section V-B, with only 10% (6/60) of instances not resulting in cross-VM contention. Indeed, 7 complete NUMA localities are identified, along with 6 groups of three FPGAs and 4 pairs of two VMs. It is further interesting to note that, in our experiments, groups in `us-west-2` almost always appeared in succession, while groups in other regions were more disjointed. More experiments could determine whether this pattern was due to user demand and usage characteristics (which may differ among regions), or whether different instance allocation strategies apply to the various data centers, instance types, or times of the day or week.

E. Probability of Co-Location

Having uncovered that up to four `f1.2xlarge` instances can interfere with each other, we further analyze the probability of the `f1.2xlarge` instances being co-located on the same server. Specifically, we use the data gathered in previous sections to calculate the probability P_K that, given a tester FPGA, launching K stressor instances will result in at least one of the K new FPGAs being placed in the same NUMA node as the tester FPGA.

Let N_K be the number of VMs for which there is PCIe contention with any of the *next* K instances launched. Moreover, let M_K be the number of VMs which have fewer than K instances launched after them *and* which are not the last in a full NUMA node detected within the experiment. The second constraint is needed because, although for groups of up to 3 FPGAs the remaining FPGAs might be found by renting more instances, if the NUMA node has been fully detected, no additional VM will correspond to the same locality, no matter how many instances are launched. Denoting by T as the total number of VMs launched, the desired probability is

$$P_K = \frac{N_K}{T - M_K} \tag{1}$$

As an example using the on-demand instances of Figure 8, $T = 10$, and for $K = 3$, $M_K = 3$ (instances 8–10 have ≤ 2 VMs launched after them), while $N_K = 3$ (instances 1, 4, and 7 are in the same NUMA nodes as instances 3, 5, and 10 respectively), so $P_3 = 3/(10 - 3) = 43\%$.

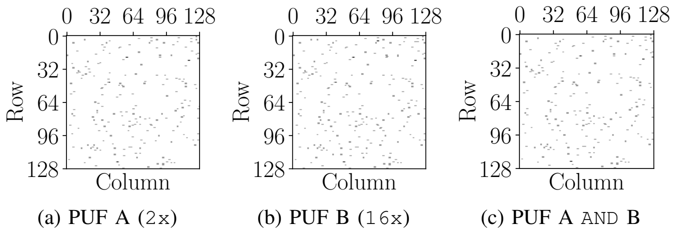


Fig. 10. Sample PUF fingerprints from a pair of overlapping FPGAs between (a) `f1.2xlarge` and (b) `f1.16xlarge` instances. The two PUFs and their (c) bitwise `AND` are almost identical.

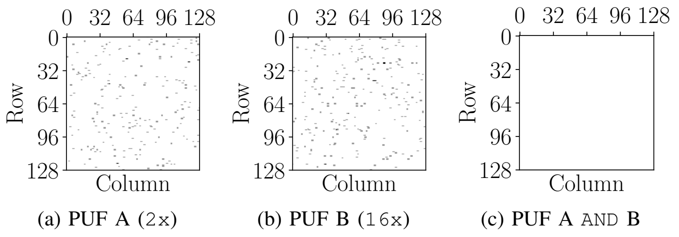


Fig. 11. Sample PUF fingerprints from non-overlapping FPGAs between (a) `f1.2xlarge` and (b) `f1.16xlarge` instances. The two PUFs are distinct, and their (c) bitwise `AND` is empty.

Figure 9 calculates P_K for $1 \leq K \leq 10$, both for individual regions and over all regions tested. Depending on the region, the probability that two consecutive VMs are co-located (i.e., $K = 1$) ranges between 38–58% (with the exception of `ap-southeast-2`), while renting just one more instance can increase this probability by approximately 10 percentage points (or 55 points for the Sydney region). Renting even more instances increases this probability further to about 80% for $K = 10$, in part due to the smaller number of instances that have at least K FPGAs launched after them (i.e., a larger M_K).

Note that for sufficiently large T and K, we expect $P_K = 75\%$, as there is a 1 in 4 chance that the sensor instance is the last FPGA in its NUMA node. However, for smaller T and K, P_K can be larger (as in Figure 9), since Amazon does not always fully pack consecutively instances within a single server: as the previous sections showed, co-located instances are often launched further apart in time.

F. Overlap between Instance Types

In this section, we use the open-source code by Tian et al. [41] to fingerprint individual FPGAs and detect overlaps between experiments repeated on different days. Specifically, we make additional measurements from the `us-east-1` region, and compare the PUF fingerprints between spot and on-demand `f1.2xlarge` instances in availability zone `c` and spot `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge` instances in zone `e`, all collected on different days.

We reach two main conclusions. First, there is an overlap between spot and on-demand VMs, and, second, unlike what prior work suggested [41], there *is* overlap between all three instance types. For example, instances 7 and 17 of Figure 3 overlap with slots 0 and 1 of instance 3 in Figure 2. Figure 10 presents extracts from the PUF fingerprints for one of the two pairs of overlapping FPGAs. The FPGAs in the `f1.2xlarge` and `f1.16xlarge` instances had 419 and 461 DRAM bit flips respectively, of which 419 bit flip locations (DRAM addresses) were identical. This allows us to conclude that the two instances correspond to the same underlying FPGA hardware. By contrast, Figure 11 shows the same `f1.2xlarge` instance along with a different FPGA slot of the `f1.16xlarge` instance. This FPGA has a fingerprint with 483 bit flips, of which none are in the same locations as those in the `f1.2xlarge` instance. As a result, these two FPGAs are distinct, as expected.

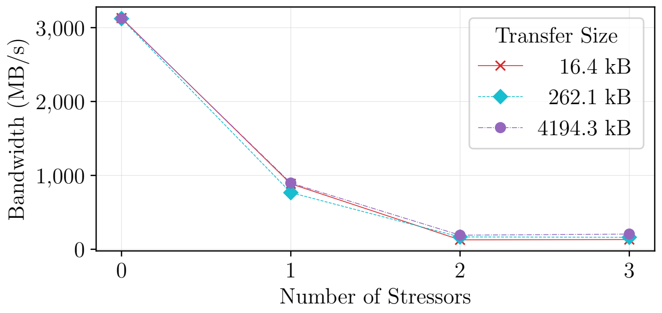


Fig. 12. Median tester bandwidth for different numbers of enabled stressors and transfer sizes (bandwidth averaged over 100 transfers).

In addition, we found an overlap between instances 1, 2, and 9 of Figure 3 and another set of 10 `f1.4xlarge` instances rented. Consequently, not only is there the potential for cross-VM contention between identical spot instance types, but also between `f1.2xlarge` and `f1.4xlarge` spot and on-demand instances (but not `f1.16xlarge` ones, as they reserve all FPGAs within the server).

It is worth noting that the overlap between different instance types is perhaps somewhat surprising. Tian et al. rented many VMs (60 for each instance type), but their goal was to find repeated hardware allocations over time, so these VMs were not rented simultaneously. AWS thus often re-used the same instances, resulting in only 10 unique `f1.2xlarge`, 6 unique `f1.4xlarge`, and 8 unique `f1.16xlarge` instances, for a total of 86 FPGAs [41]. Instead, we found an overlap of just two FPGAs in a pool of 20 `f1.2xlarge` and 5 `f1.16xlarge` instances, three FPGAs between the same set of 20 `f1.2xlarge` and an additional 10 `f1.4xlarge` instances, and no overlap between the ten `f1.4xlarge` and five `f1.16xlarge` instances, for a total of $20 \cdot 1 + 10 \cdot 2 + 5 \cdot 8 - 2 - 3 = 75$ unique FPGAs. This suggests that it is rare (but not impossible) for instance types to be repurposed, likely primarily in cases of unmet demand of smaller F1 types.

G. Practical Considerations

The presented infrastructure mapping attack is inexpensive to deploy, and anyone with access to the public cloud FPGA infrastructure can perform it, using just the utilities provided by AWS. In fact, it remains both easy and cheap to find instances in the same NUMA node: less than a minute is needed to determine whether the given stressor is co-located with any of the testers.

Another important aspect to consider is whether detecting contention is possible in the presence of interference from other tenants. To address this issue, we rent an `f1.16xlarge` instance and investigate simultaneous transmissions from 0–3 stressor FPGAs to the last one in the NUMA node. We experiment with different transfer sizes, i.e., bytes moved from the DRAM to the stressor, and average the calculated bandwidth over 100 transfers. The results, summarized in Figure 12, show that the tester bandwidth is highest when no stressors are enabled, at over 3 GBps. When a stressor is enabled (at any transfer size), the bandwidth quickly drops below 1 GBps, and is the reason why we can effectively detect co-located FPGAs with a simple threshold. Enabling a second stressor FPGA, bandwidth becomes even lower at 128–190 MBps (depending on the transfer size), and remains approximately the same when the final FPGA in the NUMA node acts as a third stressor.

As a result, it is still possible to reverse-engineer the infrastructure and detect co-location in the presence of traffic generated by a single external user, provided that the bandwidth sustained by that user remains the same during the measurement period. In other words, the tester threshold needs to be adjusted to account for the additional external traffic, e.g., from 2.0 GBps to 0.5 GBps. Moreover, the adversary can completely bypass any effects from third parties by renting `f1.4xlarge` instances, thereby using all resources in a NUMA node. In fact, doing so allows them to more quickly map the infrastructure, but at a higher cost.

VI. Related Work

In this section we summarize prior work in FPGA security (Section VI-A) and cloud-related attacks (Section VI-B).

A. Remote FPGA Attacks

In recent years, besides attacks on the FPGA bitstream itself, e.g., [10], there has been extensive research on FPGA security without physical access to the underlying hardware, with covert-channel, side-channel, and fault attacks predominantly using voltage or temperature to affect the FPGA chips [17]. Many such works, e.g., [12], [18], [29], [44], focus on attacks between different users of the FPGA, and are therefore not directly applicable to single-tenant clouds.

Although most attacks have been performed in lab environments, a covert-channel attack between separate dies ("Super Logic Regions") was shown to be possible on AWS and Huawei cloud [14], and a side-channel attack on AWS by Glamocanin et al. soon followed [15]. The former depends on alternative ring oscillator designs that bypass AWS restrictions [13], [37], while the latter uses Time-to-Digital Converters (TDCs), both of which could be detected by additional Design Rule Checks (DRCs) [19]. By contrast, our research does not focus on the FPGA chip, but the cloud FPGA infrastructure, and in particular the shared PCIe bus used by different FPGA boards within each server.

B. Cloud Security

Since the initial public deployments of cloud computing infrastructures, researchers have looked at ways to attack (and improve) their security. In early Amazon EC2 cloud architectures involving only CPUs, researchers quickly showed how to reverse-engineer the infrastructure and place an attacker VM on the same server as the victim VM [33]. The main method for doing so was through internal IP addresses and network latencies, which exposed sufficient information about the underlying setup for effectively 0% false positives, and a 40% chance of VM co-location on the same server and CPU in the AWS EC2 cloud [33]. The false positive rate and the co-location probability are all comparable to (but lower than) those in our work focusing on PCIe and FPGAs.

There is also a body of research that has demonstrated that contention of I/O resources (hard-drive throughput and network bandwidth, for example) is also a potential security vulnerability in cloud infrastructures such as EC2. For example, it is possible to cause performance degradation of co-located instances [8], [30]. Also, Richter et al. showed that when virtualizing PCIe Network Interface Cards (NICs) using Single Root I/O Virtualization (SR-IOV), it is feasible for one VM to cause congestion on the NIC ingress buffers [31]. As a possible defense, Richter et al. recommend Quality-of-Service (QoS) extensions and different scheduling algorithms to ensure that flooding a Virtual Function (VF) in one Physical Function (PF) cannot cause performance degradation in a different PF [32].

Our work further advances research in similar areas, as we have shown how to determine co-location and aspects of the scheduling algorithm using PCIe contention in FPGA-accelerated clouds. Based on our work, attacks on intentional performance degradation of the PCIe bandwidth or further work on understanding how interference can affect our new attack or disrupt other users are a natural extension as well.

VII. Conclusion

This paper identified PCIe contention as a means for mapping cloud FPGA infrastructures. We showed that it is possible to reverse-engineer the NUMA locality of different FPGAs within an AWS server, and therefore find which `f1.2xlarge` and `f1.4xlarge` instances are co-located within the same server. We also found that `f1.2xlarge` and `f1.4xlarge` instance types can be scheduled on the same AWS server, and we uncovered that the probability of successive users renting FPGAs within the same server is high. Consequently, this paper highlighted the dangers of shared infrastructures and the ability for adversaries to map and analyze whether FPGA instances are co-located by use of PCIe contention. In order to secure FPGA-accelerated infrastructures it is therefore not only necessary to consider the FPGA chip security, but also other system components accessible to the logic running on the FPGA.

References

[1] Alibaba Cloud, "Instance families," https://www.alibabacloud.com/help/doc-detail/25378.htm, 2020, Accessed: 2020-12-01.

[2] Amazon Web Services, "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, 2019, Accessed: 2020-12-01.

[3] ——, "The agility of F1: Accelerate your applications with custom compute power," https://d1.awsstatic.com/Amazon_EC2_F1_Infographic.pdf, 2020, Accessed: 2020-12-01.

[4] ——, "Amazon EC2 instance types," https://aws.amazon.com/ec2/instance-types/, 2020, Accessed: 2020-12-01.

[5] ——, "CL_DRAM_DMA custom logic example," https://github.com/aws/aws-fpga/tree/master/hdk/cl/examples/cl_dram_dma, 2020, Accessed: 2020-12-01.

[6] Baidu Cloud, "FPGA cloud compute," https://cloud.baidu.com/product/fpga.html, 2020, Accessed: 2020-12-01.

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

[8] R. C. Chiang, S. Rajasekaran, N. Zhang, and H. H. Huang, "Swiper: Exploiting virtual machine vulnerability in third-party clouds with competition for I/O resources," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, no. 6, pp. 1732–1742, Jun. 2015.

[9] 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.

[10] M. Ender, A. Moradi, and C. Paar, "The unpatchable silicon: A full break of the bitstream encryption of Xilinx 7-Series FPGAs," in *USENIX Security Symposium*, 2020.

[11] 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.

[12] I. Giechaskiel, K. B. Rasmussen, and K. Eguro, "Leaky wires: Information leakage and covert communication between FPGA long wires," in *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2018.

[13] 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.

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

[15] 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.

[16] Huawei Cloud, "FPGA accelerated cloud server," https://www.huaweicloud.com/en-us/product/fcs.html, 2020, Accessed: 2020-12-01.

[17] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud FPGAs: A survey," https://arxiv.org/abs/2005.04867, 2020, Accessed: 2020-12-01.

[18] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "FPGAhammer: Remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, vol. 2018, no. 3, pp. 44–68, Sep 2018.

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

[20] C. Lameter, "NUMA (non-uniform memory access): An overview," *ACM Queue*, vol. 11, no. 7, pp. 40–51, Jul. 2013.

[21] 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.

[22] T. Lutz, C. Fensch, and M. Cole, "PARTANS: An autotuning framework for stencil computation on multi-GPU systems," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 9, no. 4, pp. 1–24, Jan. 2013.

[23] M. Martinasso, G. Kwasniewski, S. R. Alam, T. C. Schulthess, and H. Torsten, "A PCIe congestion-aware performance model for densely populated accelerator servers," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016.

[24] C. McCurdy and J. Vetter, "Memphis: Finding and fixing NUMA-related performance problems on multi-core platforms," in *IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, 2010.

[25] D. Molka, D. Hackenberg, R. Schöne, and M. S. Müller, "Memory performance and cache coherency effects on an Intel Nehalem multiprocessor system," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2009.

[26] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore, "Understanding PCIe performance for end host networking," in *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.

[27] Nimbix, Inc., "Xilinx Alveo accelerator cards," https://www.nimbix.net/alveo, 2020, Accessed: 2020-12-01.

[28] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user FPGA environments," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2019.

[29] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb, "Characterization of long wire data leakage in deep submicron FPGAs," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2019.

[30] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of I/O workload in virtualized cloud environments," in *IEEE International Conference on Cloud Computing (CLOUD)*, 2010.

[31] A. Richter, C. Herber, T. Wild, and A. Herkersdorf, "Denial-of-service attacks on PCI passthrough devices: Demonstrating the impact on network- and storage-I/O performance," *Journal of Systems Architecture*, vol. 61, no. 10, pp. 592–599, Nov. 2015.

[32] ——, "Resolving performance interference in SR-IOV setups with PCIe Quality-of-Service extensions," in *Euromicro Conference on Digital System Design (DSD)*, 2016.

[33] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *ACM Conference on Computer and Communications Security (CCS)*, 2009.

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

[35] R. Solomon, "PCI express basics & background," https://pcisig.com/sites/default/files/files/PCI_Express_Basics_Background.pdf, 2014, Accessed: 2020-12-01.

[36] 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.

[37] 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.

[38] Tencent Cloud, "FPGA cloud server," https://cloud.tencent.com/product/fpga, 2020, Accessed: 2020-12-01.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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, 2020, Accessed: 2020-12-01.

[44] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy (S&P)*, 2018.