EDITORS: Anne Elster, anne.elster@gmail.com
Sonia Lopez Alarcón, slaeec@rit.edu

Realizing Network-Attached FPGAs in the Cloud

Miriam Leeser, Suranga Handagala, Dana Diaconu, and Zhaoyang Han, *Northeastern University, Boston, MA, 02115, USA*

Michael Zink, *University of Massachusetts, Amherst, MA, 01003, USA*

The Open Cloud Testbed (OCT) has the goal to provide bare-metal resources to the systems research community. In contrast to other infrastructures that offer such resources, the field-programmable gate arrays (FPGAs) in OCT expose their network interface directly to the experimenter, unlocking new possibilities for researchers focusing on cloud application acceleration and data center disaggregation. OCT provides two models for programming network-attached FPGAs. The first grants users direct access to the network interface. The features of this approach are demonstrated with examples that make use of User Datagram Protocol (UDP) and Transmission Control Protocol stacks. An example involving UDP is highlighted that facilitates a distributed machine learning (ML) application. The second model allows FPGA network interface programming via Programming Protocol-Independent Packet Processors, thus providing the FPGA with smart network interface card capabilities. In addition, this article outlines future research directions into network-attached FPGAs, including the security of such systems.

Field-programmable gate arrays (FPGAs) are increasingly available in the cloud. A previous paper introduced FPGAs in the Open Cloud Testbed (OCT).[1] One of the features of OCT is that the FPGAs are connected directly to the network. Most importantly, this connection is exposed to the user.

In a standard configuration, accelerators such as FPGAs and graphics processing units (GPUs) are attached to a host computer via a Peripheral Component Interconnect Express (PCIe) bus, and the host computer is attached to the network. Data are downloaded from the network to the host, and then downloaded to the accelerator. After processing, the data are returned to the host and then potentially sent out on the network to the consumer. All of this data movement is expensive in terms of both time and energy and does not involve any processing. Even with significant acceleration, the additional latency from data movement may prevent speedup for a user's application. In contrast, by connecting accelerators like FPGAs directly to the network, data can be received and processed with very low latency. For designs that target big data, multiple FPGAs can be used for processing, and data can be shared directly without involving the host. This is not only more efficient, but also opens up new possibilities for acceleration in cloud applications. A variety of different configurations for FPGAs in the cloud are described in Bobda et al.[2]

OCT supports FPGAs directly connected to a host via PCIe using the standard coprocessor model. In addition, the FPGAs have two 100-Gbps Ethernet ports that are directly connected to a top-of-rack (TOR) switch. Users of these network-attached FPGAs are supported with two models. The first model exposes the network connection directly to the user. Examples are provided in the OCT documentation[a] that makes

Digital Object Identifier 10.1109/MCSE.2024.3482068
Date of current version 9 December 2024.

[a]https://github.com/OCT-FPGA/OCT-Tutorials

use of User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) stacks implemented on the FPGA. In the second model, Programming Protocol-Independent Packet Processors (P4) is enabled on the FPGAs for users. P4[b] is a domain specific language for programming the data plane of networking devices. It was originally developed by the P4 foundation for data forwarding on network switches. Recently, P4 has been extended to devices like SmartNICs and FPGAs. A set of tutorials[c] are available that teach experimenters how to develop and deploy P4 programs on the FPGAs in OCT.

Network-attached FPGAs open up the possibility of new configurations in the data center, including disaggregation of accelerators. These new configurations also open up security challenges as discussed further in this article.

FPGAs IN THE OPEN CLOUD TESTBED

There are several places where users can get access to FPGAs in the cloud including Amazon Web Services. FPGAs are also used by Microsoft to accelerate Bing search and Azure workloads, although the FPGAs themselves are not directly programmable by the user. In both these systems, the user has no direct access to network connections on the FPGAs. In contrast, OCT provides support for users to exploit direct network connections on FPGAs for their research and provides different programming models to support this.

OCT Hardware

OCT currently offers 32 FPGAs to the research community: 24 AMD Alveo U280, four AMD VCK5000, and four AMD V70 as illustrated in Figure 1. Each of these FPGAs is housed in a host server that an experimenter can allocate as a bare metal machine. Twenty-eight (all U280s and VCK5000s) of the 32 FPGAs have two direct network links, which connect both of their QSFP28 (100GbE) interfaces to a switch for a combined maximum bandwidth of 200 Gbps.

Both VCK 5000 and V70 incorporate artificial intelligence (AI) engines, which are specialized hardware blocks designed to accelerate AI workloads efficiently. They support a range of AI frameworks and libraries, such as TensorFlow and PyTorch, through Vitis AI, AMD's AI development environment.

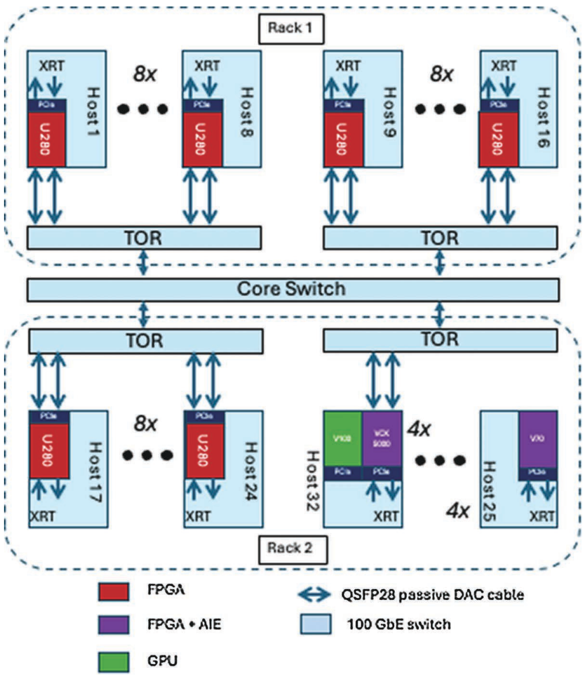

FIGURE 1. OCT hardware setup.

OCT Workflow

OCT offers a toolchain that supports the development of bitstreams that can be deployed on the FPGAs.[1] The OCT workflow involves two main stages: development and deployment.

Development Stage

OCT development tools reside on virtual machines (VMs) within the New England Research Cloud.[d] Users can remotely log into these VMs to build FPGA bitstreams and host executables using the installed tools. Alternatively, users can opt to use their own build machine, where they need to install AMD Vitis and runtime tools.

Deployment Stage

After creating the bitstreams and host executables, users transfer these from the development machine to the deployment machine(s) in OCT, where the FPGAs are installed. The process involves programming the FPGAs with the bitstreams, executing the host executables, and fetching the results back to the development machine.

OCT supports multiple different programming models for the user, complete with instructions and

[b]https://opennetworking.org/p4/
[c]https://github.com/OCT-FPGA/P4Framework.git

[d]https://nerc.mghpcc.org/

tutorials. The most popular programming approach is to make use of the AMD Vitis application development tools where researchers describe their applications in C++ and, using High Level Synthesis (HLS), develop bitstreams to program the FPGAs. With this flow, the static region of the FPGA is programmed with an AMD provided shell, enabling users to load their bitstreams into the dynamic region of the design using partial reconfiguration. For more experienced FPGA users, a custom flow based on Vivado and hardware description languages is also available. In the custom flow, users can program a custom shell using the conventional Vivado-based approach. With both of these flows, the user has access to the network connections available on FPGAs. Examples are provided in the OCT documentation.[e]

IN THE CUSTOM FLOW, USERS CAN PROGRAM A CUSTOM SHELL USING THE CONVENTIONAL VIVADO-BASED APPROACH.

P4-based FPGA support has been made available on OCT, including tutorials and several demonstration examples.[3] The P4 design further supports the potential of FPGAs as programmable network devices. A set of tutorials[f] are available. Recently an application has been implemented that makes use of a combination of P4 and HLS flows. While P4 is a limited language that only supports stateless processing, it supports the use of extern functions as a supplement. On the FPGAs, these extern functions can be designed with HLS and used in conjunction with P4 packet processing.

APPLICATIONS

Network-attached FPGAs support several different applications and improve processing and latency in the cloud.

P4 Programmable FPGA Platform

P4,[g] a rapidly developing language for network data plane programming, has had a large impact on recent networking research. It allows programmers, as opposed to network device vendors, to define and program devices directly connected to the network.

[e]https://github.com/OCT-FPGA/OCT-Tutorials
[f]https://github.com/OCT-FPGA/P4Framework.git
[g]https://p4.org/

Previous work presents a framework that allows OCT users to develop and deploy their P4 applications on cloud FPGAs.[3] Examples in the original P4 tutorials[h] have been translated to OCT FPGAs. By using the FPGAs and the P4 framework, we transform the cloud FPGAs into P4-programmable network interface cards (NICs) for network researchers to test their in-network processing applications at 100-Gbps line rate. In addition, with the support of partial reconfiguration, FPGAs in OCT provide research opportunities in "runtime" programmable networking[4] and demonstrate examples for changing P4 applications without shutting down the device.

Machine Learning on the Network

Network-attached FPGAs provide an efficient solution for high-performance machine learning (ML) accelerators in the cloud. In-network processing bypasses the need for host involvement which significantly reduces the latency. Complex ML architectures need many more FPGA resources, and it can often be impossible to fit a dataflow accelerator of the neural network on a single FPGA. As an example, experiments with a quantized ResNet-50 resulted in a split accelerator on three FPGAs. Details about dataflow accelerators and their implementation can be found in Diaconu et al.[5] Note that these dataflow accelerators deployed on multiple network-connected FPGAs can achieve state-of-the-art throughput and latency in part due to direct FPGA-to-FPGA communication.

Figure 2 shows the advantage of network data transfers for one image inference. The host is only involved to provide the initial data; in this example, this includes the input image to be classified and a file that contains external weights. The intermediate results can be transferred directly to the next FPGA through their network ports and through the UDP protocol, avoiding the slow back-and-forth communication with hosts.

Another opportunity of AI acceleration in OCT is by using the available Versal devices. The tools, available through Vitis AI, support developers who are deploying and accelerating their AI models, and they also

[h]https://github.com/p4lang/tutorials.git

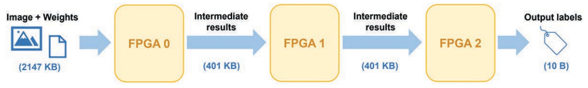

FIGURE 2. Data transfer example for one image inference of the multi-FPGA ResNet-50 accelerator.

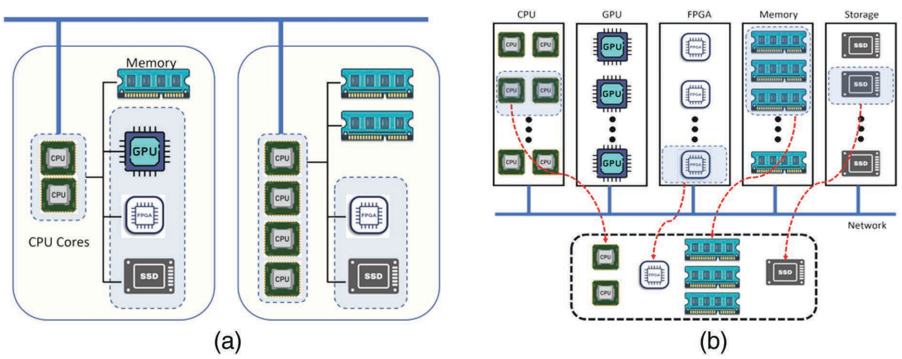

FIGURE 3. (a) Node centric model. (b) Disaggregated data center. Dashed box represents the resources requested by a user.

make the devices more accessible to users with less hardware experience. Implementing dataflow accelerators requires a certain level of hardware expertise because they are fully customized to the neural network, while the Versal devices provide a more general and straightforward accelerating platform. The AMD VCK5000, similar to the AMD U280, is equipped with dual 100 Gb/s network connections and supports both Vivado and Vitis design flows, while the AMD V70 is specifically designed for use with Vitis AI. The VCK5000 has a larger form factor suitable for high-end applications involving high speed network connectivity, whereas the V70 integrates AI capabilities into a tightly integrated, energy-efficient engine tailored for AI inference in low-power applications.

THE DATA CENTER OF THE FUTURE

By directly attaching accelerators to the network, new directions for data centers and cloud computing are enabled, as discussed in this section. We first introduce the potential to disaggregate resources both physically and logically and then discuss new security challenges presented by network-attached accelerators and initial directions for addressing them.

Disaggregation and Serverless: New Data Center Architectures

In the standard cloud model, a user requests a number of nodes on which to conduct their experiments, where a node consists of one or more CPUs, memory, and possibly accelerators and persistent storage such as solid-state drives. Each node is connected to the network through a processor, and all traffic is handled by the processor before being redirected to the other nodes, as shown in Figure 3(a). This node-centric model is inefficient for several reasons. In addition to the fact that all data must go through the host computer to get to the network as described previously, resources are not allocated very efficiently. For example, a user whose application requires a lot of memory or storage may need to request several nodes to accommodate these storage needs, while a second user whose application makes use of very little storage but requires a lot of accelerators would need to request several different nodes to accommodate their workload.

A recent trend in cloud computing is to allocate resources with a disaggregated model such that, for example, the first user can get as much storage as they need, while the second can get as many accelerators as they need, and such devices are allocated independent of the nodes they may reside in.[6,7] A fully disaggregated data center is illustrated in Figure 3(b). Note that disaggregation may be conceptual, where devices still exist within nodes, but are allocated independently. However, disaggregation is very inefficient unless resources are connected directly to the network, as they are in OCT. The first physically disaggregated resources implemented were memories.[8]

While disaggregation does not require network-attached devices, such devices make it easier to implement. Currently, GPUs are not completely connected to the network, although trends are moving in that direction with technologies such as NVLink and NVSwitch[i] from Nvidia.[9]

Disaggregation goes hand in hand with the trend of serverless computing.[j] Serverless computing allows users of cloud computing resources to request jobs to be completed independently of requesting resources. Serverless is a software abstraction, while disaggregation is a hardware abstraction. As is the case with disaggregation, while no special hardware is required to

[i]https://resources.nvidia.com/en-us-tensor-core
[j]https://arxiv.org/abs/1902.03383v1

implement a serverless configuration,[k] physically disaggregated and network-attached hardware enhances the way it can be implemented.

Disaggregated and serverless computing opens up the possibility that network-connected devices, including memory, storage, and accelerators, can be allocated independently. Such devices can be used more efficiently, but these new models open up a range of challenges, which are cataloged in Shafiei et al.[10] Among these challenges are providing the programming frameworks and system support including any scheduling required. A particular challenge is introduced by network-attached devices, namely security.

Security

Cloud security is a vast area of research. Many solutions assume that all network-attached devices are processors running an operating system where various checks can be run. Accelerators such as FPGAs typically run in bare-metal mode, where there is no OS. Accelerators directly connected to the network can both launch attacks and help to protect against attacks. For example, FPGAs can flood the network by submitting packets at line rate, through either naivete on the part of the user or malicious intent such as a denial-of-service attack. While this particular focus is on network traffic, other concerns including data security need to be considered.

MANY SOLUTIONS ASSUME THAT ALL NETWORK-ATTACHED DEVICES ARE PROCESSORS RUNNING AN OPERATING SYSTEM WHERE VARIOUS CHECKS CAN BE RUN.

While network-attached devices can be the source of security problems, they can also monitor the network at line rate and react quickly. In networking, a packet flow is defined as a sequence of packets that transmit from the same source to the same destination. A heavy hitter is defined as a packet flow that occupies a large percentage of the network bandwidth; it can be a major source of network congestion. We developed a 100-G-based heavy hitter detection accelerator on the network-attached FPGA to demonstrate the benefits of FPGAs in processing high-throughput

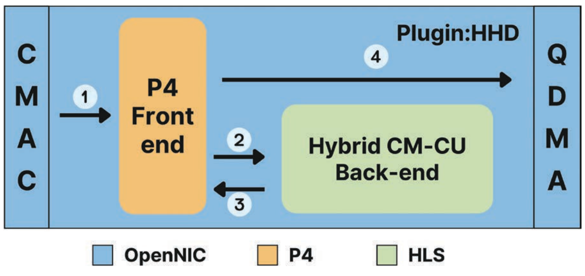

FIGURE 4. System architecture of heavy hitter detection. Data on the numbered path: 1) raw packet; 2) packet key; 3) estimate value; 4) packet with extra heavy hitter header.

network packet flows. The count-min sketch algorithm is widely used in heavy hitter detection to record the size of packet flows, as it can record key-value pairs with sublinear memory requirements. We implement count-min sketch on the FPGA as a back end for storing the key-value pairs for each packet flow; the implementation is done using high-level synthesis (HLS) where the algorithm is described in C++ and synthesized for implementation on the FPGA. We use the P4 overlay to provide the communications between FPGA and host and direct the packets to the count-min sketch back end directly from the network. The combined use of P4 and HLS for developing applications reduces the burden of designing FPGA-based accelerators. The entire design for heavy hitter detection was achieved with only hundreds of lines of code.

Figure 4 shows the system architecture of the P4-based accelerator, utilizing the open source OpenNIC Shell[l] as the core framework that provides IO support. An application, acting as a plugin block, is positioned between the 100-G media access controller (referred to as CMAC by AMD) and the queue direct memory access (QDMA) blocks. The CMAC block establishes the physical network connection and directs received packets to the application. On the output side, the QDMA block interfaces with the PCIe bus, forwarding the packets to the host. The front end of the plugin, generated from the high-level language P4, manages the packet flow within the data plane. This typical P4-based block comprises three stages: parser, control, and deparser. Utilizing the P4 parser block, packets are parsed to extract essential information, such as the IP source address and packet length, forming input tuples for the back end, the count-min sketch FPGA kernel. The control block establishes a connection with the back-end kernel, generating an additional header

[k]https://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-86.pdf

[l]https://github.com/Xilinx/open-nic-shell

field for marking heavy hitters. The deparser block reconstructs the packet header, incorporating the extra header information. For the current packet, a query is generated for the sketch algorithm to return the number of previous packets that are part of the same packet flow. If the traffic overflows a threshold, the front end will mark the corresponding packet as a heavy hitter and pass this information to the host as the additional header information.

Testing makes use of an FPGA node from OCT. It establishes a 100-G point-to-point connection with a second node in OCT. We send network traces[m] captured by the Center for Applied Internet Data Analysis to the FPGA node. These packets are real network traffic on the Equinix–Chicago backbone link from January 2016. We deploy the back-end kernel on the FPGA node to identify potential heavy hitters among these traces. With the help of P4 and HLS, 96-Gbps throughput was achieved on the network connection for the entire design.

The heavy hitter example not only helps to protect nodes in OCT from large volumes of traffic and thus helps to secure the network, but it also demonstrates a straightforward way for network researchers and cloud operators to design and deploy network applications.

CONCLUSION

In this article, we introduced network-attached FPGAs available in the Open Cloud Testbed. In comparison to commercial clouds that offer FPGAs, the ones in OCT provide the user with direct access and full control of their network interfaces. This supports a series of new research avenues for the systems community that focus on compute clouds and distributed systems.

We introduced two different models that are offered for the use of the network-attached FPGAs in OCT. The first model allows users to directly use the network interface as demonstrated by the use of a UDP stack implemented in the FPGA that supports a distributed machine learning application. The second model turns the FPGA into a Smart NIC by providing P4 capability. This feature enables the use of the FPGA by networking researchers.

Network-attached FPGAs open up many new areas of research, including realizing the disaggregated data center of the future. We are investigating in-band network telemetry, which represents an approach to network monitoring and visibility, that integrates telemetry data directly into the data plane of the network, providing real-time insights into the performance, quality, and behavior of network traffic. In addition, security issues are introduced by this new hardware model that require further investigation. We described our work to date on detecting heavy hitters using the FPGAs in OCT. We plan to continue to investigate both security threats and solutions.

We invite the research community to make use of the Open Cloud Testbed and help to design and improve the future of cloud computing.

[m]https://catalog.caida.org/dataset/passive_2016_pcap

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grants CNS-2130891, CNS-2130907, CNS-1925658, and CNS-1925464.

REFERENCES

1. M. Leeser, S. Handagala, and M. Zink, "FPGAs in the cloud," *Comput. Sci. Eng.*, vol. 23, no. 6, pp. 72–76, Nov./Dec. 2021, doi: 10.1109/MCSE.2021.3127288.
2. C. Bobda et al., "The future of FPGA acceleration in datacenters and the cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 3, pp. 1–42, 2022, doi: 10.1145/3506713.
3. Z. Han, S. Handagala, K. Patle, M. Zink, and M. Leeser, "A framework to enable runtime programmable P4-enabled FPGAs in the open cloud testbed," in *IEEE INFOCOM IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*. Hoboken, NJ, USA: IEEE, 2023, pp. 1–6, doi: 10.1109/INFOCOMWKSHPS57453.2023.10225877.
4. J. Xing et al., "A vision for runtime programmable networks," in *Proc. 20th ACM Workshop Hot Top. Netw.*, 2021, pp. 91–98, doi: 10.1145/3484266.3487377.
5. D. Diaconu, Y. Xie, M. Gungor, S. Handagala, X. Lin, and M. Leeser, "Machine learning across network-connected FPGAs," in *Proc. IEEE High Performance Extreme Comput. Conf. (HPEC)*. Boston, MA, USA: IEEE, 2023, pp. 1–7, doi: 10.1109/HPEC58863.2023.10363454.
6. R. Lin, Y. Cheng, M. De Andrade, L. Wosinska, and J. Chen, "Disaggregated data centers: Challenges and trade-offs," *IEEE Commun. Mag.*, vol. 58, no. 2, pp. 20–26, Feb. 2020, doi: 10.1109/MCOM.001.1900612.
7. Y. Shan, W. Lin, Z. Guo, and Y. Zhang, "Towards a fully disaggregated and programmable data center," in *Proc. 13th ACM SIGOPS Asia-Pacific Workshop Syst.*, 2022, pp. 18–28, doi: 10.1145/3546591.3547527.

8. K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated memory for expansion and sharing in blade servers," *ACM SIGARCH Comput. Architect. News*, vol. 37, no. 3, pp. 267–278, 2009. doi: 10.1145/1555815.1555789.
9. A. C. Elster, and T. A. Haugdahl, "Nvidia Hopper GPU and Grace CPU highlights," *Comput. Sci. Eng.*, vol. 24, no. 2, pp. 95–100, Mar./Apr. 2022, doi: 10.1109/MCSE.2022.3163817.
10. H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: A survey of opportunities, Challenges, and Applications," *ACM Comput. Surv.*, vol. 54, no. 11, pp. 1–32, 2019, doi: 10.1145/3510611.

MIRIAM LEESER is a professor of electrical and computer engineering and head of the Reconfigurable Computing Laboratory at Northeastern University, Boston, MA, 02115, USA. Contact her at mel@coe.neu.edu.

SURANGA HANDAGALA is a field-programmable gate array engineer at the Khoury College of Computer Sciences, Northeastern University, Boston, MA, 02115, USA. Contact him at s.handagala@northeastern.edu.

DANA DIACONU is a Ph.D. candidate in computer engineering and part of the Reconfigurable and GPU Computing Laboratory at Northeastern University, Boston, MA, 02115, USA. Contact her at diaconu.d@northeastern.edu.

ZHAOYANG HAN is a Ph.D. candidate in computer engineering at Northeastern University, Boston, MA, 02115, USA. Contact him at zhhan@coe.northeastern.edu.

MICHAEL ZINK is the Paros Professor in the Electrical and Computer Engineering Department, University of Massachusetts, Amherst, MA, 01003, USA. Contact him at mzink@umass.edu.