FPGA-Powered 4K120p HEVC Intra Encoder

Panu Sjövall, Vili Viitamäki, Jarno Vanne, Timo D. Hämäläinen, Ari Kulmala*
Laboratory of Pervasive Computing, Tampere University of Technology, Tampere, Finland
*Datacenter Infrastructure Modules, Nokia, Tampere, Finland

Abstract— **This paper presents a hardware-accelerated Kvazaar HEVC intra encoder for 4K real-time video coding at up to 120 fps. The encoder is implemented on a Nokia AirFrame Cloud Server featuring a 2.4 GHz dual 14-core Intel Xeon processor and two Arria 10 PCI Express FPGA accelerator cards. The presented encoder is a speed-optimized version of our 1st generation 4K40p HEVC intra encoder. The proposed speedup techniques include 1) Increasing the number of FPGA cards to two; 2) Remapping the simplest multiplications from DSP blocks to logic for better FPGA utilization; 3) Making task scheduling more flexible to improve utilization rate of hardware accelerators; and 4) Increasing the pipeline depth and duplicating time-sensitive resources in the hardware accelerator. As a result, up to three hardware accelerator instances can be accommodated in a single Arria 10 so the encoder is able to make use of six accelerators. According to our experiments, the proposed encoder obtains threefold speedup over our 1st generation encoder. Our proposal is also shown to outperform all other encountered FPGA and ASIC implementations.**

Keywords— High Efficiency Video Coding (HEVC); Ultra High Definition Television (UHDTV); Kvazaar Intra coding; field-programmable gate array (FPGA); real-time

I. Introduction

Live Internet video is forecast to grow 15-fold in five years, accounting for 13% of all Internet video traffic by 2021 [1]. This growth comes from a plurality of new end users and multimedia applications but also from higher spatial and temporal video resolutions that are rapidly gaining ground. For example, 4K *Ultra High Definition Television* (*UHDTV*) format features 3840 × 2160 pixels (2160p) and frame rates up to 120 *frames per second* (*fps*) [2].

Despite the fast progress of transmission and storage technologies, the holistic growth of video volume makes more efficient video coding inevitable. The latest international video coding standard, *High Efficiency Video Coding* (*HEVC/H.265*) [3], [4], is developed to address these needs. This work deals with *all-intra* (*AI*) coding configuration [5] of HEVC Main Profile. It is shown to improve intra coding efficiency by 23% over that of the preceding standard AVC/H.264 [6] for the same objective quality, but at a cost of over threefold increase in coding complexity [7]. Therefore, implementing a real-time HEVC intra encoder for UHDTV format with a reasonable coding efficiency, implementation cost, and power budget requires efficient encoder optimizations and powerful computing platforms.

Multithreading [8] and *single instruction multiple data* (*SIMD*) optimizations [9] are primary design techniques for complexity reduction in *software* (*SW*) HEVC encoders. Further speedup and lower power dissipation is typically sought by offloading the compute-intensive coding tools to *hardware* (*HW*) accelerators or implementing the entire HEVC encoder on HW [10]-[13].

Our recent work [14] shows that a pure SW implementation of HEVC intra encoder is able to attain real-time coding speed for 4K30p format and formats up to 4K60p can be supported by using several software encoder instances in parallel [15]. The respective speeds are also reported for HW accelerated intra encoders [11], [16] and high-end frame rates of 4K UHDTV format are only reached with several HW encoder instances [13].

The main motivation of this work was to implement a real-time HEVC intra encoder for up to 4K120p format. The presented solution is a direct continuation to our previous work [16] where Kvazaar open-source HEVC encoder [17] is accelerated to encode 4K video at 40 fps on Nokia AirFrame Cloud Server [18]. The adopted server setup included a 2.4 GHz dual 14-core Xeon processor and an Arria 10 *PCI Express* (*PCIe*) FPGA accelerator card. Servers like AirFrame have gained a lot of traction in the recent years due to the advent of cloud gaming, telco clouds, and edge computation.

In this work, the same AirFrame server is equipped with two Arria 10 PCIe cards. In addition, up to three HW accelerator instances can be accommodated on a single FPGA by remapping the simplest multiplications to logic blocks and only allocating DSP blocks to the most compute-intensive multiplications. Individual HW accelerator instances are also boosted by using a higher pipeline depth and duplicated resources, whereas a proposed task scheduling improves the utilization rate of the instances. Together, the proposed techniques result in around threefold encoding speed over that of [16].

The original HW accelerator is implemented in [16] with Catapult C [19] *high-level synthesis* (*HLS*) [20] tool that enables automatic hardware description language generation from C source code of Kvazaar. The same approach is applied in this work since HLS offers much shorter design and verification times than manual design approaches. This is particularly true in resource remapping and pipeline modifications.

The remainder of this paper is structured as follows. Section 2 describes the applied platform and the selected SW/HW partitioning of Kvazaar on it. Section 3 presents the pipeline optimizations made for the HW accelerator instances. Section 4 introduces the proposed task-scheduling scheme among the accelerator instances. In Section 5, 4K performance of the proposed encoder is benchmarked against our earlier solution and other prior-art. Section 6 concludes the paper.

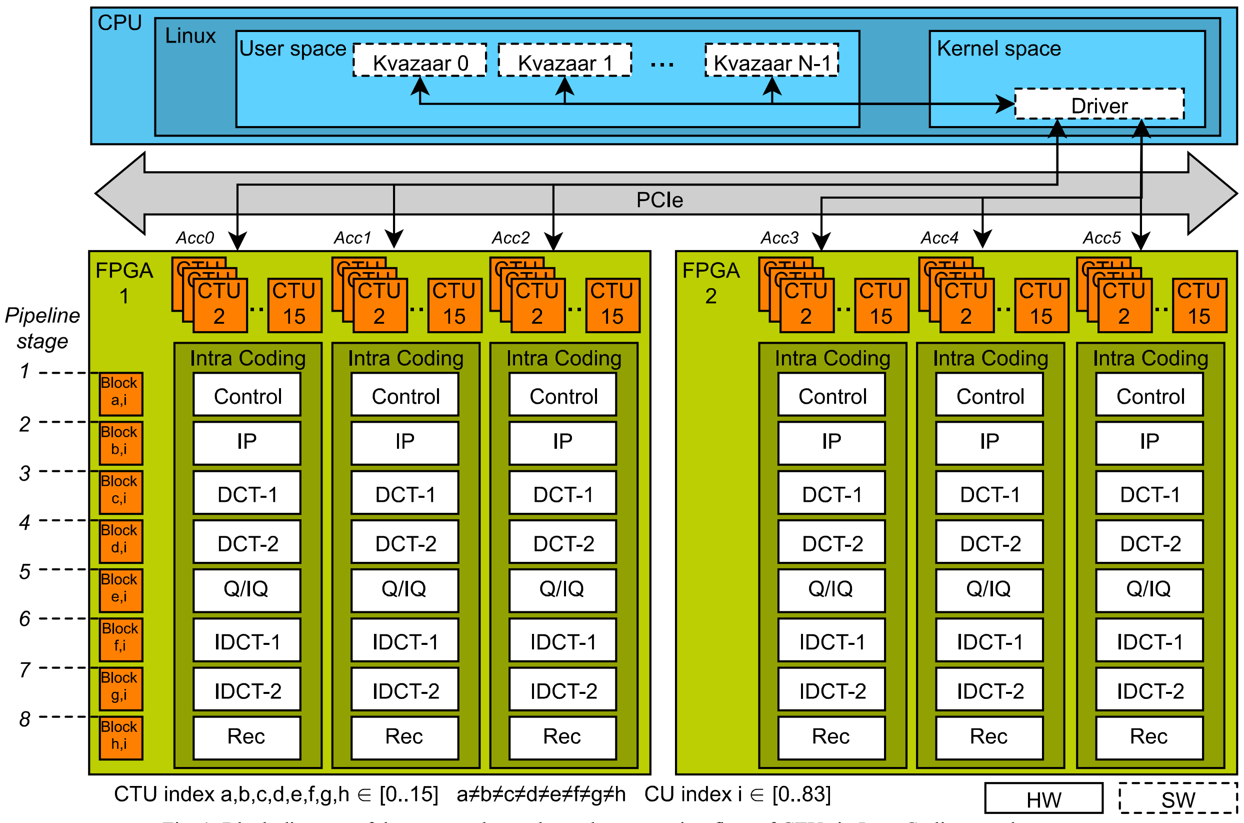

Fig. 1. Block diagram of the proposed encoder and a processing flow of CTUs in Intra Coding accelerators.

II. Overview of the Proposed System

Fig. 1 shows the block diagram of the underlying SW/HW platform. The backbone of the system is a Nokia AirFrame Cloud server [18] with two Xeon E5-2680 v4 processors and 256 GB of memory. Two Arria 10 FPGA cards are connected to the CPU via a PCIe bus. The operating system is CentOS 6.8.

A. Kvazaar Partitioning

On Xeon processors, Kvazaar [17] is run in the user space and the Linux driver in the kernel space. The Linux driver is used for the CPU-PCIe-FPGA interfacing. A single Arria 10 FPGA has enough resources for three Intra Coding accelerator instances including the needed peripherals and on-chip memories. The FPGA interface is made of the Avalon-MM Hard IP for PCIe, separate *Direct Memory Access* (*DMA*) blocks for reading and writing, and the on-chip memories of the Intra Coding accelerator. A more detailed functionality of the platform is described in our previous work [16].

Kvazaar implements a basic HEVC block partitioning in which the pictures are partitioned into *coding tree units* (*CTUs*) of size 64 × 64. CTUs can be optionally divided into four equal-sized *coding units* (*CUs*) and the division can be recursively continued until the maximum hierarchical depth of the HEVC quadtree is reached. The proposed encoder supports Kvazaar ultrafast preset [17] with extended coding tree depth so that CUs of size 32 × 32, 16 × 16, and 8 × 8 are supported. It also implements *Wavefront Parallel Processing* (*WPP*) and picture-level parallel processing for parallel CTU coding. These schemes can be enabled concurrently.

The most computationally intensive Kvazaar coding tools including *intra prediction* (*IP*), *discrete cosine transform* (*DCT*), *quantization* (*Q*), *inverse Q* (*IQ*), *inverse DCT (IDCT*), and *reconstruction* (*Rec*) are implemented with HLS and synthesized to FPGA. *Context-adaptive binary arithmetic coding* (*CABAC*) and other control-intensive coding tools are executed on CPU. In addition, the CPU takes care of raw input video reading and outputting the encoded bit stream. Mapping the major share of CTU coding to FPGA decreases the power dissipation through lower CPU usage and accelerates the whole encoding process.

B. System Configuration

In this work, the FPGA driver is upgraded to support practically any number of FPGAs, but the FPGA count is here limited to two by the available PCIe slots. Therefore, the system can contain six accelerator instances (Acc_0 - Acc_5) at maximum.

The proposed system is also configurable at run time to the chosen number of Kvazaar instances without any performance compromises. This way, the user can choose whether to encode a single video with the maximum speed or several videos in parallel. Different Kvazaar instances can also encode input videos with different encoding parameters and resolutions at the same time. This is made possible by processing each CTU individually in the Intra Coding accelerators.

III. Proposed Hardware Pipeline

Fig. 1 illustrates the processing flow of CTUs in Intra Coding accelerators. Each accelerator is able to take care of 16 CTUs

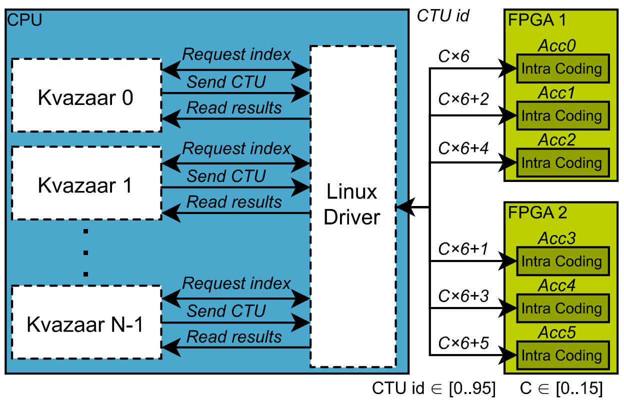

Fig. 2. CTU load balancing between Kvazaar instances and accelerators.

(0..15) simultaneously, so up to 96 CTUs can be under way in parallel with six accelerators. An eight-stage pipeline of a single accelerator can process eight blocks of separate CTUs at a time and the remaining eight CTUs are buffered for faster access. The processed blocks move sequentially through HEVC encoding stages. Altogether, each CTU can contain 4 + 16 + 64 = 84 separate CUs at maximum when CUs of size 32×32, 16×16, and 8×8 are supported.

A. *Intra Prediction Pipelining*

In our 1st generation encoder, IP and the creation of reference pixels were done in the same pipeline stage. Generating the reference pixels from the border pixels caused an overhead, which almost doubled the delay of the IP stage with 8×8 blocks. Therefore, the reference pixel generation was moved from the IP stage to the control stage. Now, the reference pixels of successive CUs from different CTUs are generated and buffered in advance. This way, the control stage is not blocked by the IP and the IP has an adequate small delay between predictions.

B. *DCT / IDCT Pipelining*

Our 1st generation encoder used only a single transform unit for the DCT and another unit for the IDCT, i.e., both algorithms ran the transform twice with the same transform unit. First from the input and second from the transpose memory. Although this design had sufficient speed for smaller number of parallel CTUs in a single Intra Coding accelerator, it caused a bottleneck when aiming higher CTU parallelism.

In the proposed work, there are two transform units for both the DCT and IDCT. In addition, the memory size of the transpose memories was doubled, allowing all transform units to run at the same time and enabling successive block pipelining. This modification practically doubled the processing speed of DCT [21] and IDCT [22] and increased the overall hardware pipeline by two stages. Although this modification increased the area of the whole Intra Coding accelerator, the speed improvement was more significant.

C. *Remapping Multiplications from DSP to Logic*

Our prior encoder implementations relied heavily on DSPs, mostly because they were implemented on FPGAs having half the logic area but still ~75% of the DSPs of Arria 10. Hence,

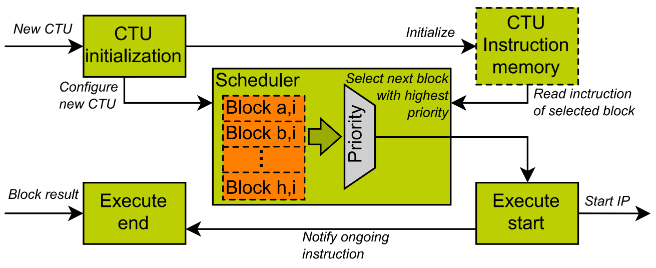

Fig. 3. Block scheduling in Intra Coding accelerator.

adding a third Intra Coding accelerator would have caused Arria 10 to run out of DSPs.

Even though the DCT and IDCT transform units were doubled in this work, we were able to fit a third Intra Coding accelerator in a single Arria 10 FPGA. This was achieved by replacing all DSPs in IP and DST transform as well as constant multiplications in DCT and IDCT with logic elements. More economic utilization of DSPs and other HLS code optimizations allowed for better routing of our design on FPGA and made it possible to increase the maximum frequency from 125 MHz to 175 MHz.

D. *Other Optimizations*

In our 1st generation encoder, a single Intra Coding accelerator supported eight parallel CTUs and the CPU was used to encode CTUs whenever the accelerators had no space for a new CTU. In this work, the additional CPU encoding was not used anymore since the proposed improvements have made the accelerator much faster at processing a CTU than the CPU. In addition, the increase of parallel CTUs supported on a single Intra Coding accelerator from eight to 16, caused encoding even a single CTU with the CPU to bottleneck the system. Waiting for available processing time from the accelerators and waiting for the result is faster than encoding a CTU with SW.

Performing the CTU encoding solely on the FPGA reduced the overall CPU usage and the CPU is now mostly waiting for results from the FPGA. This allows the CPU to perform other processing, even while encoding HEVC 4K120p. Further improving the CPU utilization and maximizing thread usage, the DMAs in the FPGA now generate interrupts when ready. Previously, the kernel driver polled the DMAs, but the increase in FPGAs and accelerators caused the locking mechanism in the kernel to use a major part of the processing time of a thread. With interrupts and semaphores, the thread can now sleep while waiting for the DMA completion and yield processing time for other threads.

IV. Task Scheduling and Resource Management

Scheduling of intra coding tasks is also improved to make the most of Kvazaar SW instances on a CPU and Intra Coding accelerators on FPGA.

A. *CTU Load Balansing*

Fig. 2 shows the process of scheduling processing time for different Kvazaar instances and choosing the best available Intra

TABLE I. CODING SPEED OF 4K VIDEO WITH DIFFERENT NUMBER OF INTRA CODING ACCELERATORS

Sequence (2160p)	Software	Single FPGA			Two FPGAs		
	No acceleration	1 accelerator	2 accelerators	3 accelerators	2 accelerators	4 accelerators	6 accelerator
	Speed (fps)	Speed (fps)	Speed (fps)	Speed (fps)	Speed (fps)	Speed (fps)	Speed (fps)
Beauty	17	25	49	64	50	96	125
Bosphorus	20	27	53	65	54	102	127
HoneyBee	17	26	50	64	51	98	124
Jockey	21	29	54	65	58	104	126
ReadySetGo	19	27	52	64	53	99	123
ShakeNDry	16	22	44	63	45	85	115
YachtRide	18	26	51	64	51	98	123
Average	18	26	50	64	52	97	123

TABLE II. COMPARISON OF THE PROPOSED AND RELATED INTRA ENCODERS

Architecture	Technology	Frequency	Resolution	Cells	DSPs
[10]	ASIC	357 MHz	1080@44fps	2296k gates	-
[11]	ASIC	200/400 MHz	2160@30fps	1086k gates	-
[11]	Arria II	100/200 MHz	1080@60fps	93k ALUTs	481
[12]	Zyng ZC706	140 MHz	1080@30fps	84k LUTs	34
[13]	Custom 3x FPGA	N.A.	1080@60fps	N.A.	-
[16]	CPU + Arria 10	125 MHz	2160@40fps	308k ALUTs	862
Proposed	CPU + Arria 10	175 MHz	2160@60fps	552k ALUTs	1227

Coding accelerator for a new CTU. The Linux driver is accessible by all Kvazaar instances, which request processing time on the FPGA from the driver. If there are no available resources, Kvazaar instances need to wait. Waiting instances are served in request order. The driver assigns new CTUs to different Intra Coding accelerators according to the CTU id provided by the driver. The CTU id is a running number limited by available resources, i.e., the number of Intra Coding accelerators and the number of CTUs per accelerator.

B. *Block Scheduling in Intra Coding Accelerator*

Fig. 3 shows how the Intra Coding accelerator determines the next block to the pipeline. For each CTU, a set of instructions are generated to signal the scheduler the encoding order of the blocks in a CTU. The next block of a CTU is valid for processing if the previous block of the same CTU is done. The scheduler assigns priorities to the valid blocks and chooses the one with the highest priority. The priority is higher when the next block in line is of equal size or larger than the previous one. This policy aims to keep the pipeline utilization high and it prevents larger blocks from bottlenecking smaller ones.

V. CODING SPEED ANALYSIS

Table I tabulates the obtained encoding speeds with different number of Intra Coding accelerators using the 8-bit 4:2:0 4K120p test video sequences from [23]. The average results show that our implementation is able to reach 4K30p with two accelerators, 4K60p with three accelerators, and 4K120p with six accelerators. The maximum speed of the accelerated system is 6.8 times as high as that of the pure software version. Coarsely speaking, doubling the number of accelerators doubles the encoding speed.

Our 1st generation encoder was able to encode 4K30p with a single Intra Coding accelerator but it was limited to use CU sizes of 8×8 and 16×16. In addition, it utilized the remaining CPU power for CTU encoding. Disabling 32×32 blocks in the current version would also increase its 4K coding speed to 30 fps with a single accelerator even without utilizing the CPU. With medium preset [17] and *rate-distortion-optimized quantization* (*RDOQ*) disabled, our proposal is able to encode 4K60p with six Intra Coding accelerators.

Table II tabulates the performance figures of the proposed and existing HEVC intra encoders on ASIC and FPGA. To make comparison more straightforward, our proposal is configured to use only a single FPGA with which 4K format can be encoded up to 60 fps (Table I). Our 1st generation encoder was already able to outperform related FPGA implementations and compete equally with the existing ASIC implementations. The proposed 2nd generation encoder even beats these ASIC approaches.

VI. CONCLUSION

This paper presented our 2nd generation HEVC encoder for real-time 4K intra coding. The proposed encoder was prototyped on Nokia AirFrame Cloud Server composed of a dual 14-core Intel Xeon processor and two Arria 10 FPGAs. On AirFrame, our solution is able to encode 4K video at 120 fps or four 4K videos at 30 fps.

The implemented HW acceleration speeds up the encoder by 6.8 times over the pure SW implementation and the obtained performance is three times as high as that of our 1st generation encoder. The speedup was achieved by increasing the number of FPGAs to two, improving FPGA utilization by allocating the simplest multiplications to logic, increasing the efficiency of pipeline in Intra Coding accelerator, and improving the utilization rate of the accelerators by better task scheduling.

The Intra Coding accelerators of the encoder are entirely implemented with HLS tools from C source code of Kvazaar HEVC intra encoder. HLS is generally known to reduce design and verification times over traditional design flows. This work further shows that the shorter development time does not come at a cost of coding performance.

ACKNOWLEDGMENT

This work was supported in part by Nokia, the European Celtic-Plus Project 4KREPROSYS, and the Academy of Finland (decision no. 301820).

REFERENCES

[1] Cisco, *Cisco Visual Networking Index: Forecast and Methodology*, 2016-2021, Jun. 2017.

[2] *Parameter values for ultra-high definition television systems for production and international programme exchange*, document ITU-R Rec. BT.2020-2, ITU-R, Oct 2015.

[3] *High Efficiency Video Coding*, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, Apr. 2013.

[4] 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.

[5] J. Lainema, F. Bossen, W. J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1792-1801.

[6] *Advanced Video Coding for Generic Audiovisual Services*, document ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC, Mar. 2009.

[7] J. Vanne, M. Viitanen, T. D. Hämäläinen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1885-1898.

[8] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1827-1838.

[9] Y. J. Ahn, T. J. Hwang, D. G. Sim, and W. J. Han, "Implementation of fast HEVC encoder based on SIMD and data-level parallelism," *EURASIP J. Image Video Process.*, vol. 16, Dec. 2014, pp. 1-19.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] A. Ylä-Outinen, A. Lemmetti, M. Viitanen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: HEVC/H.265 4K30p intra encoder," *in Proc. IEEE Int. Symp. Multimedia*, Taichung, Taiwan, Dec. 2017.

[15] T. K. Heng, W. Asano, T. Itoh, A. Tanizawa, J. Yamaguchi, T. Matsuo, and T. Kodama, "A highly parallelized H.265/HEVC real-time UHD software encoder," *in Proc. IEEE Int. Conf. Image Processing*, Paris, France, Oct. 2014.

[16] 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.

[17] *Kvazaar HEVC encoder* [Online]. Available: https://github.com/ultravideo/kvazaar

[18] *AirFrame data center solution* [Online]. Available: https://networks.nokia.com/solutions/airframe-data-center-solution

[19] *Catapult: Product Family Overview* [Online]. Available: http://calypto.com/en/products/catapult/overview

[20] 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.

[21] P. Sjövall, V. Viitamäki, J. Vanne, and T. D. Hämäläinen, "High-level synthesis implementation of HEVC 2-D DCT/DST on FPGA," *in Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, New Orleans, Louisiana, USA, Mar. 2017.

[22] V. Viitamäki, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "High-level synthesized 2-D IDCT/IDST implementation for HEVC codecs on FPGA," *in Proc. IEEE Int. Symp. Circuits Syst.*, Baltimore, Maryland, USA, May 2017.

[23] *Test Sequences* [Online]. Available: http://ultravideo.cs.tut.fi/#testsequences