4K-UHD REAL-TIME HEVC ENCODER WITH GPU ACCELERATED MOTION ESTIMATION

Fumiyo Takano, Hiroaki Igarashi, Tatsuji Moriyoshi

System Platform Research Laboratories, NEC Corporation, Japan
{f-takano@ce, h-igarashi@hf, moriyosi@ce}.jp.nec.com

ABSTRACT

This paper proposes an efficient implementation of an HEVC encoder with highly parallelized motion estimation on GPUs. It is an obstacle to parallelization of conventional motion estimation that the processing results of the neighboring blocks are used for efficient data compression. This paper especially proposes two new methods on the encoder for relaxing the data dependencies by dividing the motion estimation process into multiple steps and using the results of the previous steps, instead of the results of neighboring blocks in the same step. This allows the full use of many processor cores on GPUs while maintaining compression efficiency. The experimental results show that the proposed encoder achieves 60 fps real-time encoding for 4K-UHD sequences and 10x speed-up with an acceptable small bit-rate increase compared to the x265 encoder.

Index Terms— H.265/HEVC, GPU, Parallel Processing

1. INTRODUCTION

Recently, demand for services with high-resolution video, such as 4K-UHD, has increased. The data size of a 4K-UHD video is 8 times larger than that of a conventional Full-HD video. To transmit huge 4K-UHD data, high efficiency video coding (HEVC) [1] which achieves double the compression ratio than H.264/AVC (a previous codec), is expected. While the compression ratio of HEVC is high, higher computation complexity has become a major issue. Even the x265 encoder [2], a well-known highly optimized practical HEVC encoder, takes over 10 times longer than the playback duration to encode the same video.

To solve this problem, accelerated implementations with a graphic processing unit (GPU) have been studied [3][4][5]. Since a GPU is equipped with thousands of processor cores, the peak performance of a GPU is 6 times higher than that of a CPU. Massive parallelism to fully use many processor cores is essential to attain high performance on GPUs. Conventional GPU-based methods [3][4] are focused on increasing in the degree of parallelism, with block-level parallelization, and greatly increasing the speed of the motion search process, which is the most time-consuming function, as shown in Figure 1. However, the processing speed of these methods is far from real-time encoding. The processing time of these methods can be over 20 times longer than the playback duration. This insufficient processing speed is caused by using full search algorithm, which has a high degree of parallelism but requires high computational complexity.

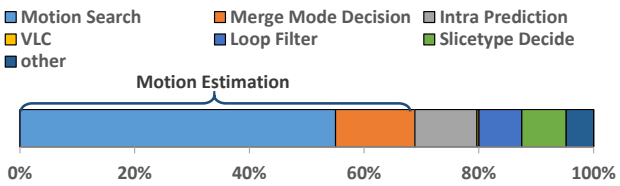

Figure 1. Encoding time distribution of x265 encoder

It is also essential to accelerate the other functions of encoder for total encoding acceleration. The GPU implementations of the other functions, such as the intra prediction [6] and the loop filter [7], have been proposed. However, any GPU implementation of Merge mode decision which is the second highest workload function, has not been introduced. The Merge mode decision has strict data dependence to conform to the HEVC specification unlike motion search which has loose data dependence. The data dependence makes it difficult to parallelize the Merge mode decision process for GPUs.

Another conventional GPU implementation [5], which applies GOP- and slice-level parallelization achieves 4K-UHD real-time encoding. This work offloads only a part of the motion estimation process to the GPU, while the other processes which are difficult to parallelize for a massive scale are executed on CPUs. Because these CPU processes require high computational complexity, this implementation requires 64 CPU cores. Therefore, this implementation requires a large-scale system consisting of two PCs with four CPUs and one PC with GPUs. A large-scale system is not suitable for practical use cases, such as video editing and video surveillance, from the point of view of reliability, footprint, and power consumption.

We propose a practical HEVC encoder with GPU acceleration running on a single PC system. Most parts of our encoder are offloaded to GPUs. We especially propose two key methods for motion estimation. One is a parallel motion search method based on hierarchical search to reduce computation complexity. The other is a parallelization method of the Merge mode decision process. These methods divide motion search and mode decision processes into respective multiple steps, and use data of neighboring blocks in the

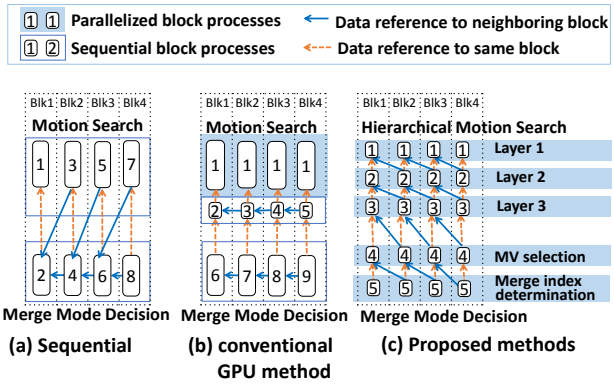

Figure 2. Processing flow and data dependence

previous step. This enables the full use of many processor cores on GPUs while maintaining compression efficiency.

2. HEVC MOTION ESTIMATION

This section explains the process of motion estimation and its data dependence. The HEVC initially divides each picture into pixel blocks that are called coding units (CUs). The encoding process is carried out on each CU. In the motion estimation process, a motion vector (MV) of each CU representing the motion between an encoded picture and a current picture is estimated, then a predicted picture is produced with the encoded picture and MVs. Only MVs and difference signals between the current and predicted picture are encoded. The motion estimation process consists of motion search and Merge mode decision process.

Motion search is a process of searching MVs by using block matching with the following cost function.

$$Cost = SAD + \lambda R$$

where SAD stands for the sum of absolute difference and indicates the difference between the predicted and current picture, R is the bit cost determined by the difference between a current MV and a motion vector predictor (MVP), and λ is a Lagrangian multiplier. An MVP is derived from MVs of neighboring blocks.

For further compression of MV information, HEVC introduces a new coding tool called "Merge mode". For a Merge mode block, an MV is copied from one of the neighboring blocks, and only an index to identify the MV source block is signaled. The source MV is selected from five candidates. The Merge mode decision process compares these Merge candidates and MV as motion search result then determines the best MV and whether to use the Merge mode. In the case of using the Merge mode, the Merge index is also determined.

3. CONVENTIONAL GPU PARALLELIZATION

To achieve better GPU performance, massive parallelism without data dependence is essential. Since a GPU has thousands of simple processor cores, it is suitable for a massive amount of simple operations without branch conditions.

A general sequential implementation of motion estimation is shown in Figure 2(a). The numbers in the figure represent the processing order, and the processes having the same number are carried out in parallel. Although the MVP and Merge mode increase compression efficiency, they bring the data dependence among blocks because the MVP and Merge candidates need to be derived after the neighboring blocks are processed. Figure 2(a) shows that motion search is processed after Merge mode decision process on neighboring blocks because the motion search and Merge mode decision processes refer to the result of the Merge mode decision process on the neighboring blocks. The MVP and Merge mode are obstacles in parallelizing motion estimation process.

A GPU-parallelization method has been proposed [3] for motion search. This conventional method relaxes the data dependence caused by MVPs and consists of two steps: a step of GPU-parallel processing without neighboring MVs, and a step of CPU-sequential processing with neighboring MVs, as shown in Figure 2(b). In the GPU step, an approximate MV (AMV) is estimated by full search on ±64 pixels while the zero vector is used as an estimated MVP. The GPU step is processed in parallel (process 1 in Figure 2(b)) because neighboring MVs are not referred. In the CPU step, the CPU refines MVs by searching small ranges around AMVs with the accurate MVPs. The CPU step is processed sequentially (processes 2 to 5 in Figure 2(b)). The motion search implementation of the conventional method is 115 times faster than x265 with full search. However, hexagon search which is a fast sequential algorithm, as a default setting called "medium preset" of x265 is commonly used for practical use. The conventional method is not adequate for practical use, since the processing time of the conventional implementation is 6 times as long as that of x265 with hexagon search. In addition, although another conventional motion search implementation [4] based on full search is 510 times faster than HM [8] which is the HEVC reference software encoder, [4] is also 4 times slower than x265 with hexagon search. The two main reasons the conventional methods use full search which requires high computational complexity instead of hexagon search are follows. One is that full search only consists of simple operations without branch conditions, and the other is that full search has a massive degree of parallelism enabled by pixel-, vector- and block-level parallelism. Another problem of the conventional methods is that they only searches less than ±64 pixels. The computation complexity of full search increases proportionally to the search range. A higher-resolution video has a longer MV, so encoding 4K-UHD videos requires a larger search range than ±64.

Furthermore, the conventional implementations do not execute the Merge mode decision process in parallel, because the Merge mode decision process has strict data dependence. As the motion search process with GPU accelerates, the next bottleneck is the Merge mode decision process. Parallelization of the Merge mode decision process for GPUs is also essential to reduce the entire encoding time.

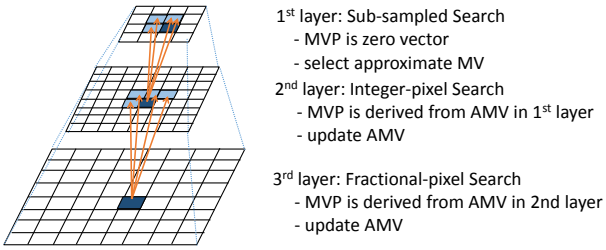

Figure 3. Overview of parallel hierarchical search

4. PROPOSED GPU ENCODER

In this section, we propose a parallel motion search based on hierarchical search to reduce computation complexity and a parallelization method for the Merge mode decision. The flow of the proposed methods is shown in Figure 2(c). These methods process all blocks in parallel to take advantages of GPUs. At the end of this section, we also give an overview of our encoder.

4.1. Parallelization of hierarchical search

In this subsection, we discuss our proposed parallelization method based on hierarchical search, which has low computational complexity. As mentioned above, the conventional GPU parallelization method [3] requires high computational complexity, and its search range is insufficient for 4K-UHD resolution. Hierarchical search has been proposed [9] as a search algorithm with a large search range and low computational complexity. Hierarchical search divides the search process into multiple layers and gradually narrows the search area in each layer. However, the MVP is required to narrow the search area. The conventional GPU parallelization method, which is used to replace an MVP with a zero vector, cannot be applied to hierarchical search. To solve this problem, this proposed method uses an approximate MVP derived from AMVs as a result of the previous layer. Thus, all blocks can be processed in parallel because there is no data dependence among the blocks in the same layer.

This proposed method consists of three layers, as illustrated in Figure 3. In the first layer, pictures are sub-sampled to 1/16, then AMVs are searched in a large range around the zero vector on the sub-sampled pictures. The search ranges of our implementation are ±265x±128 for P pictures and ±128x±64 for B pictures in integer-pixel accuracy. The search range configuration is equivalent to [9]. The zero vector is used as an MVP. In the second layer, AMVs are refined by searching three ±4x±2 areas on the original samples. The three-centers of the search areas are AMV of the first layer, approximate MVP and the zero vector. An approximate MVP is estimated using the first-layer AMVs. In the third layer, AMVs are further refined to fractional-pixel accuracy. An approximate MVP is also refined by referring to the AMVs of the second layer.

With this proposed method, using an approximate MVP derived from AMVs of the previous layer eliminates the data dependence among the blocks in the same layer. Therefore, all blocks in each layer can be processed in parallel.

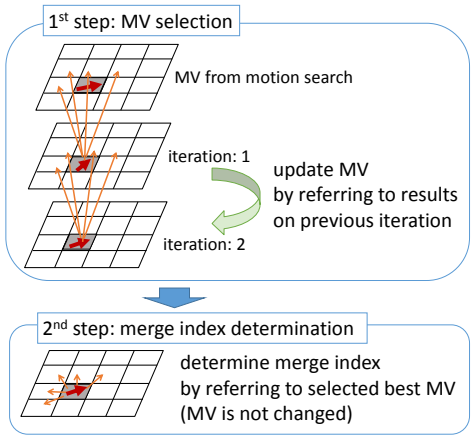

Figure 4. Overview of parallel merge mode decision

4.2. Parallelization of Merge mode decision

We now discuss our proposed parallelization method of Merge mode decision. The conventional sequential Merge mode decision process selects the best MV by referring to neighboring MVs as the Merge candidates then determines a Merge index for the selected best MV. Because this process for each block depends on the results of neighboring blocks, the process cannot be parallelized in block-level. Although the MVP on motion search can be substituted with an approximate vector, the Merge candidates require the correct neighboring MVs to conform to the HEVC specification.

For block-level parallelization, we divide the Merge mode decision process into two steps, as shown in Figure 4, one is MV selection, and the other is Merge index determination. In the MV selection step, approximate Merge candidates are derived from neighboring MVs resulting from the motion search process, then the best MV is selected from these candidates. In the Merge index determination step, the correct Merge candidates are derived from neighboring MVs resulting from the MV selection step, then the correct Merge index for the best MV is determined. The second step does not change the MV, it only sets the Merge index. Conformance with the HEVC specification is guaranteed by the second step. All blocks are processed in parallel in each step because the process of a block does not depend on the results of other blocks.

However, the compression efficiency with using the approximate candidates may deteriorate because the true optimal MV is not selected. To solve this problem, this proposed method iterates the MV selection step to approach the true optimal MV. In each iteration, the approximate candidates and MVs are updated by referring to neighboring MVs in the previous iteration, as shown in Figure 4. Approaching the true optimal MV improves compression efficiency.

Table 1. Comparison between proposed and previous methods

	[3]	[4]	Proposed methods
Search Algorithm	Full	Full	Hierarchical
Search Range	64x64	32x32	256x128/128x64
Merge Decision	Sequential	Sequential	Parallel

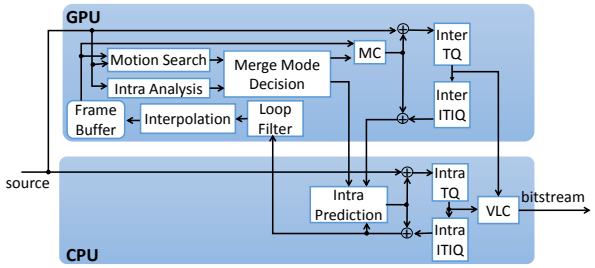

Figure 5. Block diagram of our encoder

To summarize the proposed methods, Table 1 lists the features of the proposed and conventional methods. In this table, we can see the search range of the proposed method is larger than that of the conventional methods, and only the proposed methods process Merge mode decision in parallel.

4.3. Structure of proposed encoder

This subsection introduces the structure of our encoder. Figure 5 illustrates a block diagram of the encoder, which is used on CPUs and GPUs. In HEVC, variable length coding (VLC) has only scanty parallelism. Since intra prediction, transformation and quantization (TQ), and inverse TQ (ITIQ) for intra blocks require accurate neighboring block information, they need to be processed sequentially. Therefore, these functions are processed on CPUs. The other functions such as motion search, Merge mode decision, motion compensation (MC), intra analysis, loop filter, inter TQ, and ITIQ, are offloaded to GPUs. The inter TQ is parallelized in the block-level because it has no data dependence [10]. The data dependence of intra analysis, which is a process for determining the best intra mode, can be relaxed using the original sample prediction [6]. Our encoder greatly benefits from GPU acceleration since most functions are offloaded to GPUs.

5. EXPERIMANTAL RESULTS

We evaluated the proposed encoder on the test platform listed in Table 2 and test conditions listed in Table 3. We used BD-Rate [11], which represents the increase ratio in the bit-rate on the same visual quality as an indicator of compression efficiency.

First, we evaluated the performance of our encoder and compared it with the x265 version 1.4. We used the default preset for the x265, except the GOP structure set, as listed Table 3, and the tuning option for PSNR. Figure 6 shows the encoding speed and BD-Rate of the two encoders. The anchor of the BD-Rate is x265 with medium preset. In the evaluation, the number of iterations on the proposed parallel Merge mode

Table 2. Experimental environment

CPU	Intel Xeon E5-2667 v3 (8 cores) 3.2GHz x 2
GPU	NVIDIA GeForce GTX TitanX x 2

Table 3. Experimental conditions

Test Sequence	SVT[12] 3840x2160, 10bit, 4:2:0, 5 sequences
GOP structure	M4N32
QP	27, 32, 37, 42

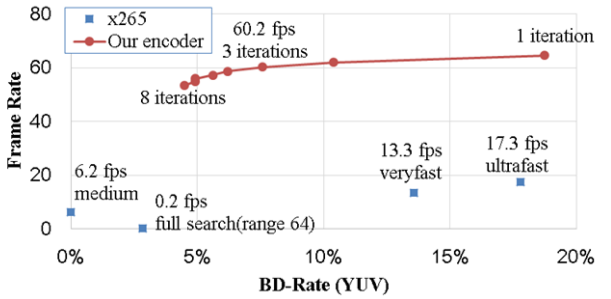

Figure 6. Performance comparison with x265

decision method was set from 1 to 8. Figure 6 shows that the increase in the number of iteration improved the BD-Rate, although it degraded the encoding speed. Our encoder with 3 iterations achieved 60 fps real-time encoding with 7.5% BD-Rate which is sufficiently small for practical applications. Moreover, when compared to ultrafast preset, our encoder attained better performance in both processing speed and compression efficiency.

Next, we compared our encoder with the conventional GPU implementations [3][4]. Our encoder is 10 times faster than the x265 with medium preset whereas the conventional implementations are slower than the medium preset as described in Section 3. As regards compression efficiency, the BD-Rate in the conventional implementations is similar to that in x265 full search with 64x64 search range. The BD-Rate in full search is larger than medium preset since the search range 64x64 is too small for 4K-UHD. As a result, the difference in the BD-Rate between our encoder and the conventional methods is even smaller than the difference between our encoder and x265 with medium preset. Summarizing the above, our encoder achieved better encoding speed with acceptably small bit-rate increase than the conventional implementations.

6. CONCLUSIONS

This paper proposed an efficient implementation of HEVC encoder with GPU acceleration. The key technologies of the encoder are the parallelization methods for hierarchical motion search and Merge mode decision. The proposed methods relax the data dependence by dividing the processes into multiple steps, and using the data of neighboring blocks in the previous step. The experimental results show that the proposed encoder achieves 60 fps real-time encoding for 4K-UHD sequences and 10 times speed-up with sufficiently small bit-rate increase compared to x265.

7. REFERENCES

[1] ITU-T Recommendation H.265 “High efficiency video coding,” 2015.

[2] x265 project, http://x265.org/

[3] F. Wang, D. Zhou and S. Goto, "OpenCL based high-quality HEVC motion estimation on GPU," 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 1263-1267.

[4] Yih-Chuan Lin and Shang-Che Wu, "Parallel motion estimation and GPU-based fast coding unit splitting mechanism for HEVC," 2016 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, 2016, pp. 1-7.

[5] T. K. Heng et al., "A highly parallelized H.265/HEVC real-time UHD software encoder," 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 1213-1217.

[6] S. Radicke, J. U. Hahn, Q. Wang and C. Grecos, "A Parallel HEVC Intra Prediction Algorithm for Heterogeneous CPU+GPU Platforms," in IEEE Transactions on Broadcasting, vol. 62, no. 1, pp. 103-119, March 2016.

[7] D. F. de Souza, A. Ilic, N. Roma and L. Sousa, "GPU acceleration of the HEVC decoder inter prediction module," 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Orlando, FL, 2015, pp. 1245-1249

[8] High Efficiency Video Coding (HEVC), https://hevc.hhi.fraunhofer.de/

[9] D. Zhou, J. Zhou, G. He and S. Goto, "A 1.59 Gpixel/s Motion Estimation Processor With - 211 to +211 Search Range for UHDTV Video Encoder," in IEEE Journal of Solid-State Circuits, vol. 49, no. 4, pp. 827-837, April 2014.

[10] H. Igarashi, F. Takano and T. Moriyoshi, "Highly parallel transformation and quantization for HEVC encoder on GPUs," 2016 Visual Communications and Image Processing (VCIP), Chengdu, 2016, pp. 1-4.

[11] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves,” ITU-T VCEG M-33, 2001.

[12] “The SVT high definition multi format test set”, https://tech.ebu.ch/docs/hdtv/svt-multiformat-conditionsv10.pdf