Bolt65 – performance-optimized HEVC HW/SW suite for Just-in-Time video processing

Igor Piljić*, Leon Dragić*, Alen Duspara*, Mate Čobrnić*, Hrvoje Mlinarić*, Mario Kovač*
* Faculty of electrical engineering and computing, University of Zagreb, Zagreb, Croatia
{igor.piljic, leon.dragic, alen.duspara, mate.cobrnic, hrvoje.mlinaric, mario.kovac}@fer.hr

***Abstract* – This paper presents Bolt65 HEVC software/hardware suite, consisting of encoder, decoder and transcoder, that is being developed at Faculty of electrical engineering and computing, University of Zagreb. One of the primary focus of Bolt65 is achieving Just-in-Time requirements that would enable video encoding/transcoding on demand. To achieve this goal, Bolt65 tries to maximally utilize all software and hardware components of the system on different architectures, from homogeneous CPU architectures with vector extensions to heterogeneous, accelerator-based architectures that have different types of processing cores. CPU-only implementation of Bolt65 was compared with referent HM HEVC software in three different encoding configurations: All Intra (AI), Low delay (LD) and Random Access (RA). Results show that Bolt65 achieves significant speed-up in all configurations and bitrate savings in AI and RA modes while sacrificing quality in order to achieve just-in-time requirements.**

Keywords - HEVC;video codec;Just-in-Time;Bolt65; heterogeneous accelerator-based architectures

I. INTRODUCTION

Statistics show that Global video IP traffic will be 82 percent of all consumer Internet traffic by 2022, up from 75 percent in 2017. The annual global IP traffic will reach 4.8 ZB (ZB = 1000 Exabytes) by 2022, 3.9 ZB of which will be video content [1]. Handling this enormous amount of data is a very challenging task for the video content providers in years to come. Another factor that highlights this problem is the fact that a range of different devices that can play video content is constantly growing. With such diversity of devices, that have different decoding capabilities, computing resources, network bandwidths, and resolutions, a single copy of the encoded video cannot match requirements of all devices.

Current video providers tackle with this problem usually by pre-encoding and storing multiple copies of the same video on the server and sending the version that best satisfies the requirements of the user. Such an approach has very high storage costs and pre-encoded video streams may still not exactly match end-user requirements. Furthermore, emerging spatial resolutions (4K, 8K...), as well as the long-tail distribution of video content (90 percent of videos are viewed by only 10 percent of users and vice versa) make this concept hardly sustainable. Just-in-Time (JiT) video encoding/transcoding has one of the key roles in resolving these issues. Video transcoding refers to the problem of adapting on-the-fly Internet video content based on user's device features or specific operational conditions. Adaptation involves changing video properties, such as spatial, temporal and amplitude resolution, bitrate and video format. Instead of storing multiple copies on the server, only one version with the highest quality can be stored and transcoded on demand in real-time. Although JiT transcoding increases efficiency while providing the best possible QoE, it is extremely compute and data-intensive operation.

High efficiency video coding (HEVC/H.265) standard is novel video compression standard developed by Joint Collaborative Team on Video Coding (JCT-VC) as a joint activity of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations [2][3]. HEVC standard shows a significant advance in compression efficiency compared to its predecessors, such as Advanced Video Coding (AVC). At the same subjective quality, HEVC saves approximately 50 percent bitrate but increases computational complexity and resource requirements by up to 10 times [4].

II. BOLT65 HEVC CODEC

Bolt65 is a codename for an ongoing project on Faculty of Electrical Engineering and Computing in Zagreb. The aim of the project is to develop a "clean room" software/hardware suite consisting of an encoder, decoder, and transcoder. Special focus is set on the performance-efficiency achieved by low-level optimizations and hardware-software co-design by exploiting heterogeneous accelerator-based architectures. Another important focus of Bolt65 is the just-in-time requirement which sets constraints on processing time making Bolt65 suitable for encoding/transcoding on demand. The project is intended for research and development of current and future video processing algorithms and high-performance computing architectures. Bolt65 suite is written from scratch in C++ and is supported on Windows and Linux operating systems. In the following subsections, the main features of Bolt65 are presented.

A. Input and output

Bolt65 encoder supports input video data in YUV and Y4M format with 8-bit pixel samples and 4:2:0 chroma sampling, which corresponds with the Main profile defined in HEVC standard version 1. Prefetching frames from the input file has been implemented to avoid waiting for the frame data to be loaded after each cycle. The output of the encoder is HEVC bitstream of the Main profile.

Bolt65 decoder supports decoding HEVC bitstreams generated by Bolt65 encoder. In the current version, the decoder is used only in cascade with encoder forming a transcoder. In the future, the decoder will be in conformance with the Main profile defined in HEVC standard.

Bolt65 transcoder supports homogeneous transcoding of HEVC bitstreams. The input to the process is HEVC encoded video bitstream while the output is HEVC encoded video bitstream with modified video properties. Transcoder supports downscaling of the spatial, temporal and amplitude resolutions. The current version of the transcoder is implemented as cascaded pixel-domain transcoding architecture.

B. Block structures

The input video frame is divided into Coding Tree Units (CTU). Each CTU consists of three Coding three blocks (CTB), one luma block and two corresponding chroma blocks and associated syntax elements [5]. Bolt65 supports CTU sizes of NxN, where $N \in \{16,32,64\}$, meaning that luma CTB is NxN, while, due to 4:2:0 color subsampling, chroma CTBs are (N/2xN/2). CTUs can be divided into four smaller units called Coding Units (CU) following a quadtree structure, where CTU represent tree root. Bolt65 supports CU sizes of NxN, where $N_{min} \leq N \leq N_{max}$ and $N \in \{8,16,32\}$. N_{min} and N_{max} are defined in the configuration, if not, default values of $N_{min}=8$ $N_{max}=32$ are used.

Each leaf CU can act as a root for residual quadtree (RQT). The residual quadtree is a tree of transform units (TU) containing transform blocks (TB) that can be created to enable the adaptation of the transform functions to the varying space-frequency characteristics of the residual signal. In Bolt65 luma TB sizes are MxM, while chroma TB sizes are (M/2)x(M/2), where M=N and $M \in \{8,16,32\}$, N being the size of RQT root.

Leaf CUs can also be split to up to four prediction units (PU) that can be used for more precise motion estimation. HEVC standard supports 8 partitioning modes for splitting CU to PU: MxM, Mx(M/2), (M/2)xM, (M/2)x(M/2), Mx(M/4), Mx(3M/4), (M/4)xM and (3M/4)xM. Currently, in Bolt65, only MxM partitioning mode is used.

C. Intra and Inter prediction

Each of the leaf CUs can be partitioned in one or more prediction units (PU) that are predicted by taking advantage of spatial (Intra-prediction) [6] or temporal (Inter-prediction) [7] dependency of video frames. Prediction output represents a block of predicted samples which is then subtracted from the original block to form a residual (i.e. prediction errors) used as an input in following steps of the encoder. Splitting CU to smaller PUs can achieve better coding efficiency, but at the same time increases computational complexity.

Both prediction modes are supported by Bolt65. In intra prediction, all 35 modes, 33 angular and 2 non-directional (DC and planar) are evaluated. Several algorithms for determining distortion are available: Sum of absolute differences (SAD), Sum of absolute transform differences (SATD) and Mean square error (MSE). Determining which of these algorithms will be used as a block matching measurement will affect coding efficiency and computing complexity, which is an important aspect to consider with Just-in-Time requirement.

Motion estimation process in inter prediction is performed by searching the best candidate from the previously encoded frame. Search algorithms that are supported in Bolt65 are Three Step Search (TSS), Diamond Search and Full Search. In HEVC, fractional half- and quarter-pixel motions are possible, but they require additional interpolation filter. Since interpolation is a compute-intensive operation, each of the mentioned search algorithms has an option to exclude fractional motions from a search process.

D. Transformation and quantization

Residual blocks are transformed from spatial to the frequency domain using a discrete cosine transform (DCT) and then quantized with defined quantization parameter (QP). In the decoding loop, dequantization and inverse transformation are used to form a reconstructed frame.

Bolt65 supports HEVC standard integer transform for all transform block sizes of NxN, where $N \in \{4,8,16,32\}$. Since all functions (DCT, IDCT, quantization, and dequantization) are highly parallelizable, advanced vector extensions (AVX, AVX2) on x86 instruction set architectures for Intel processors are used for their optimization.

A high-throughput fully pipelined FPGA-based accelerator for Bolt65 DCT has been designed and implemented. The architecture consists of two cascaded 1D DCT cores with a constant throughput of 32 pixels per cycle in all size modes. The accelerator receives the data through a 512-bit bus which enables fetching 32 16-bit samples in a single clock. The pipeline for worst-case scenario consists of 75 stages which means that for processing a single 32x32 matrix, 106 cycles are necessary. The accelerator was integrated with Bolt65 software encoder/decoder/transcoder in MANGO project [8].

E. In-loop filters

The in-loop filters are applied in the encoding and decoding loops to reduce artifacts that can appear on block boundaries, increasing the quality of reconstructed pictures. There are two types of in-loop filters in HEVC: Deblocking filter [9] and Sample adaptive offset (SAO) [10]. The current version of Bolt65 encoder implements the deblocking filter, which can be enabled or disabled depending on the timing constraints set.

The implementation of SAO filtering is planned and will be available in future version of Bolt65.

F. *Parallelization*

HEVC standard introduces two novel parallelization concepts, Tile and Wavefront parallel processing, along with the slices, which were also available in previous AVC standard. Tiles, performance-wise, outperform other parallelization concepts in HEVC [11], so this concept was implemented in Bolt65. Tiles are rectangular-shaped groups of CTUs separated by vertical and horizontal boundaries, which can be processed independently and can be split uniformly or non-uniformly. Load balancing algorithms, such as TTLB or Tile-cost balancing algorithm, can be based on historical data, dynamically change tile boundaries to efficiently distribute the workload. In the case when there are more tiles than processing cores, several algorithms for the thread to core assignment which increase the performance of the encoder/transcoder are available.

G. *Entropy coding*

After quantization, all coefficients, along with other syntax elements are entropy coded using context-adaptive binary arithmetic coding (CABAC) [12]. Bolt65 performs entropy coding on a tile level, forming parts of the bitstream in parallel and then merges them to form a final bitstream.

H. *Configurations*

As defined in Common Test Conditions [13], three configurations of the encoder are available:

- Intra, main
- Random-Access, main
- Low-delay, main

For each video sequence, four quantization parameter values are to be used: 22, 27, 32 and 37. These values define the QP values used for the I-frames in a sequence (configuration files further define QP values used for other frames).

I. *External dynamic control*

Both Bolt65 encoder and transcoder execution can be controlled during runtime by an external process that can monitor the current state of the encoding/transcoding. Changing some of the parameters, such as quantization parameter, search algorithm or GOP structure can be used to increase or decrease coding efficiency, power consumption or performance, depending on the current requirements posed by an external factor of the system. Hence, different policies can be implemented and used on the application, without the need for modification of existing source code.

III. PERFORMANCE EVALUATION AND ANALYSIS

Characteristics of Bolt65 were evaluated and compared to the newest version of referent HM encoder [14]. Benchmark was conducted on 9 video sequences shown in Table I. Validation and verification of encoded bitstreams were performed using StreamEye software [15].

TABLE I. TEST VIDEO SEQUENCES

Video sequence	Class	Resolution	Number of frames (frame rate)
Shields	E	1280x720	504 (50fps)
ParkRun	E	1280x720	504 (50 fps)
KristenAndSara	E	1280x720	600 (60 fps)
BasketballDrive	B	1920x1080	500 (50fps)
BQTerrace	B	1920x1080	600 (60fps)
BlueSky	B	1920x1080	217 (25fps)
PedestrianArea	B	1920x1080	375 (25 fps)
RushHour	B	1920x1080	500 (25 fps)
Traffic	A	2560x1600	150 (30 fps)

All experiments were conducted on an environment shown in Table II, using one core of the processor.

TABLE II. TEST ENVIRONMENT

Processor	Intel i5 4570 (3.2 GHz)
Memory	32 GB
L1 cache	32 kB
L2 cache	256 kB
Compiler	Intel C++ 18.0
Operating system	64-bit Windows 10 Pro

All three encoder configurations, mentioned in Section I.A, are used based on HM default configuration files that are available with the software (*encoder_intra_main.cfg*, *encoder_lowdelay_main.cfg*, *encoder_randomaccess_main.cfg*). Some of the encoding parameters used on both encoders are shown in Table III.

TABLE III. ENCODER CONFIGURATION

Coding option	Parameter
QP	32, fixed
Search algorithm	Fast, search area 64
GOP	All Intra, Low delay, Random Access (every 32. frame is I frame)
In loop filters	Both, deblocking filter and SAO, disabled
CTU size	64x64
QP	32, fixed
Search algorithm	Fast, search area 64
GOP	All Intra, Low delay, Random Access (every 32. frame is I frame)
In loop filters	Both, deblocking filter and SAO, disabled

During the encoding process, three aspects were considered:

TABLE IV. COMPARISON OF HM REFERENCE ENCODER AND BOLT65

Video sequence	All Intra (AI)			Low delay (LD)			Random Access (RA)		
	Speedup	**PSNR**	**Bitrate**	**Speedup**	**PSNR**	**Bitrate**	**Speedup**	**PSNR**	**Bitrate**
Shields	42.34x	-10.69%	8.21%	148.99x	-13.61%	-90.92%	224.42x	-21.72%	63.82%
ParkRun	42.31x	-12.23%	20.40%	176.57x	-13.39%	-55.58%	235.12x	-18.50%	107.67%
KristenAndSara	41.76x	-10.30%	-13.15%	226.82x	-9.91%	-72.91%	234.53x	-17.69%	948.17%
BasketballDrive	41.02x	-5.88%	-0.76%	230.69x	-11.67%	-62.38%	236.78x	-14.52%	274.89%
BQTerrace	42.47x	-8.51%	0.30%	188.20x	-11.27%	-79.21%	268.39x	-15.33%	230.00%
BlueSky	41.01x	-10.22%	8.55%	180.13x	-11.15%	-86.07%	188.77x	-19.48%	70.20%
PedestrianArea	40.40x	-5.89%	9.78%	264.59x	-12.10%	-55.59%	233.00x	-15.50%	524.38%
RushHour	40.27x	-5.05%	9.17%	279.96x	-11.90%	-37.81%	231.28x	-13.17%	776.18%
Traffic	42.79x	-9.08%	10.16%	224.05x	-9.10%	-70.71%	234.17x	-17.63%	293.63%
AVERAGE	**41.6x**	**-8.6%**	**5.8%**	**213.3x**	**-11.5%**	**-67.9%**	**231.83x**	**-17.0%**	**365.4%**

TABLE V. COMPARISON OF HM REFERENCE ENCODER AND BOLT65 – DIFFERENCE IN PSNR (DB) AND BITRATE (BPS)

Video sequence	All Intra (AI)		Low delay (LD)		Random Access (RA)	
	PSNR (Δ dB)	**Bitrate (Δ bps)**	**PSNR (Δ dB)**	**Bitrate (Δ bps)**	**PSNR (Δ dB)**	**Bitrate (Δ bps)**
Shields	-3.30	218.35	-4.05	-2341.96	-6.46	1643.95
ParkRun	-3.50	1055.56	-3.76	-2490.48	-5.29	4710.81
KristenAndSara	-3.66	-147.80	-3.49	-192.70	-6.35	2779.10
BasketballDrive	-2.04	-14.10	-3.80	-1285.80	-4.92	5385.40
BQTerrace	-2.69	14.60	-3.46	-3249.80	-4.82	9039.90
BlueSky	-3.48	181.22	-3.69	-2043.09	-6.52	1634.22
PedestrianArea	-2.17	57.33	-4.18	-255.60	-5.61	2320.20
RushHour	-1.94	802.20	-4.24	-2773.00	-4.97	55520.00
Traffic	-3.06	367.00	-3.04	-1231.40	-5.98	5224.80
AVERAGE	**-2.87**	**281.57**	**-3.74**	**-1762.65**	**-5.66**	**9806.49**

- Speedup as a ratio between performance times
- Peak-Signal-To-Noise (PSNR) for quality assessment (observed for luma component)
- Bitrate for coding efficiency

Results for each of the configuration and for each video sequence are shown in Table IV and Table V. As can be seen from the table IV Bolt65 achieves significant speedup for all three encoder configurations, from 41.6x in All Intra to 231.84x in Random Access on average. In default RA configuration for HM encoder, the focus is set on accomplishing the highest video quality at the cost of bitrate and performance, which is why in RA mode Bolt65 encounters the highest loss in video quality and biggest gains in bitrate and performance time. On the contrary, LD configuration of HM encoder is set to achieve best coding efficiency, i.e. smallest bitrate. Hence, bitrate savings of HM encoder when compared to Bolt65 are highest in LD mode. Since HM encoder includes more encoding tools, such as symmetric and asymmetric prediction units, deeper transform trees, complex Rate Distortion Optimization algorithms, etc., video quality is still higher than for Bolt65 encoder. However, the implementation of these tools in the encoder scheme significantly increases performance time, which makes Just-in-Time encoding hardly feasible.

As can be observed from the presented results speedup gained by Bolt65 encoder comes at the cost of reduced video quality for all video sequences. This can be expected since Bolt65 is designed for fast encoding, aiming to satisfy Just-in-time requirements, where the focus is set on processing time.

IV. CONCLUSION

Bolt65 is a SW/HW suite, consisting of a video encoder, decoder and transcoder and supporting hardware architecture in the form of FPGA accelerators intended for just-in-time processing. The suite was developed at FER, the University of Zagreb as a “clean room” solution based on HEVC standard with special focus set on the performance-efficiency. It was achieved by low-level optimizations and hardware-software co-design by exploiting heterogeneous accelerator-based architectures. In the paper, we benchmark the encoder and use HM reference encoder 16.2 as a baseline for comparisons. The results show that our encoder achieves significant speed-up and bitrate savings as it sacrifices quality for meeting the just-in-time requirement. In low-delay configuration, HM reference encoder offers better coding efficiency due to basic RDO mode decision algorithm used in Bolt65.

The solution is currently in the early stages, so some advanced features are still missing, however, further development and introduction of new features is planned. More advanced Rate-Distortion Optimization (RDO) mode decision algorithm will be introduced in future versions of the encoder and transcoder which should improve results by better balancing between the coding efficiency, quality, and complexity of the encoder/transcoder. Since SW/HW codesign is an important focus of the research group, more hardware-based accelerator kernels for heterogeneous high-performance computers will be designed and integrated.

ACKNOWLEDGMENT

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 671668.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2017–2022," 26 November 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2]] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[3]] G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[4] F. Bossen, B. Bross, K. Sühring, and D. Flynn, “HEVC Complexity and Implementation Analysis,” IEEE Trans. Circuits Syst. Video Techn. vol. 22(12), pp. 1685-1696, 2012.

[5] I. Kim, J. Min, T. Lee, W. Han and J. Park, "Block Partitioning Structure in the HEVC Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1697-1706, Dec. 2012.

[6] J. Lainema, F. Bossen, W. Han, J. Min and K. Ugur, "Intra Coding of the HEVC Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1792-1801, Dec. 2012.

[7] K. Ugur, A. Alshin, E. Alshina, F. Bossen, HanW, J. Park, J. Lainema (2013) Motion compensated prediction and interpolation filter design in H.265/HEVC. IEEE J Sel Top Signal Process 7(6): 946–956

[8] Flich, Jose; Agosta, Giovanni; Ampletzer, Philipp; Atienza, David; Brandolese, Carlo; Cilardo, Alessandro; Fornaciari, William; Hoornenborg, Ynse; Kovač, Mario; Piljić, Igor; Duspara, Alen; Dragić, Leon; Knezović, Josip; Sruk, Vlado; Hofman, Daniel; Maitre, Bruno; Massari, Giuseppe; Mlinarić, Hrvoje; Papastefanakis, Ermis; Roudet, Fabrice; Tornero, Rafael; Zoni, Davide. "MANGO: Exploring Manycore Architectures for Next-GeneratiOn HPC Systems," 2017 Euromicro Conference on Digital System Design (DSD), Vienna, 2017, pp. 478-485

[9] A. Norkin, G. Bjøntegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou,G. Van der Auwera (2012) HEVC deblocking filter. IEEE Trans Circuits Syst Video Technol 22(11):1746–1754

[10] C.M. Fu, C.Y. Chen, Y.W. Huang, S. Lei (2011) Sample adaptive offset for HEVC. In: IEEE 13th international workshop on multimedia signal processing (MMSP) 2011

[11] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, “An overview of tiles in HEVC,” IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, pp. 969-977, Dec. 2013.

[12] C. Auyeung, J. Xu,G. Korodi , J. Zan,D. He,Y. Piao ,E. Alshina , J. Min, J. Park (2011) A combined proposal from JCTVC-G366, JCTVC-G657, and JCTVC-G768 on context reduction of significance map coding with CABAC, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-G1015, Geneva, Nov. 2011

[13] F. Bossen, “Common test conditions and software reference configurations,” Tech. Rep. JCTVC-H1100, 2012.

[14] Joint Collaborative Team on Video Coding Reference Software ver. HM 16.20, [online] Available: http://hevc.hhi.fraunhofer.de/.

[15] Elecard: StreamEye software. Available: https://www.elecard.com/products/video-analysis/streameye