OPENCL PARALLELIZATION OF THE HEVC DE-QUANTIZATION AND INVERSE TRANSFORM FOR HETEROGENEOUS PLATFORMS

Diego F. de Souza, Nuno Roma, Leonel Sousa

INESC-ID, IST, Universidade de Lisboa
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
difs@sips.inesc-id.pt, nuno.roma@inesc-id.pt, leonel.sousa@inesc-id.pt

ABSTRACT

To tackle the growing demand for high efficient implementations of video decoders in a vast set of heterogeneous platforms, a high performance implementation of the HEVC de-quantization and inverse Discrete Cosine Transform (IDCT) modules is proposed. To efficiently take advantage of the several different GPU architectures that are currently available on these platforms, the proposed modules consist on unified OpenCL implementations, allowing their migration and acceleration in any of the available devices of current heterogeneous platforms. To achieve such objective, the memory accesses were highly optimized and no synchronization points were required, in order to attain the maximum performance. The presented experimental results evaluated the proposed implementation in three different GPUs, achieving processing times as low as 6.39 ms and 6.51 ms for Ultra HD 4K I-type and B-type frames, respectively, corresponding to speedup factors as high as 18.9× and 16.5× over the HEVC Test Model (HM) version 11.0.

Index Terms— Video coding, HEVC, de-quantization, transform coefficient decoding, Graphics Processing Unit (GPU), parallel processing

1. INTRODUCTION

In the past few years, the *High Efficiency Video Coding* (HEVC) standard established as the new state of the art on video compression. When compared with the previous standards, it has been shown that HEVC encoders can achieve equivalent subjective visual quality as H.264/AVC encoders, when using approximately 50% less of the bit rate. However, such coding efficiency comes at cost of a substantial increase of the computational complexity of both the video encoder and the decoder.

One of the introduced improvements by HEVC refers to the new quantization and transform modules. Up to 10.1% of the bit rate is saved when larger transform sizes (16×16 and 32×32) are used on top of the smaller transform sizes (4×4 and 8×8), in order to better reduce the spatial pixel correlation [1]. However, the usage of larger core transforms also significantly increases the HEVC encoder and decoder complexity. This is particularly relevant when the implementation of video decoders is restricted by performance constrained terminals, as it is the case of mobile and portable devices.

On the other hand, the recently proposed Open Computing Language (OpenCL) [2] framework has arisen as a highly viable alternative to exploit task-parallel and data-parallel compute capability across multiple heterogeneous devices. As a result, OpenCL has been gradually supported by several CPU – Intel, AMD, ARM, etc. – and GPU manufacturers – AMD/ATI, NVIDIA, Intel, ARM (Mali), Imagination Technologies (PowerVR), etc. – allowing the realization of unified program implementations that can be executed in any of the available CPU and GPU devices, which offers a highly convenient way to migrate and accelerate the most computational demanding parts of the program.

In this paper, it is presented a highly optimized parallel implementation of the HEVC de-quantization and inverse transform modules based on an extensive exploitation of the parallel processing power offered by modern GPUs. By making use of an unified programming based on OpenCL, the obtained experimental results demonstrated the ability to achieve processing times as low as 6.39 ms and 6.51 ms for Ultra HD 4K I-type and B-type frames, respectively, corresponding to speedup factors as high as 18.9× and 16.5× when compared to the CPU reference software.

Such an efficient and highly flexible parallelization is regarded as an important complementary step towards its integration with other video decoding modules that have already been parallelized [3], envisaging the possibility to full parallelize the HEVC decoder in the GPU, capable of handling real time HEVC decoding for Ultra HD 4K video sequences.

The remaining of this paper is organized as follows. Section 2 revises the last proposed algorithms for IDCT parallel implementation and the HEVC transform coefficient decoding. In Section 3, the HEVC de-quantization process and inverse transform is briefly revised. The proposed algorithm and consequent parallel implementation is presented in Sec-

This work was supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under projects SFRH/BD/76285/2011, PTDC/EEI-ELC/3152/2012 and PEst-OE/EEI/LA0021/2013.

tion 4. The experimental results and the addressed conclusions are shown in Sections 5 and 6, respectively.

2. BACKGROUND WORK

Along the past years, several video encoding and decoding modules have been implemented in GPU devices, such as the efficient motion estimation implementations on CPU + GPU platforms that were proposed by Xiao et al. [4] and Momcilovic et al. [5], for HEVC and H.264/AVC respectively.

Regarding the DCT, an OpenCL implementation of the real (non-integer) DCT for image compression was already proposed in [6], by using a floating-point representation. Nevertheless, not only is such non-integer transform not compliant with the most recent video standards, but the strict temporal requirements that are imposed in video coding are significantly more demanding than in image processing. In what concerns the transform module for video, several algorithms were already proposed to alleviate the complexity of the encoder side, like the zero block detection [7], which was based in [8] to eliminate redundant computations. In the decoder side, parallel implementations often pose difficult challenges, not only because the decoder should be able to support bitstreams produced by any encoder configuration, but also because the processing platform at the decoding device often imposes highly restrictive processing capabilities.

As an example, Yong et al. [9] exploited the NEON Single Instruction Multiple Data (SIMD) instruction set extension of the ARM platform to accelerate the transform and inverse transform modules, to be applied in mobile and tablet devices, achieving a speed up of 5.6 over the reference software for Full HD sequences.

Hardware implementation of the HEVC IDCT was also proposed in [10], where the proposed architecture can decode Ultra HD 4K video sequences at 30 frames per second with an operating frequency of 90 MHz. However, such implementations represent different compromises in terms of energy efficiency, resources utilization and programmability, preventing a fair comparison with high-performance computing platforms, like GPUs.

3. HEVC RESIDUAL PROCESSING MODULE

According to the HEVC standard, each frame is divided in Coding Three Units (CTU) of 64×64, 32×32 or 16×16 samples. Each CTU can be further partitioned in smaller blocks, denoted by Coding Units (CU), according to a quadtree structure. Each CU is divided in the Prediction Unit (PU) and the Transform Unit (TU). The TU is further split in smaller blocks, named Transform Blocks (TB), which are applied to core transforms of sizes 4×4, 8×8, 16×16 and 32×32.

After the entropy decoding, the residual data of different TUs can be processed in parallel. Each TU is composed by 3 TBs, one for the luma and two for the chroma component.

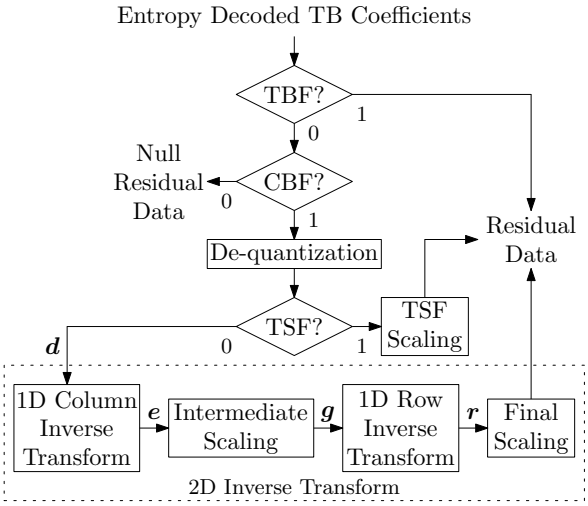

Fig. 1. Processing flow of the HEVC residual data decoding.

When the usual 4:2:0 chroma subsampling is adopted, the chroma TB is four times smaller than the corresponding luma TB, although they have the same size for TB of 4×4 pixels.

Similarly to the H.264/AVC, only integer core transforms are specified by the HEVC standard, to avoid the introduction of rounding drifts, both at the encoder and the decoder, caused by different rounding and floating point representations. As a consequence, the HEVC 4×4 to 32×32 transform kernels are based on the integer DCT [1]. The inverse DCT kernels are the same for luma and chroma TBs, except for the 4×4 luma TB of intra blocks, where an integer Inverse Discrete Sine Transform (IDST) is applied.

For each TB, the overall procedure is controlled by three flags: the Transquant Bypass Flag (TBF), the Coded Block Flag (CBF) and the Transform Skip Flag (TSF). When the TBF is set, the de-quantization and the inverse transform are bypassed, which means that the residual data already corresponds to the decoded coefficients. The CBF indicates when there is some residual data in the TB, while the TSF signs the decoder to skip the 2D inverse transform module. In this case, the residual data is directly obtained after the de-quantization.

Figure 1 depicts the processing of the HEVC residual data, corresponding to the coefficients that are obtained at the output of the entropy decoder. The de-quantization module implements the HEVC inverse scaling, which depends of the quantization parameter (QP) and on the adopted TB size. These scaling factors ensure the preservation of the normalization property in the transform domain. The 2D inverse transform is specified according to the usual separable 1D column and row decompositions, where the core transform is chosen according to TB size and prediction mode: *i*) each column of the de-quantized transform coefficient block (column vector $\boldsymbol{d}$) is firstly transformed into a row vector $\boldsymbol{e} = \boldsymbol{d}^T C$ (C denotes the core transform matrix); *ii*) each sample is conveniently scaled by a function $\boldsymbol{g} = f(\boldsymbol{e})$; *iii*) each scaled

intermediate sample row (row vector $\boldsymbol{g}$) is transformed into the column vector $\boldsymbol{r} = C^T \boldsymbol{g}^T$, which is finally scaled to obtain the residual data.

4. PARALLEL GPU IMPLEMENTATION

To ensure the maximum efficiency of the proposed parallelization on the GPU, the residual data obtained at the entropy decoder is appended with a control word by the CPU. For such purpose, convenient control information is encoded by using single-byte packets for each 4×4 block, which reduces the required memory transfers to the GPU. Such control information, with the respective bit positions, is described in Table 1.

Bits 0 and 1 are used as a binary code to represent the used luma TB size: 4×4, 8×8, 16×16 or 32×32. Since the coefficients from all TBs shall be sent to the GPU, the flags TBF and CBF can be merged into one (TBF = 1 or CBF = 0). When this composite flag is set, the TB coefficients will be either the residual data or a null block and the whole process is bypassed. Due to the existing dependency between the three flags presented in Figure 1 and by merging the flags TBF and CBF, there are only three possible flag combinations for each TB, meaning 27 flag combinations for luma and chroma TBs. This information can be stored in a five bits binary code (bits 2 to 6), which can be easily encoded in the CPU by using a lookup table and decoded in the GPU with bitwise operations, to reduce memory requests. The most significant bit is used to select the integer IDST or IDCT core transform, used only for the 4×4 luma TB.

Figure 2 represents the processing flow of the HEVC de-quantization and 2D inverse transform for a single 4×4 TB. The same procedure is straightforwardly extended when others TB sizes are processed. In the de-quantization step, the OpenCL work-items (WI_x) start by performing the de-quantization scaling and store the resulting transposed block in the local memory, to be used in the next steps. According to the OpenCL specification, this local memory can be shared

Bit	Data information
7	Prediction mode: inter or intra
6 to 2	Binary code to encode the three flags
1 and 0	Binary code to encode the four possible TB sizes

Table 1. HEVC IDCT information and respective bit position.

by all work-items in that work-group [2]. In the particular case of the GPU, this local memory is on-chip and has higher bandwidth and lower latency than private or global memory. If the TSF is set, the block will be immediately stored in the global memory, as residual data.

For each line of the de-quantized transposed block ($\boldsymbol{d}$) each work-item is responsible for: *i*) the dot product with one single column of the core transform; and *ii*) the subsequent intermediate scaling. Since each work-item is responsible for the same column j of the core transform matrix in both 1D inverse transforms, the assigned vector $\boldsymbol{c}_j = [c_{0j}, c_{1j}, \ldots, c_{Nj}]^T$ is copied from the constant memory to the private memory region of the work-item, which has a significant lower latency than the constant memory. The resulting vector is stored back in the local memory as a column vector, to be later handled in the next 1D inverse transform.

Upon the computation of the 1D row inverse transform and of the final scaling step, one pixel-domain residual block line is obtained, which is stored in the OpenCL global memory. Since the resulting data is aligned in raster scan order, only one memory instruction is needed to implement this step. For larger TB sizes, only half of the work-items in the work-group are needed to process the chroma TBs.

To avoid any divergence among work-items, four independent OpenCL kernels are launched in parallel to process the residual data, one for each TB size. At this respect, it is worth noticing that the whole algorithm was implemented without the need for any synchronization point, which could induce a reduction of the overall performance. In the Figure 3, it is shown an execution flow example with two OpenCL command queues (CQ), where K_N is the OpenCL kernel for

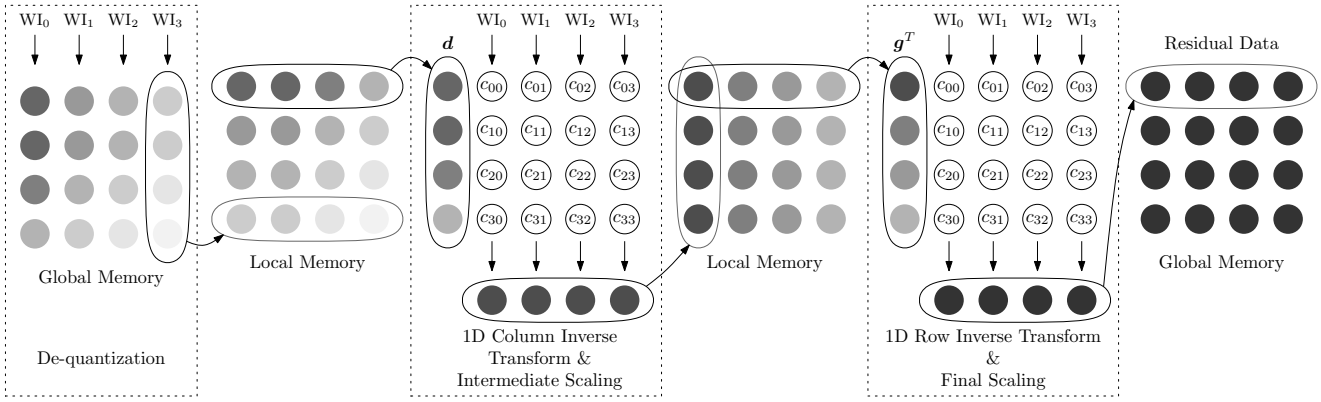

Fig. 2. De-quantization and 2D inverse transform implementation in the GPU for one 4×4 TB processed by 4 work-items (WI).

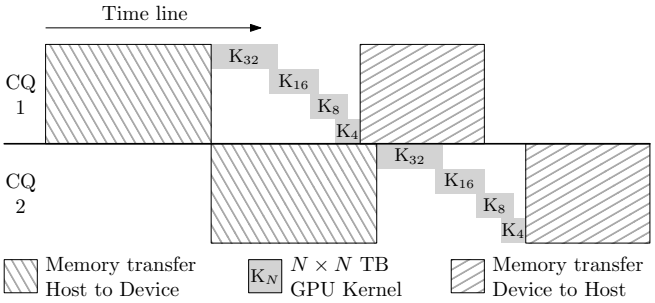

Fig. 3. Overlap of memory transfers and GPU kernels when using two OpenCL command queues (CQ).

the $N \times N$ TB. To ensure an efficient overlap between the required data transfers to the GPU with the kernel executions, the residual frame to be decompressed is divided horizontally and processed in different CQ. Accordingly, the received coefficients are sent to the GPU and the residual data is obtained after the four OpenCL kernel executions. Furthermore, for GPUs with overlapped kernels capability, the kernel overlapping is also applied at the end of each kernel, in order to use the full capability of the GPU.

Moreover, to take full advantage of the maximum GPU processing capability, the number of work-items assigned to each work-group is different for each individual OpenCL kernel (K_N). Accordingly, the work-group size is chosen by taking into account the GPU capability and the amount of resources that are required for each kernel (e.g., amount of local and private memory). Nevertheless, the work-group size is always a multiple of the TB size.

5. EXPERIMENTAL RESULTS

To assess and experimentally evaluate the processing throughput of the proposed OpenCL parallel implementation, the common test conditions and configurations defined in [11] were adopted, by considering video bit streams of all sequences from classes A and B (the greatest frame resolutions). An additional set of sequences with 3840×2160 pixel resolution (Ultra HD 4K) from the SVT High Definition Multi Format Test Set [12], composed by *CrowdRun*, *ParkJoy* and *DucksTakeOff*, was also evaluated. Those sequences will be referred to as class S (see Table 2).

The proposed de-quantization and inverse transform parallel implementation was integrated in the HEVC Test Model (HM) version 11.0 [13] decoder, which was also used as one of the benchmarks, for baseline comparison purposes. Although the proposed parallel implementation can be executed in any OpenCL supporting device (including mobile GPUs), it was decided to evaluate the attained processing efficiency in state-of-the-art GPU architectures supporting the latest communication-computation kernel overlapping techniques. Accordingly, the proposed implementation was prototyped in an off the shelf desktop system, which includes an Intel® Core™ i7-4770K CPU @ 3.50GHz and three NVIDIA GPUs compliant with OpenCL version 1.1: Tesla K20c @ 706 MHz (K20c), Tesla K40c @ 876 MHz (K40c) and GeForce GTX 780 Ti @ 1046 MHz (G780).

The size of the 1D work-groups for each kernel K_N were chosen according to the resources that are available in each of the three used GPUs and to each OpenCL kernel requirements. As an example, 64 work-items were used in each work-group for the K_{32} kernel, since this kernel requires 32×32×16 bits space in the OpenCL local memory to store the intermediate transformed block for each group of 32 work-items. In fact, a greater number of work-items per work-group would decrease the kernel performance, since the work-group occupancy in the GPU multiprocessor would also diminish due to the local and private memory limitation. However, the 1D work-group size can increase when the TB

Class (Resolution)	Sequence	FPS	I Frames (All Intra)				B Frames (Random Access)			
			HM 11.0	K20c	K40c	G780	HM 11.0	K20	K40	G780
S (3840×2160)	CrowdRun	50	117.32	12.61	8.95	6.44	55.78	10.03	6.72	6.50
	ParkJoy	50	115.64	12.94	9.22	6.46	107.46	13.20	9.38	6.51
	DucksTakeOff	50	120.79	12.15	8.61	6.39	41.44	9.97	6.56	6.47
A (2560×1600)	Traffic	30	59.05	6.90	4.89	3.43	9.24	5.35	3.46	3.39
	PeopleOnStreet	30	59.23	6.71	4.77	3.41	23.23	5.47	3.57	3.42
	Nebuta	60	63.84	7.16	5.86	4.13	53.76	6.02	4.97	3.58
	SteamLocomotive	60	61.99	6.96	4.95	3.60	28.58	5.70	3.95	3.45
B (1920×1080)	Kimono	24	32.10	4.00	3.02	2.35	9.04	3.04	2.16	2.00
	ParkScene	24	30.18	3.79	2.70	2.19	5.44	3.02	2.12	1.94
	Cactus	50	29.60	3.77	2.69	2.26	7.79	3.15	2.27	2.08
	BQTerrace	60	31.73	3.53	2.37	2.11	8.68	3.05	2.19	2.05
	BasketballDrive	50	32.56	3.84	2.66	2.29	9.70	3.20	2.29	2.11

Table 2. Frame processing time (in milliseconds) for the parallel implementations of the HEVC de-quantization and inverse transform modules.

size decreases. As an example, for K_{16}, K_8 and K_4, 96, 128 and 224 work-items were used per work-group, respectively.

Table 2 depicts the obtained average frame processing times for the HEVC de-quantization and inverse transform for each of the considered test sequences. A QP equal to 22 was adopted, corresponding to the most time consuming recommended case [11]. The experimental data was divided in I and B frames, from *All Intra* (AI) and *Random Access* (RA) configurations, respectively. The *Low Delay* configuration was not taken into account, since the corresponding results for B frames have a similar timing as in the RA configuration. The same happens with the I frames in the RA and AI configurations. Accordingly, this setup avoids a misleading analysis between I and B frames.

Naturally, the average execution time increases with the frame resolution. Nevertheless, although the HM 11.0 CPU implementation presents a significant variability from frame to frame (since B frames require a lower amount of operations than I frames), the timing for all GPU implementations is almost constant for a specific resolution and it is independent from the QP, slice type or video content, since it mainly depends on the amount of data to be processed.

As it can be observed, all the conducted evaluations revealed the capability to process all video sequences in real-time, reaching speed up values as high as 18.9× when compared to the HM 11.0 CPU implementation. In the worst case scenario, corresponding to 50 frames per second (fps) for class S, one frame should be decoded in less than 20 ms. For such processing conditions, the G780 GPU implementation required less than 6.51 ms to perform the de-quantization and inverse transform, leaving more time to execute the remaining video decode modules.

6. CONCLUSION

An efficient OpenCL parallelization of the HEVC de-quantization and inverse transform to be executed on GPU accelerators was presented in this paper. The conducted GPU parallelization can handle the required real-time decode requisites, by adopting a new approach for the HEVC de-quantization and inverse transform with minimal memory accesses and no synchronization points. To the best of the authors' knowledge, the proposed GPU parallelization of the HEVC residual data processing module is one of the first in the literature. By exploiting the full GPU computational resources, the proposed implementation was able to achieve speedup values as high as 18.9×, obtained for the Ultra HD 4K (3840×2160 pixels) video sequence *DucksTakeOff*, corresponding to an average frame processing time of 6.39 ms.

REFERENCES

[1] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (HEVC) standard," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 6, pp. 1029–1041, Dec. 2013.

[2] Khronos OpenCL Working Group, version: 1.1, Doc. Revision: 44, *The OpenCL Specification*, Jun. 2011.

[3] D. F. de Souza, N. Roma, and L. Sousa, "Cooperative CPU+GPU deblocking filter parallelization for high performance HEVC video codecs," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 5026–5030.

[4] W. Xiao, F. Wu, J. Xu, and G. Shi, "Fast HEVC encoding with GPU assisted reference picture selection," in *Advances in Multimedia Information Processing – PCM 2013*, vol. 8294 of *Lecture Notes in Computer Science*, pp. 233–244. Springer International Publishing, 2013.

[5] S. Momcilovic, A. Ilic, N. Roma, and L. Sousa, "Dynamic load balancing for real-time video encoding on heterogeneous CPU+GPU systems," *Multimedia, IEEE Transactions on*, vol. 16, no. 1, pp. 108–121, Jan. 2014.

[6] C. G. Kim and Y. S. Choi, "A high performance parallel DCT with OpenCL on heterogeneous computing environment," *Multimedia Tools and Applications*, vol. 64, no. 2, pp. 475–489, 2013.

[7] K. Lee, H.-J. Lee, J. Kim, and Y. Choi, "A novel algorithm for zero block detection in high efficiency video coding," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 6, pp. 1124–1134, Dec. 2013.

[8] L. A. Sousa, "General method for eliminating redundant computations in video coding," *Electronics Letters*, vol. 36, no. 4, pp. 306–307, Feb. 2000.

[9] H. Yong, R. Wang, W. Wang, Z. Wang, S. Dong, B. Han, and W. Gao, "Acceleration of HEVC transform and inverse transform on ARM NEON platform," in *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, Nov. 2013, pp. 169–173.

[10] Q. Shang, Y. Fan, W. Shen, S. Shen, and X. Zeng, "Single-port SRAM-based transpose memory with diagonal data mapping for large size 2-D DCT/IDCT," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2014.

[11] F. Bossen, "Common test conditions and software reference configurations," Doc. JCTVC-L1100 of JCT-VC, Geneva, Switzerland, Jan, 2013.

[12] L. Haglund, "The SVT high definition multi format test set," Tech. Rep. Version 1.0, Sveriges Television AB (SVT), Sweden, Feb. 2006.

[13] JCT-VC, "Subversion repository for the HEVC test model version HM 11.0," 2013.