OpenHardwareVC: An Open Source Library for 8K UHD Video Coding Hardware Implementation

Wei Gao*
gaowei262@pku.edu.cn
Peking University Shenzhen Graduate School
Peng Cheng Laboratory
China

Hang Yuan
Yang Guo
Lvfang Tao
Zhanyuan Cai
{hyuan,ltao,caizhanyuan}@pku.edu.cn
guoyang@stu.pku.edu.cn
Peking University Shenzhen Graduate School
Peng Cheng Laboratory
China

Ge Li
geli@pku.edu.cn
Peking University Shenzhen Graduate School
China

ABSTRACT

The hardware-accelerated real-time compression of 8K Ultra-High-Definition (UHD) video is an exemplary application that empowered by the latest video coding standard. However, the coding tools added to the recently released third-generation audio video coding standard (AVS3) greatly increase the coding complexity, which seriously hinders the efficient implementation of hardware encoder. In order to break the known bottleneck, this paper presents the first open source software library for 8K UHD video coding hardware implementation, namely OpenHardwareVC. Specifically, based on the analysis of the original AVS3 software algorithm, we provide the hardware acceleration designs of the four major coding stages, including coding unit (CU) partition, intra prediction, transform and entropy coding, in this library. Simulation results on Xilinx VU440 FPGA show that the real-time compression of 8K UHD videos at 30 frames per second (fps) can be easily supported based on software-described modules packaged in this library. The release of this library is quite favorable for the hardware design and system implementation of UHD video coding, which is also beneficial to the promotion of the new coding standard. The open source library for OpenHardwareVC is available at https://git.openi.org.cn/OpenHardwareVC.

CCS CONCEPTS

• **Computing methodologies → Image compression**; • **Software and its engineering → Open source model**.

KEYWORDS

video coding, real-time coding, 8K UHD, open source software, hardware implementation

*Corresponding author.

ACM Reference Format:
Wei Gao, Hang Yuan, Yang Guo, Lvfang Tao, Zhanyuan Cai, and Ge Li. 2022. OpenHardwareVC: An Open Source Library for 8K UHD Video Coding Hardware Implementation. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3503161.3548543

1 INTRODUCTION

Because of the ultimate pursuit of humans in visual experience, applications of 8K (7680×4320) Ultra-High-Definition (UHD) [12] video have become popular with the improvement of communication and video compression technologies in recent years. However, the current commonly used wired or wireless networks can not support the real-time transmission of raw 8K video. Thus, the real-time compression of 8K videos still is a great challenge, especially on lightweight hardware platforms. For example, in the current 8K live-broadcasting system, the camera and real-time coding server must be connected by a high-bandwidth cable, which seriously limits the flexibility of scene capture and further affects the visual experience of viewers. Besides, other wireless devices such as drones and satellites also have strong demand for highly efficient hardware implementation of 8K real-time compression. Considering the advantages of all-intra (AI) coding configuration in coding latency, computational complexity and reconstruction quality compared to other modes, the hardware integration of 8K real-time intra coding is highly demanded.

The third-generation audio video coding standard (AVS3) was recently released by China AVS working group [19]. Compared with AVS2, the previous generation coding standard, many introduced coding tools have remarkably improved the encoding performance. Unfortunately, these coding tools make each encoding stage more complex, and the overall computational complexity also increases dramatically, which further hinders the hardware implementation of real-time encoders. Actually, implementing the codec on the hardware platform is an essential part of coding standard promotion. There are various works focusing on this topic in the previous coding standards [4, 8–10, 14]. Nevertheless, the researches on AVS3 hardware optimization are still relatively lacking, especially open source works. Hence, it is extremely urgent to explore the hardware-friendly optimization for the major modules in AVS3 AI coding.

In order to break the above bottleneck, we develop a hardware implementation-oriented open source library for the major coding stages based on HPM-6.0, which is the reference software of AVS3. The Verilog-based code is released on the website [1]. The developed library has following contributions: (1) We have released OpenHardwareVC, the first hardware implementation open source library to achieve the real-time hardware encoding of 8K UHD videos with AVS3 AI configuration. (2) The hardware optimization problems of software algorithms in the four key coding stages, including Coding Unit (CU) partition, intra prediction, transform and entropy coding, have been solved. Moreover, to the best of our knowledge, each optimized module has achieved state-of-the-art performance. (3) Simulations show that all the modules in the library can support the real-time AI coding for 8K UHD videos at 30 frames per second (fps).

2 IMPLEMENTATION OF KEY STAGES FOR VIDEO CODING

2.1 CU Partition

CU partition is the first major module in video coding. During this process, an original frame is divided into small Coding Units (CU). Actually, more refined partition is more beneficial to capture coding gains [17]. Thus, the split rule in AVS3 has been greatly expanded compared with the previous generation standards, where the split modes have reached 6, i.e., Non-Split, Quad Tree (QT), vertical Binary tree (BT), horizontal BT, vertical Extend Quad tree (EQT) and horizontal EQT [6, 19]. The CU size range has been improved from { 8×8 to 64×64 } to { 4×4 to 128×128 }. However, the complex split rule becomes the bottleneck for 8K UHD video real-time compression on hardware platform. Besides, the recursively Rate-Distortion Optimization (RDO) decision strategy and large-scale data in 8K videos also are the main hindrances [13].

By analyzing the actual coding performance of coding rules in 8K video compression, we find that coding gains of smaller CU size and EQT modes are pretty limited. Hence, the smallest CU size is adjusted to 16×16, and the EQT modes are simplified from the allowed split modes for the lightweight optimization. In addition, a One-Pass CU partitioning method is developed to break the original recursive decision based on the hardware-friendly decision tree model, which can directly determine the split mode according to the calculated gradient (GRA) and histogram (HIST) features for luma and chroma components, respectively. Moreover, based on the Map-Reduce model, we propose a novel feature calculation method to deal with the severe dataflow latency caused by the large-scale 8K video data [5]. Specifically, the GRA and HIST feature computations are split into the GRA and HIST calculation of each SCUs for parallel computing. Afterward, the original GRA and HIST can be combined from the calculated values of SCUs. Figure 1 gives the simplified framework of the proposed CU partitioning algorithm. Due to all 128×128 size CUs will be split into 64×64 in AVS3 intra coding, the framework are designed based on 64×64 size CU. Note that the GRAs and HISTs of the 16 SCUs will be firstly calculated (CAL) and stored in the corresponding buffer. For further acceleration, the features calculation of the partitioned CUs within the current

[1]https://git.openi.org.cn/OpenHardwareVC

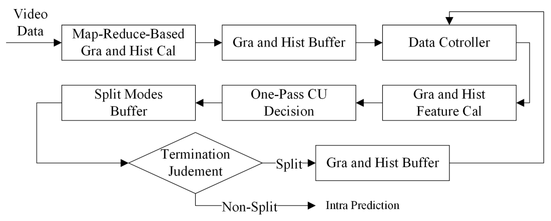

Figure 1: Simplified Framework of the Proposed CU Partitioning Algorithm.

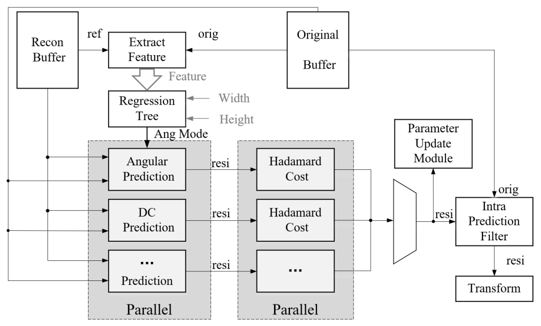

Figure 2: Hardware-Friendly Optimization Approach for Intra Prediction.

64×64 size CU can reuse the computed data to reduce redundant calculations.

Configuration examples of the CU partition module are provided as follows, where "clk", "rst", "$start$", and "$done$" are signals for instance control. "$addr$", "ce" and "we" indicate wires connected to the memory. "pf_mode_cnt" and "$pf\_split\_mode\_array$" denote the usage count and specific information of the split modes. "pf_width", "pf_height" and "pf_bet_depth" are the width, height and depth of the current CU. "$pf\_data\_buffer\_u0\_address0$" is the buffer of the pixel values in current CU.

- *enc_main_CU_split enc_main_CU_split_fu(.ap_clk(clk), .ap_rst(rst), .ap_start(start), .ap_done(done), .ap_idle(idle), .ap_ready(ready), .pf_mode_cnt(cnt), .pf_split_mode_array(array), .pf_split_mode_array_address0(add), .pf_split_mode_array_ce0(ce) , .pf_split_mode_array_we0 (we), .pf_width(width), .pf_height(height), .pf_bet_depth(depth), .pf_data_buffer_u0_address0(add) ,· · ·);*

2.2 Intra Prediction

To achieve the real-time intra prediction task in AVS3, three optimization methods for hardware acceleration of intra prediction are proposed, namely not split prediction unit (PU), mode bits estimation and modified intra mode decision. The details of these algorithms are given in [3].

The data scheduling of proposed intra prediction hardware optimization is shown in Figure 2. First, Original Buffer and Recon Buffer are used to store the original and reference samples of the coding CU, respectively. Four difference values between them are calculated in Extract Feature module. Afterward, angular modes except vertical and horizontal are predicted in the Regression Tree module. Although the coding quality can be improved by configuring more angular modes, the increasing hardware resources also should be considered. As shown in Figure 2, the intra prediction modules of all candidate modes are performed in parallel. Similarly, their hadamard costs for residual samples are also parallelly calculated in Hadamard Cost module [20], where the mode with minimal cost will be selected as the optimal option. Moreover, the processed pixels after Intra Prediction Filter module are stored into Recon Buffer for the subsequent CU prediction. Note that the processing of chroma blocks are performed after luma blocks. The filtered residual samples are transformed to coefficients in subsequent modules. Based on the above scheduling strategy, the different blocks can share the resources without increasing the delay of the overall module.

In the Verilog code, the instantiation of intra prediction main module is as follows, where "org_addr0" and "org_addr1" are wires for Original Buffer. "$mpm0$" and "$mpm1$" represent the most possible mode (MPM) list. "$stride$" is the stride of current coding frame and usually is the value of frame width.

- *enc_intrapred enc_intrapred_fu(.ap_clk(clk), .ap_rst(rst), .ap_start(start), .ap_done(done), .ap_idle(idle), .ap_ready(ready), .org_address0(org_addr0), .org_address1(org_addr1), .cu_w(cu_width), .cu_h(cu_height), .stride_org(stride), · · · , .mpm0(mpm0), .mpm1(mpm1), .ap_return(return));*

2.3 Transform

Transform is also an important parts in video coding, which can effectively eliminate the information redundancy. It is well known that transform is computationally intensive, so it is necessary to realize hardware acceleration, especially for 8K UHD sequences. In AVS3 [19], there are two kinds of two-dimensional (2D) discrete cosine transform (DCT) [1] including type II (DCT-II) and type VII (DST-VII) in all intra structure [18]. In the proposed approach, we prune and optimize the transform algorithm to reduce computational complexity. Specifically, only DCT-II is retained after exploring the tradeoff among computational complexity, R-D performance and hardware resource consumption. Besides, three types of high-throughput hardware acceleration strategies are applied to further speed up the computation, including calculation pipelining, logical-loop unrolling and module-level parallelism.

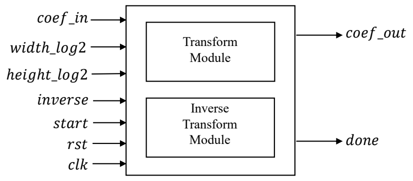

Figure 3: Proposed schematic architecture of DCT/IDCT module.

Figure 3 is the proposed schematic architecture of DCT/IDCT module [7]. "clk", "rst", and "$start$" are used as input enable signals, and "$done$" as the output enable signal. We define two input data "$width_{log2}$" and "$height_{log2}$" to specify the size of the input transform block. In order to save the data width, we use a logarithmic definition. In the calculation process of transform block, we traverse each row of the block, process one row each time, and output one column of the result. Since the input and output single data are 16 bits, the data bit width of the input and output data "$coef_{in}$" and "$coef_{out}$" is 1024. We designed a control bit named "$inverse$" to decide whether to transform or inverse transform. The top-level functions of the transform module are "enc_trans" (for DCT) and "com_itrans" (for IDCT), respectively. The example of instantiation for DCT module is as follows:

- *enc_trans enc_trans_fu(.ap_clk(clk), .ap_rst(rst), .ap_start(start), .ap_done(done), .cu_width_log2($width_{log2}$), .cu_height_log2($height_{log2}$), .coef_i(coef_in), .coef_o(coef_out),· · ·);*

2.4 Entropy Coding

Entropy coding is the process where coding information is organized as a binary sequence, and the binary arithmetic coding algorithm is applied to further compress the data based on entropy modeling. The entropy encoder library is packaged with IP implementation for the AVS3 entropy coding algorithm, mainly optimized for high throughput and high efficiency. As depicted in Figure 4. The whole module is composed of a parallelizable binarizer submodule (BinEnc) for the arrangement and binarization of transform coefficients, and an entropy coding engine sub-module (BitEnc) designed to compress the resulting binary string with arithmetic coding capabilities. Due to the bin-level data dependency caused by the nature of entropy modeling, the BitEnc sub-module is designed and described as a non-stop coding pipeline for incoming bins, buffered for steady supply. For architecture details, please refer to our individual paper that focuses on the design of the entropy encoder module [15]. After synthesis, implementation, and simulation, the evaluated throughput on Xilinx FPGA reaches 298 Mbin/s under the given constraints for hardware resources. The details of hardware utilization are reported in Section 3.

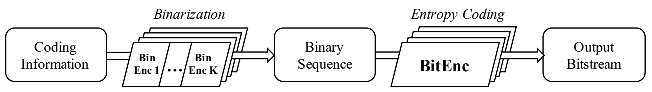

Figure 4: The components of the entropy encoder library.

For custom deployment and integration of the proposed entropy encoder module, hierarchical parallelismcan be applied to optimize the trade-off between throughput and hardware efficiency. For parallelism at intra sub-module level, multiple BinEnc sub-modules that designed to handle vairous syntax elements can be instantiated for simultaneous binarization, to match the peak bin-processing rate of the single-way BitEnc sub-module. The interaction between BinEnc(s) and BitEnc is considered at the inter sub-module level. Finally, to further improve throughput, patch-level parallelism is an option to support parallel processing of sub-pictures.

Configuration examples of the entropy encoder library are provided as follows. For the binarizer, "$addr$", "ce" and "q" stand for wires connected to the memory for the quantized coefficients. "log_w" and "log_h" describe the size of the block. "n_sig" counts the number of non-zero coefficients, while "$type$" marks the property of luma / chroma component. For the entropy coding engine,

Table 1: Coding Efficiency and FPGA Resource Utilization. The Proportions of Resource Utilization on VU-440 FPGA are Also Given.

Item	CU Partition	Intra Prediction	Transform	Entropy Coding
BRAM_18K	1 (0.0%)	320 (6.3%)	84 (1.7%)	52 (1.0%)
DSP48E	110 (3.8%)	2388 (82.9%)	352 (12.2%)	0 (0.0%)
FF	252957 (5.0%)	111888 (2.2%)	130324 (2.6%)	10456 (0.2%)
LUT	588010 (23.2%)	432136 (17.1%)	194012 (7.7%)	20860 (0.8%)
BD-PSNR	-0.15dB	-0.13dB	-0.02dB	/
Frame Rate	30fps @ 8K UHD			

wires tagged with "*bin_∗*", "*ctx_∗*", and "*bst_∗*" represent the wires connected to the queue for uncompressed binary sequence, to the buffer for context modeling, and to the queue for output bitstream.

- Instantiation of the BinEnc for coefficients binarization: *enc_eco_run_length_cc enc_eco_run_length_cc_fu(.ap_clk(clk), .ap_rst(rst), .ap_start(start), .ap_done(done), .coef_address0(addr),.coef_ce0(ce), .coef_q0(q), .log2_w(log_w), .log2_h(log_h), .num_sig(n_sig), .ch_type(type) ,· · ·);*
- Instantiation of the BitEnc for bitstream generation: *eco_engine_process eco_engine_process_fu(.ap_clk(clk), .ap_rst(rst), .ap_start(start), .ap_done(done), .queue_V_bin_dout(bin_dout), .sbac_ctx_run_ce0(ctx_run_ce),bst_new_V_val_din(bst_val), · · ·);*

3 EVALUATION

To verify the specific performance of the developed modules in our released library, we simulated them on Xilinx VU-440 FPGA at 200 MHz [16], where the results, including hardware resource utilization and supported frame rate, are shown in Table 1. In addition, the coding performance of each module is also illustrated in Table 1. All the compressing experiments followed the common test conditions (CTCs) of AVS3, and the coding configuration is all-intra. Since 8K UHD video is not included in CTC, the test experiments selected the sequences in PP8KLite (Peking University and Peng Cheng Laboratory 8K Dataset - Lite Version) [11]. Except that the entropy coding module is lossless, the coding loss of other modules compared to the original encoder HPM-6.0 is measured by BD-PSNR [2]. Obviously, although the resource utilizations of different modules are quite diverse, each module can effectively realize real-time encoding of 8K UHD videos at 30 fps. Moreover, considering the importance of these modules for UHD video coding hardware implementation, the encoding losses generated by the hardware-friendly optimizations are negligible.

4 CONCLUSION

In response to the dilemma of real-time encoding implementation of 8K UHD videos on hardware platform, this paper released the first software open source library for hardware implementation, namely OpenHardwareVC. The library mainly solves the hardware acceleration problem of four major stages, i.e., CU partition, intra prediction, transform and entropy coding, in AVS3 AI coding configuration. Simulation results on FPGA prove that the designed modules are sufficient for real-time hardware encoding of 8K videos at 30 fps. We believe that the release of this library can strongly support the hardware implementation and chip design of 8K UHD video coding, and effectively promote the application of AVS3 standard.

5 ACKNOWLEDGMENTS

This work was supported by The Major Key Project of PCL, National Key R&D Program of China (2020AAA0103501), Natural Science Foundation of China (61801303, 62031013), Guangdong Basic and Applied Basic Research Foundation (2019A1515012031), Shenzhen Fundamental Research Program (GXWD20201231165807007-20200806163656003), and Shenzhen Science and Technology Plan Basic Research Project (JCYJ20190808161805519).

REFERENCES

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE transactions on Computers* 100, 1 (1974), 90–93.

[2] G. Bjøntegaard. VCEG, Austin, TX, USA, Doc. VCEG-M33, 2018. Calculation of average PSNR differences between RD-curves. (VCEG, Austin, TX, USA, Doc. VCEG-M33, 2018).

[3] Zhanyuan Cai and Wei Gao. 2021. Efficient Fast Algorithm and Parallel Hardware Architecture for Intra Prediction of AVS3. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[4] Young-Kyu Choi, Hyuk-Jae Lee, and Soo-Ik Chae. 2021. High Throughput CBAC Hardware Encoder With Bin Merging for AVS 2.0 Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 11 (2021), 4439–4453. https://doi.org/10.1109/TCSVT.2020.3047925

[5] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. https://doi.org/10.1145/1327452.1327492

[6] Wen Gao and Siwei Ma. 2014. An overview of AVS2 standard. *Advanced Video Coding Systems* 22 (Jan. 2014), 35–49.

[7] Yang Guo, Wei Gao, Siwei Ma, and Ge Li. 2022. Accelerating Transform Algorithm Implementation for Efficient Intra Coding of 8K UHD Videos. *ACM Trans. Multimedia Comput. Commun. Appl.* 18, 4, Article 113 (mar 2022), 20 pages. https://doi.org/10.1145/3507970

[8] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang. 2016. CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network. *IEEE Transactions on Image Processing* 25, 11 (2016), 5088–5103. https://doi.org/10.1109/TIP.2016.2601264

[9] Grzegorz Pastuszak. 2019. Generative multi-symbol architecture of the binary arithmetic coder for UHDTV video encoders. *IEEE Transactions on Circuits and Systems I: Regular Papers* 67, 3 (2019), 891–902.

[10] Grzegorz Pastuszak. 2020. Multisymbol architecture of the entropy coder for H.265/HEVC video encoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 12 (2020), 2573–2583.

[11] PP8KLite 2021. *Peking University and Peng Cheng Laboratory 8K Dataset - Lite Version.* Retrieved Jan, 2022 from https://git.openi.org.cn/gaowei262/PP8KLite

[12] Masayuki Sugawara, Seo-Young Choi, and David Wood. 2014. Ultra-High-Definition Television (Rec. ITU-R BT.2020): A Generational Leap in the Evolution of Television [Standards in a Nutshell]. *IEEE Signal Processing Magazine* 31, 3 (2014), 170–174. https://doi.org/10.1109/MSP.2014.2302331

[13] G.J. Sullivan and T. Wiegand. 1998. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine* 15, 6 (1998), 74–90. https://doi.org/10.1109/79.733497

[14] Heming Sun, Dajiang Zhou, Landan Hu, Shinji Kimura, and Satoshi Goto. 2017. Fast Algorithm and VLSI Architecture of Rate Distortion Optimization in H.265/HEVC. *IEEE Transactions on Multimedia* 19, 11 (2017), 2375–2390. https://doi.org/10.1109/TMM.2017.2700629

[15] Lvfang Tao and Wei Gao. 2022. Low Power Entropy Coding Hardware Design for H.264/AVC Baseline Profile Encoder. In *IEEE International Conference on Multimedia and Expo (in press).*

[16] Vivado 2019. *UltraScale FPGA Product Tables and Product Selection Guide.* Retrieved Aug, 2021 from https://www.xilinx.com/support/documentation/selection-guides/ultrascale-fpga-product-selection-guide.pdf

[17] Meng Wang, Junru Li, Li Zhang, Kai Zhang, Hongbin Liu, Shiqi Wang, Sam Kwong, and Siwei Ma. 2020. Extended Coding Unit Partitioning for Future Video Coding. *IEEE Transactions on Image Processing* 29 (2020), 2931–2946. https://doi.org/10.1109/TIP.2019.2955238

[18] Audio Video Coding Standard Workgroup. 2019. AVS Proposal M4772: Implicit selection of transforms for intra coding. ftp://47.93.196.121/Public/avsdoc/1906_Chengdu/contrib/M4772.zip. Accessed in May 2022.

[19] Jiaqi Zhang, Chuanmin Jia, Meng Lei, Shanshe Wang, Siwei Ma, and Wen Gao. 2019. Recent development of AVS video coding standard: AVS3. In *Picture Coding Symposium (PCS)*. IEEE, 1–5.

[20] Jia Zhu, Zhenyu Liu, Dongsheng Wang, Qingrui Han, and Yang Song. 2013. Fast prediction mode decision with hadamard transform based rate-distortion cost estimation for HEVC intra coding. In *2013 IEEE International Conference on Image Processing*. 1977–1981. https://doi.org/10.1109/ICIP.2013.6738407