Dynamic Resource Allocation for HEVC Encoding in FPGA-Accelerated SDN Cloud

Panu Sjövall, Arto Oinonen, Mikko Teuho, Jarno Vanne, Timo D. Hämäläinen
Computing Sciences, Tampere University, Finland
{panu.sjovall, arto.oinonen, mikko.teuho, jarno.vanne, timo.hamalainen}@tuni.fi

***Abstract*—This paper presents a novel approach to accelerate, distribute, and manage video encoding services in large-scale cloud systems. A proof-of-concept application is Kvazaar HEVC intra encoder, whose functionality is partitioned between FPGAs and processors. Typically, only 1-2 FPGA boards can be attached per cloud server, which severely limits the flexibility of the cloud systems. Our solution is based on Software Defined Networking (SDN), in which practically any number of FPGAs and servers can be deployed. The system features a resource manager that is responsible for allocation, deallocation, and load balancing of resources upon service requests or changes in network infrastructure. Our prototype cloud system is composed of three Intel Xeon servers, two HP SDN switches, and two Intel Arria 10 FPGAs. The servers and FPGAs have 20GbE and 40GbE connections to the SDN switches, respectively. The prototype system can encode two 4K HEVC streams at 60 fps and the performance is predicted to scale almost linearly with the number of servers and FPGAs.**

Keywords— High Efficiency Video Coding (HEVC), Kvazaar HEVC encoder, field-programmable gate array (FPGA), Software-defined networking (SDN), High-level synthesis (HLS)

I. Introduction

Video coding, deep neural networks, and data analytics are the main drivers of hardware acceleration in cloud computing. Kvazaar HEVC intra encoder has been previously accelerated on a *field-programmable gate array* (*FPGA*) [1], [2] using *High Level Synthesis* (*HLS*) [3], [4] and 4× increase in coding speed was obtained with two FPGAs. However, the FPGA boards were connected to the server via the PCIe bus, which limits the number of FPGAs per server. Moreover, PCIe FPGA cards cannot fully act as independent computing nodes.

In this work, our ultimate goal is to deploy flexible combinations of servers and FPGAs, so that the same FPGA can be shared by many servers and vice versa. In addition, FPGA acceleration should act as a microservice for as easy deployment as software resources in cloud computing. The challenge is that implementing a full protocol stack for communication and application abstraction on an FPGA takes a major portion of the FPGA resources. Hence, it introduces too much overhead for an application. Our solution is to offload most of the network functionality from the FPGA by using *software-defined networking* (*SDN*), in which any data flow is programmable and the network interface can be at very low level [5]. Our main contributions are listed below:

- Dynamic resource allocation for HEVC encoding services on a changing setup of software and hardware resources
- Usage of SDN for offloading most network functions from the FPGA
- An advanced partitioning scheme for sharing execution between servers, FPGAs, and SDN switches
- HLS implementations for the network interface, control, and HEVC accelerator logic

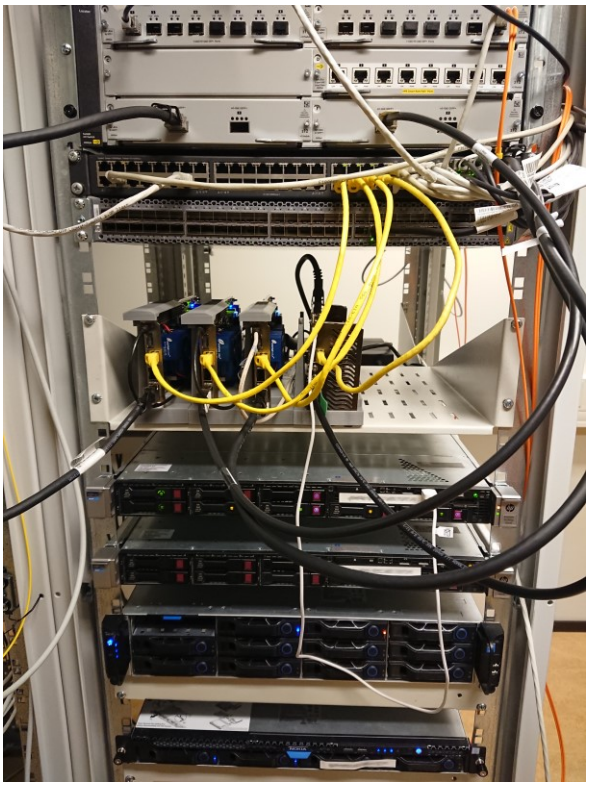

Fig. 1. Snapshot of the server rack.

- A prototype system implementing real-time 4K HEVC intra encoder

This paper is organized as follows. Section II considers the related work. Section III presents the proposed system consisting of Kvazaar FPGA accelerators, servers, and SDN switches. It also describes how the proposed resource manager and SDN are used to dynamically distribute HEVC encoding services between software and hardware resources. Section IV analyses the performance of the proposed system. Section V concludes the paper.

II. Related Work

The mainstream approaches for cloud FPGA acceleration are based on PCIe boards that are attached to the host server [6]-[9] or connected via Ethernet [10]-[12]. However, the host PCIe was still needed in [10] and all server traffic was routed through an FPGA making it even more tightly coupled to a server than in the other proposals. In [11] and [12], a complete network interface was implemented on an FPGA, so the full FPGA independency was attained at the cost of FPGA area.

For the time being, several HEVC encoders have been implemented on an FPGA [1], [2], [13]-[16], but none of them have utilized network interfaces between the processor and FPGA logic. To the best of our knowledge, this is the first paper that addresses fully independent FPGAs and servers in video encoding acceleration.

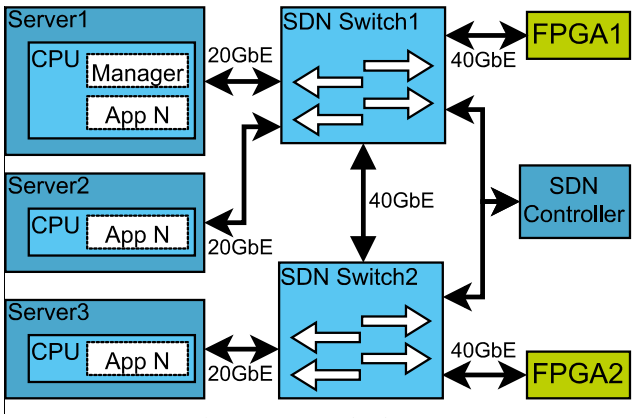

Fig. 2. Prototype cloud system.

TABLE I. CLOUD SYSTEM SPECIFICATIONS

Device	Type	CPU	Memory
Server1	HP Server	Xeon E5-2630	96GB
Server2	Nokia Airframe Cloud server [17]	Xeon E5-2680v4	256GB
Server3	Nokia Airframe Cloud server [17]	Xeon E5-2680v3	256GB
Switch1	HPE FlexFabric 5900AF 48G 4XG 2QSFP+	-	-
Switch2	HP Switch 5406Rzl2	-	-
FPGA1	Intel Arria 10 GX FPGA Dev Kit [18]	-	-
FPGA2	Intel Arria 10 GX FPGA Dev Kit [18]	-	-
Controller	HP VAN SDN Controller [19]	-	-

III. PROPOSED SYSTEM PARTITIONING

Fig. 1 and 2 show a snapshot and the corresponding network structure of our prototype cloud system, respectively. Table I tabulates the component specifications for Fig. 2. The SDN controller manages connections between the servers and FPGAs by modifying data flows in SDN switches. Each FPGA is connected to an SDN switch via one *40 Gigabit Ethernet* (*40GbE*) link and each server with 2×10GbE links.

A. Server Interfacing

In the proposed system, the servers use Linux operating system, e.g., CentOS or Ubuntu. Each server has two 10GbE SFP interfaces, which are configured to use IEEE 802.3ad Dynamic link aggregation (802.3ad, LACP) that combines the interfaces into a single load-balanced logical link with an effective bandwidth of 20GbE. As the proposed system operates on Ethernet frames, the criteria for load balancing are derived from the source and destination MAC addresses and the Ethernet type.

Because the proposed system utilizes the data link layer (layer 2), there are no built-in reliability mechanisms available with Ethernet frames. Therefore, the CPU and FPGA keep track on how many packets need to be received and sent for each *coding tree unit* (*CTU*) [20] in HEVC encoding. A lost packet causes a timeout and the same data is then re-sent to the FPGA for re-encoding. Using the IPv4 and UDP protocols from the network and transport layers (layer 3-4) would add some overhead in data rates and FPGA design complexity, but it would make it possible to send packets over different LANs. Wrapping the CTU payload inside the UDP packets would allow inclusion of UDP ports in the criteria for load balancing. These aspects will be addressed in the future.

B. FPGA Interfacing

Fig. 3 depicts the network interface on the FPGA. It includes Intel 40G Ethernet IP block and our own implementations of RX/TX Parsers and ETH Writer modules. The RX Parser decodes the Ethernet frames, ensures that the incoming frame is valid, and configures the correct accelerator instance. The TX Parser is responsible for generating the

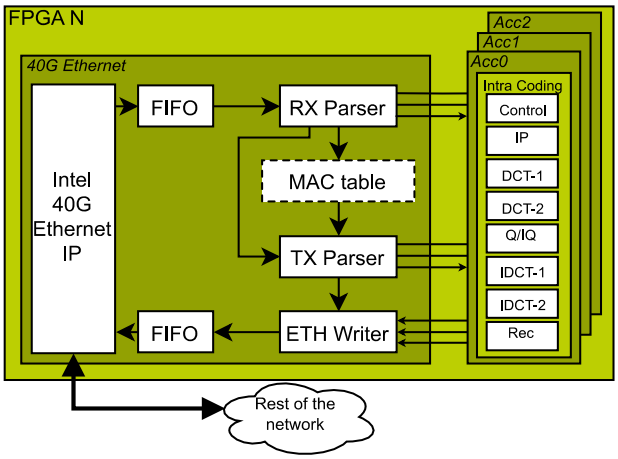

Fig. 3. Proposed FPGA accelerator architecture.

TABLE II. KVAZAAR INTRA CODING SETTINGS

Feature [20]	Kvazaar parametrization
Coding unit sizes	16×16, 8×8
Prediction unit sizes	16×16, 8×8
Transform unit sizes	16×16, 8×8
Intra prediction modes	35 (DC, planar, 33 angular)
Parallelization	Wavefront parallel processing Picture-level
Sample adaptive offset	Disabled
Sign bit hiding	Disabled
RD optimized quantization	Disabled
Transform skip	Disabled
Quantization parameter	22

Ethernet headers, gathering the payload, and controlling the ETH Writer. The frame size and the number of Ethernet frames generated per CTU are configurable. Implementing these three modules in C and using Catapult-C HLS tool [21] simplified the design process on FPGA and lowered the bar for design iterations over the corresponding approaches with VHDL or Verilog. With HLS, these blocks could also be easily modified for any accelerator usage.

The RX and TX parsers utilize a look-up-table for MAC addresses to identify which server sent the CTU for encoding. This way, the server MAC address can be translated and the results are sent back to the correct server. This approach also allows multiple servers to use the same FPGA at the same time and all results are forwarded back correctly. Fast FIFO memories compensate for differences in data widths and rates between the physical 40G Ethernet IP and our FPGA logic.

C. Kvazaar Cloud FPGA Accelerator

The execution of Kvazaar encoding is partitioned between CPUs and FPGA accelerators. First, the CTU structures of a video frame are initialized by the CPU and sent to the accelerator which implements most of the coding tools. Only the final steps, *context-adaptive binary arithmetic coding* (*CABAC*) and video stream construction, are left for the CPU. The implementation supports Kvazaar ultrafast preset [22] detailed in Table II.

A block diagram of the accelerator architecture is also shown in Fig. 3. The core component is the Kvazaar HEVC intra coding unit [1], [2], which was implemented with Catapult-C HLS tool from Kvazaar [23] open-source C code. Altogether, three accelerator instances (*Acc0*, *Acc1*, and *Acc2*) can be placed on a single Arria 10 FPGA. Each accelerator is able to process up to 16 CTUs in parallel. Software parts of the encoder can be executed on any server running Linux.

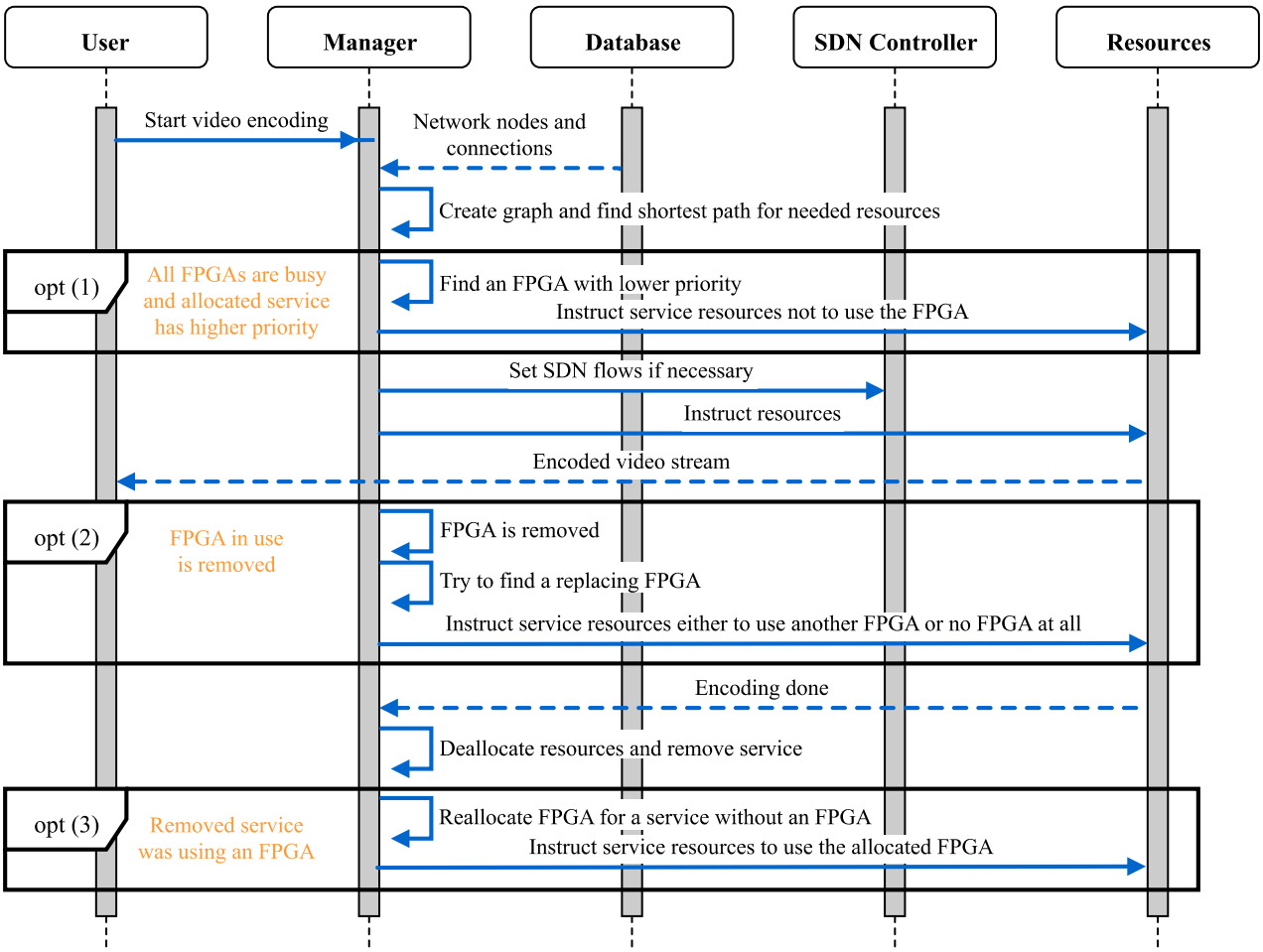

Fig. 4. Message sequence chart of dynamic resource allocation for a HEVC encoding service.

D. SDN

The SDN-controlled switches make it easy to connect the FPGAs to the network. As the data flows are automatically set by the resource manager, the FPGA does not need to support a full set of Internet protocols. Different FPGAs are identified by their MAC addresses, which is sufficient for routing the packets correctly. For example, when the SDN controller sees a MAC address assigned to a certain FPGA, it routes all associated packets to it. The data from the FPGA is routed back to the server when the source refers to the FPGA and the destination is the MAC address of the server.

E. Dynamic Resource Allocation Manager

Fig. 4 shows a message sequence chart of how our dynamic resource allocation is used for an HEVC encoding service. First, a Kvazaar HEVC encoding service is started by a user request. The resource manager collects all the needed network components including devices, switches, and connections from the database. Then, it creates a network graph and defines the most economical paths for the components, e.g., the shortest paths from a video source to a server and from the server to an FPGA. The same server and FPGA can be allocated multiple times to different services, but by monitoring the resource usage, the manager tries to optimize resource utilization for the best performance.

The manager also supports prioritization of services. Encoding speeds can thus be balanced by giving higher resolution videos a higher priority in FPGA acceleration. When a higher priority service is invoked and no FPGAs are available, the manager moves the execution of a lower priority service from the FPGA to the CPU, as shown in Fig. 4 with the option (1).

After the manager has allocated the needed resources, it sets the necessary SDN flows by using the API of the SDN controller. For example, in Fig. 2, *Server1* can access the *FPGA2* connected to a different switch by using their original MAC addresses, without any *Address Resolution Protocol* (*ARP*) messages. After the setup is ready, the manager uses POST messages to inform the resources to start the service, maybe with some additional configuration information (e.g., IDs and encoding parameters).

The manager brings robustness to the encoding process, e.g., the system is able to recover from FPGA removal. When an FPGA is switched off, the manager automatically switches the services from the removed FPGA to a CPU and starts checking equivalent replacements for the removed FPGA. This is illustrated in Fig. 4 by the option (2). Switching an encoding service from FPGA to CPU takes around one second due to the implemented encoder timeouts. Instead, switching between FPGAs and from CPU to FPGA take place instantaneously.

After a service is completed, the manager deallocates the resources in use. If an FPGA is deallocated, it is automatically assigned to the next service having no assigned FPGA, as described in Fig. 4 with the option (3). The reallocation favors services with the highest priority and the longest running time on a CPU.

TABLE III. PERFORMANCE OF KVAZAAR HEVC INTRA ENCODER ON CPU, CPU + PCIE FPGA CARD [2], AND THE PROPOSED SYSTEM (CPU + FPGA)

Sequence [24] (2160p)	Speed (fps)			Avg. frame latency (ms)			CPU utilization (%)		
	CPU only	PCIe	Proposed	CPU only	PCIe	Proposed	CPU only	PCIe	Proposed
Beauty	31	70	60	49	24	33	94	56	60
Bosphorus	43	70	62	33	18	25	94	33	35
HoneyBee	34	70	61	41	19	27	96	49	48
Average	36	70	61	41	20	28	95	46	48

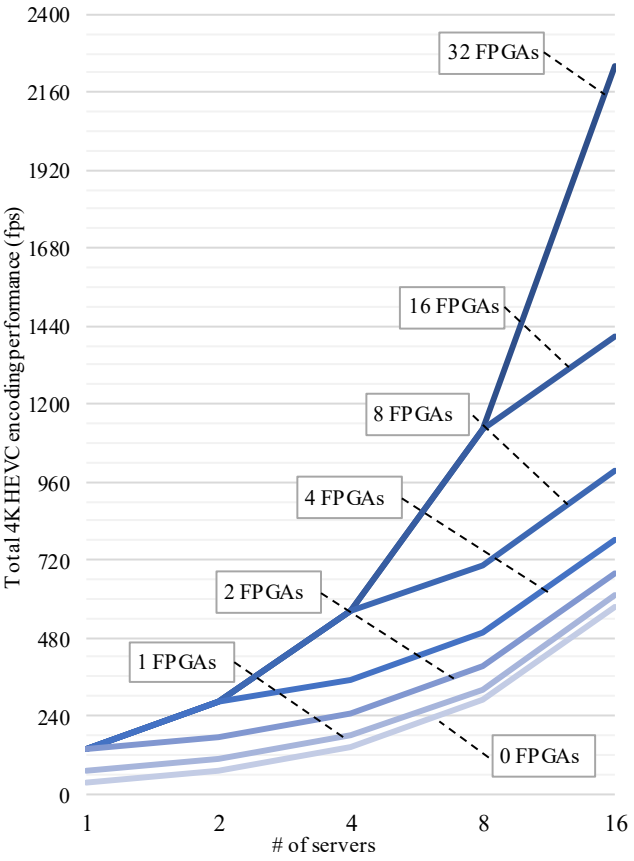

Fig. 5. Predicted encoding performance with differently scaled setups.

IV. PERFORMANCE ANALYSIS

Table III reports the performance results for Kvazaar HEVC encoder on three different platforms: 1) CPU-only; 2) CPU with a single PCIe FPGA card [2]; and 3) the proposed prototype cloud system containing a single CPU and FPGA. For fair comparison, the resources of the cloud setup are unified with that of the PCIe approach. Furthermore, all these setups use an equivalent Intel Xeon CPU and the latter two equivalent Arria 10 FPGA for acceleration.

According to our results, the proposed cloud approach speeds up HEVC encoding by 1.5-2 fold over that of the CPU-only case. However, the average coding speed of our proposal is around 9 fps slower than that of the PCIe approach. There are three reasons for the slowdown: the usage of a 20Gbps Ethernet link in place of a 32Gbps PCIe bus, the overhead of Ethernet packets, and reduced parallelism due to longer waiting times of Ethernet frames. Nevertheless, our proposal is still able to encode 4K resolution test videos at 60 fps.

It is notable that the average frame latency with the fiber connection is around 40% higher than that of the PCIe bus. On the other hand, the Ethernet interfacing still has 31% smaller average frame latency than with CPU-only encoding. The CPU utilization is nearly the same in the cloud and PCIe approaches. Both solutions accelerate HEVC encoding and still use around 50% less CPU resources than the CPU-only case.

The proposed system is able to encode 4K video at 90 fps with two FPGAs and a single server (Fig. 2). In this case, the maximum speed is limited by the 20GbE connection. Alternatively, our system can encode two 4K sequences at 60 fps with two servers and two FPGAs. Using 40Gbps network cards on the servers in place of 2×10GbE would remove this limitation and provide smaller latency as well as faster encoding speed. However, acquiring these network cards is left for the future.

The system also appears to be robust, as Kvazaar execution can be switched between CPUs and FPGAs on the fly depending on the resource availability. This means, in practice, that the system can switch an encoding process between FPGA accelerators and recover from an FPGA removal, all without interrupting the encoding process. These features are visualized in [25].

Despite the minor performance penalty, the fiber connected FPGAs allow much better scalability than the dedicated PCIe-based approach. For example, both approaches would need three Kvazaar accelerator instances on FPGA for 4K60p encoding, but only the cloud approach is able to attain the same speed with three smaller FPGAs, each having a single accelerator instance.

The proposed dynamic resource allocation and partitioning scheme leaves lots of room for further performance scaling. Fig. 5 predicts the total encoding performance of differently scaled up systems with the 40GbE links in servers. For example, the graphs show equal performance for the systems composed of 16 servers or 4 servers with 8 FPGAs. A high-end system with 16 servers and 32 FPGAs has potential to encode 64 HEVC streams at 4K30p or 16 streams at 4K120p simultaneously.

V. CONCLUSION

This paper presented an automated approach for managing services with a lightweight framework that connects multiple servers and FPGAs in an SDN based cloud. The combination of a dedicated resource manager and SDN makes it possible to have practically any number of independent FPGAs on the network without wasting FPGA resources for communication and application abstraction.

Instead of using complicated network protocols, the proposed system uses the SDN controller and SDN switches for routing data. A dedicated SDN controller allows scaling the network to a large-scale cloud infrastructure without losing the speed and connectivity of a small network.

The proposed system was also validated in practice with a proof-of-concept real-time 4K HEVC encoder implementation. It was shown to attain near the same speed as the previous PCIe equivalent implementation but with much better scalability and robustness.

ACKNOWLEDGMENT

This work was supported in part by the European Celtic-Plus project VIRTUOSE, the Academy of Finland (decision no. 301820), Nokia Foundation, and the Finnish Foundation for Technology Promotion.

REFERENCES

[1] P. Sjövall, V. Viitamäki, A. Oinonen, J. Vanne, T. D. Hämäläinen, and A. Kulmala, "Kvazaar 4K HEVC intra encoder on FPGA accelerated Airframe server," *in Proc. IEEE Workshop Signal Process. Syst.*, Lorient, France, Oct. 2017.

[2] P. Sjövall, V. Viitamäki, J. Vanne, T. D. Hämäläinen, and A. Kulmala, "FPGA-powered 4K120p HEVC intra encoder," *in Proc. IEEE Int. Symp. Circuits Syst.*, Florence, Italy, May 2018.

[3] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? A study on the state of high-level synthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst*., vol. 38, no. 5, May 2019, pp. 898-911.

[4] P. Coussy, D. D. Gajski, M. Meredith, and A. Takach, "An introduction to high-level synthesis," *IEEE Des. Test Comput.*, vol. 26, no. 4, Jul.-Aug. 2009, pp. 8-17.

[5] M. Vajaranta, V. Viitamäki, A. Oinonen, T. D. Hämäläinen, A. Kulmala, and J. Markunmäki, "Feasibility of FPGA accelerated IPsec on cloud," *in Proc. Euromicro Symp. Digit. Syst. Des.*, Prague, Czech Republic, Aug. 2018.

[6] A. Putnam *et al*., "A reconfigurable fabric for accelerating large-scale datacenter services," *IEEE Micro*, vol. 35, no. 3, May-June 2015, pp. 10-22.

[7] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA accelerators for efficient cloud computing," *IEEE Int. Conf. Cloud Comput. Technol. Sci*, Vancouver, British Columbia, Canada, Nov.-Dec. 2015.

[8] Z. Zhu, A. X. Liu, F. Zhang, and F. Chen, "FPGA resource pooling in cloud computing," *IEEE Trans. Cloud Comput*., Early Access.

[9] J. Lallet, A. Enrici, and A. Saffar, "FPGA-based system for the acceleration of cloud microservices," *in Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast.*, Valencia, Spain, Jun. 2018.

[10] A. M. Caulfield *et al*., "A cloud-scale acceleration architecture," *in Proc. Annual IEEE/ACM Int. Symp. Microarchitecture*, Taipei, Taiwan, Oct. 2016.

[11] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling FPGAs in hyperscale data centers," *in Proc. IEEE Int. Conf. Ubiquitous Intell.,* Beijing, China, Aug. 2015.

[12] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached FPGAs for data center applications," *in Proc. Int. Conf. Field-Programmable Technol.*, Xi'an, China, Dec. 2016.

[13] J. Zhu, Z. Liu, D. Wang, Q. Han, and Y. Song, "HDTV1080p HEVC intra encoder with source texture based CU/PU mode pre-decision," *in Proc. Asia and South Pacific Design Automation Conf.*, Singapore, Jan. 2014.

[14] G. Pastuszak and A. Abramowski, "Algorithm and architecture design of the H.265/HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, Jan. 2016, pp. 210-222.

[15] S. Atapattu, N. Liyanage, N. Menuka, I. Perera, and A. Pasqual, "Real time all intra HEVC HD encoder on FPGA," *in Proc. IEEE Int. Conf. Application-specific Syst., Architectures and Processors*, London, United Kingdom, Jul. 2016.

[16] K. Miyazawa, H. Sakate, S. Sekiguchi, N. Motoyama, Y. Sugito, K. Iguchi, A. Ichigaya, and S. Sakaida, "Real-time hardware implementation of HEVC video encoder for 1080p HD video," *in Proc. Picture Coding Symp.*, San Jose, California, USA, Dec. 2013.

[17] *AirFrame data center solution.* Accessed on: Sep. 20, 2019. [Online]. Available: https://www.nokia.com/networks/solutions/airframe-data-center-solution/

[18] *Arria 10.* Accessed on: Sep. 20, 2019. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/boards_and_kits/dev-kits/altera/kit-a10-gx-fpga.html

[19] *HP Virtual Application Networks SDN Controller.* Accessed on: Sep. 20, 2019. [Online] Available: https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c03967699

[20] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1649-1668.

[21] *Catapult High-Level Synthesis.* Accessed on: Sep. 20, 2019. [Online]. Available: https://www.mentor.com/hls-lp/catapult-high-level-synthesis/

[22] *Kvazaar HEVC encoder.* Accessed on: Sep. 20, 2019. [Online]. Available: https://github.com/ultravideo/kvazaar

[23] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: open-source HEVC/H.265 encoder," *in Proc. ACM Int. Conf. Multimedia*, Amsterdam, The Netherlands, Oct. 2016.

[24] *Test Sequences.* Accessed on: Sep. 20, 2019. [Online]. Available: http://ultravideo.cs.tut.fi/#testsequences

[25] P. Sjövall, M. Teuho, A. Oinonen, J. Vanne, and T. D. Hämäläinen, "Visualization of dynamic resource allocation for HEVC encoding in FPGA-accelerated SDN cloud," *in Proc. IEEE Int. Conf. Visual Commun. Image Process.*, Sydney, Australia, Dec. 2019.