Load Balancing Based on Transform Unit Partition Information for High Efficiency Video Coding Deblocking Filter

Hochan Ryu, Seanae Park, Eun-Kyung Ryu, and Donggyu Sim

In this paper, we propose a parallelization method for a High Efficiency Video Coding (HEVC) deblocking filter with transform unit (TU) split information. HEVC employs a deblocking filter to boost perceptual quality and coding efficiency. The deblocking filter was designed for data-level parallelism. In this paper, we demonstrate a method of distributing equal workloads to all cores or threads by anticipating the deblocking filter complexity based on the coding unit depth and TU split information. We determined that the average time saving of our proposed deblocking filter parallelization method has a speed-up factor that is 2% better than that of the uniformly distributed parallel deblocking filter, and 6% better than that of coding tree unit row distribution parallelism. In addition, we determined that the speed-up factor of our proposed deblocking filter parallelization method, in terms of percentage run-time, is up to 3.1 compared to the run-time of the HEVC test model 12.0 deblocking filter with a sequential implementation.

Keywords: High efficiency video coding (HEVC), Deblocking filter, Parallelization method, Load balancing, Video coding.

I. Introduction

With recent advances in multimedia technologies, the demand is increasing for higher resolution video services with ultra-high definition. In addition, the widespread distribution of smartphones and tablet personal computers and innovations in network technology are facilitating various high-resolution video services, even in mobile environments. To address these market challenges, a compression standard has become necessary for the efficient compression of high-resolution videos. Thus, the ISO/IEC Moving Picture Experts Group and the ITU-T Video Coding Experts Group established a joint collaborative team for video coding and developed the next-generation video compression standard, known as High Efficiency Video Coding (HEVC). In January 2013, the Final Draft International Standard for HEVC was released [1], [2].

With the establishment of the HEVC standard, fast and optimized encoders and decoders are now essential to ensuring that HEVC is quickly distributed in the market [3]–[5]. HEVC uses two in-loop filters (a deblocking filter and a sample adaptive offset) to improve perceptual quality and coding efficiency [6], [7]. These in-loop filters are applied to the decoded picture to minimize errors and improve subjective and objective image quality. In addition, the in-loop filtered frames are used as reference pictures in inter-prediction modules based on the prediction performance. However, for HEVC decoders, these two in-loop filters increase the computational load. Consequently, the in-loop filtering in the decoding process accounts for approximately 12% of the total complexity of the HEVC test model (HM) 12.0 decoder with the HEVC main profile [1], [8]. We computed the complexity portion by measuring the running time with HM 12.0. To obtain the figures, we employed respective all-intra, random access, low-delay *B*, and low-delay *P* coding configurations with quantization parameters (QPs) of 22, 27, 32, and 37. Table 1 shows the decoding time of the

Manuscript received Sept. 6, 2016; revised Jan. 18, 2017; accepted Feb. 2, 2017. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A2A1A11052210).

Hochan Ryu (jolie@kw.ac.kr), Seanae Park (psea1118@kw.ac.kr), Eun-Kyung Ryu (dms0314@kw.ac.kr), and Donggyu Sim (corresponding author, dgsim@kw.ac.kr) are with the Department of Computer Engineering, Kwangwoon University, Seoul, Rep. of Korea.

Table 1. Portion of in-loop filter time of the whole decoding time.

Sequence	QP	Decoding time (s)			Portion (%)	
		DBF	SAO	Total	DBF	SAO
BasketballDrive	22	6.93	2.31	87.07	7.96	3.23
	27	5.79	1.96	69.71	8.31	2.81
	32	5.04	1.17	60.99	8.26	1.92
	37	4.48	0.73	54.93	8.16	1.33
BQTerrace	22	11.65	3.10	133.84	8.70	2.32
	27	8.97	2.50	85.42	10.50	2.92
	32	7.76	1.37	70.55	11.00	1.94
	37	6.87	1.13	63.51	10.81	1.78
Cactus	22	9.18	2.40	87.22	10.53	2.75
	27	7.61	1.69	63.96	11.90	2.64
	32	7.05	1.16	55.61	12.67	2.08
	37	6.37	0.74	49.53	12.85	1.49
Kimono	22	4.39	1.07	42.90	10.22	2.49
	27	3.33	0.78	31.66	10.52	2.45
	32	2.99	0.45	27.85	10.73	1.62
	37	2.70	0.23	25.41	10.61	0.91
ParkScene	22	6.52	1.54	69.32	9.41	2.22
	27	3.81	0.90	34.42	11.06	2.62
	32	3.45	0.63	29.56	11.69	2.13
	37	3.03	0.37	25.97	11.68	1.44
Average		5.90	1.34	58.48	10.38	2.15

*DBF: deblocking filter.

deblocking filter and sample adaptive offset (SAO). On average, the deblocking filter accounts for 10.38% and SAO accounts for 2.15%.

To reduce the computational complexity of fast video decoders, many studies have been conducted on acceleration of the decoding speed using data-level parallelism such as single-instruction multiple data (SIMD) [9], [10]. Unlike motion compensation and inverse transform modules, SIMD is not appropriate for application in in-loop filters. The HEVC deblocking filter consists of multiple stages such as the filtering on/off decision, strong/weak filter selection, and pixel-wise filtering for 4 × 4 pixel lines. For each line, each pixel is filtered with different filter coefficients. Thus, we cannot load multiple data with the same pattern. Depending on the pixel location, different loading types and different coefficients are required. Thus, it is not easy to apply SIMD to the deblocking filter. In addition, vertical loading is required for the vertical deblocking filter. In this case, it would not be efficient to load the pixel data. We do not contend that SIMD cannot be used for deblocking; however, it is not easy to apply SIMD to it. Accordingly, the proportion of in-loop filters becomes higher in optimized decoders. Thus, in-loop filter parallelization is necessary and effective in fast, optimized decoders on multi-core platforms.

Today, mobile devices and personal computers equipped with multi-core processors are common, and the trend is expected to continue. To maximize processor efficiency, many applications are structurally modified to facilitate parallel processing. Moreover, new applications are being developed that assume parallelization capability. For real-time decoding, many video decoder parallelization methods have been investigated to maximize the efficiency of multi-core processors. With respect to the parallelization of deblocking filters, Jo and others [11] proposed a method in which a slice is equally partitioned into many regions, and each region is then allocated to a core. However, a parallel algorithm in which a frame is equally divided has an unequal number of boundaries to which the deblocking filter is applied. This lowers the parallelization efficiency on account of the workload imbalance. Ikeda and others [12] proposed a parallelized algorithm that allocates a core to each coding tree unit (CTU) row. However, it has a problem in maximizing the parallelization efficiency on account of the workload imbalance over the partitioned regions. In addition, coding unit (CU) partition information is used to estimate the complexity of the deblocking filter [13]. Acceleration of parallelization is improved for the existing algorithms. Nevertheless, the algorithm cannot handle accurate deblocking complexity with CU partition information.

Parallelization efficiency can differ according to how work items are assigned to cores or threads. When an actual processing quantity can be accurately estimated, it enables the equal assignment of work to each core or thread. It can thus decrease the idle time of the cores and improve parallelization efficiency. In contrast, with inaccurate complexity estimation, the last core processed determines the total processing time, which decreases parallelization efficiency. In this paper, we propose a method for maximizing parallelization efficiency, wherein the deblocking filter complexity is considered when assigning data to each core. This approach effectively reduces the complexity of the HEVC deblocking filter. With a minimum computational load, we estimate the complexity of the CTU based on the CU depth and transform unit (TU) split information. The total computation complexity is then equally divided among the total number of cores or threads. The decoding speed factor of the proposed deblocking filter based on complexity estimation is 2% and 6% higher than those of the existing uniform distribution deblocking filter parallelism and CTU row distribution parallelism, respectively.

The remainder of this paper is organized as follows. In Section II, we introduce the HEVC deblocking filter and previous research in video decoder parallelization. In

Section III, we describe the proposed HEVC deblocking filter parallelization method. In Section IV, we demonstrate and analyze the performance of the proposed parallelization method in a comparative evaluation. Finally, we present our conclusions in Section V.

II. HEVC Deblocking Filter and Existing Video Decoder Parallelization Methods

The HEVC deblocking filter is similar to the H.264/AVC deblocking filter in terms of its basic concept; however, the HEVC deblocking filter has lower complexity. Moreover, the HEVC deblocking filter was designed to have a reduced level of dependence between adjacent blocks so that it can be performed in parallel. Proper parallelization of the HEVC deblocking filter is important for realizing maximum parallelization performance in practical implementations.

In video compression, quantization results in both quality degradation and artifact blocking due to the block-based coding structures. Since blocking artifacts significantly affect perceptual quality, HEVC also employs a deblocking filter to reduce the artifacts. In each picture, the vertical boundary of the prediction unit (PU) or TU is first horizontally filtered. Then, the horizontal boundary is vertically filtered. This design is more efficient in terms of data-level parallelism than the horizontal and vertical filtering of H.264/AVC at the macro-block level because the former eliminates the dependence between adjacent blocks [14]. While the H.264/AVC deblocking filter is appropriately used on 4 × 4 block boundaries, the HEVC deblocking filter should be applied on 8 × 8 block boundaries to lower the computational load.

Figure 1 shows the HEVC deblocking filtering process. In the HEVC deblocking filter, the boundary candidates to which the deblocking filter will be applied are first determined. These boundary candidates are 8 × 8 TU or PU boundaries. For each candidate boundary, the boundary strength (BS) (0, 1, or 2) is computed depending on the prediction modes, transform coefficients, reference picture index, and motion vector. When the BS is 0, no deblocking filter is applied to the boundary. For boundaries with a BS of 1 or 2, the β and t_c values are computed. We must again decide whether the boundary is to be filtered, depending on computed β and luminance differences between neighboring pixels. When the luminance difference is sufficiently small, the difference arises from quantization, rather than from the original edges. Then, we determine whether a strong or weak filter is appropriate depending on the β and t_c values.

Parallelization can be categorized as either task-level parallelism (TLP) or data-level parallelism (DLP). TLP classifies all tasks at a functional level and assigns them to

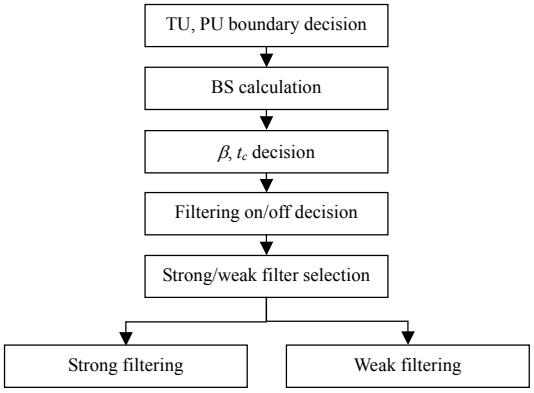

Fig. 1. HEVC deblocking filtering process.

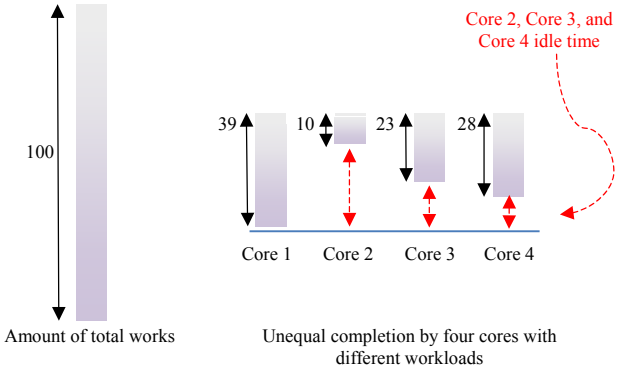

Fig. 2. Workload imbalance problem between the cores.

cores or threads, which requires that data be transferred between dependent tasks. TLP is generally used when the data transfer quantity between tasks is not large. DLP divides all the data and assigns each data item to a core or thread. DLP can yield maximum parallelism performance when there is no dependence among the tasks assigned to the cores or threads. In DLP, parallelization efficiency determines how to divide the workload and assign the partitioned jobs to the cores. Because each core performs all the video decoding processes for the designated area of a picture, there is no overhead for data communication between threads. In addition, because DLP has a higher scalability than TLP, it can be easily modified to increase or decrease the number of cores.

In DLP, when the quantity of the workload is proportional to the quantity of data, partitioning of equal amounts of data is generally desirable. However, when the work required for each amount of data varies, then equal data partitioning reduces the parallelization efficiency on account of the workload imbalance between the cores. As shown in Fig. 2, three cores remain idle until the last task is completed because they have different workloads. If one or several cores are overburdened, the other cores become idle, even though their tasks were completed. Consequently, the parallelization performance degraded. In contrast, if the workload is equally distributed

among the cores, the total work will be simultaneously finished, thereby increasing parallelization efficiency. However, it is not possible to allocate equal workloads when it is not known how to optimally divide the tasks before processing. Thus, it is very important to accurately estimate workloads prior to execution.

Thread-queue parallelization is a widely used DLP method in the development of video decoders. This method divides a given workload into certain units and schedules the cores to maximize parallel performance. Additionally, it ensures the equal division of the workload. When decoding a picture, we can divide one task into multiple tasks, such as one CTU or more. Then, we can use thread-queue parallelization by which a single core executes the next task immediately after the initial task is finished. However, the existence of many units can cause more overhead, such as context switching, with respect to core scheduling.

The number of HEVC CTUs in one picture (3,840 × 2,160) is 2,040. In this case, the scheduling overhead can become a dominant time-consuming part. The maximum size of the HEVC CTU is 64 × 64. When a pixel is expressed in 8 bits and 4:2:0 chroma subsampling, the data quantity of a CTU is approximately 6 kB. This is 16 times more than that of an H.264/AVC macroblock. Thus, it is known that the optimum number of job partitions is required to reduce the context switching overhead before job assigning.

Figure 3 shows a video DLP method that partitions the picture into many areas and assigns each partitioned area to a core. This method simply facilitates parallelization without requiring additional scheduling after a partitioned area is assigned to a core. Furthermore, the data can be loaded sooner using an independent bus with direct memory access because the next block to be processed can be predicted. By this parallelization method, an entire picture can be equally partitioned and the cores can be assigned to the partitioned pictures for parallelization. However, an equal amount of data does not generate the same amount of workload for the video decoders. Deblocking filters are adaptively on/off depending on the block locations, and different regions require different computational loads. Therefore, equal-sized picture partitioning does not maximize parallelization performance because of the workload imbalance, and parallelization performance depends on the visual characteristics of the pictures.

Considerable research has been devoted to the video coding standard, H.264/AVC, with respect to reducing the dependency of the deblocking filter on multi-core platforms [15], [16]. In the H.264/AVC deblocking filter, horizontal and vertical filtering are sequentially employed at the macroblock level. In HEVC, on the other hand, horizontal and vertical deblocking filters can be applied at the picture level. Thus, we can say that the dependency of HEVC deblocking filtering is quite low.

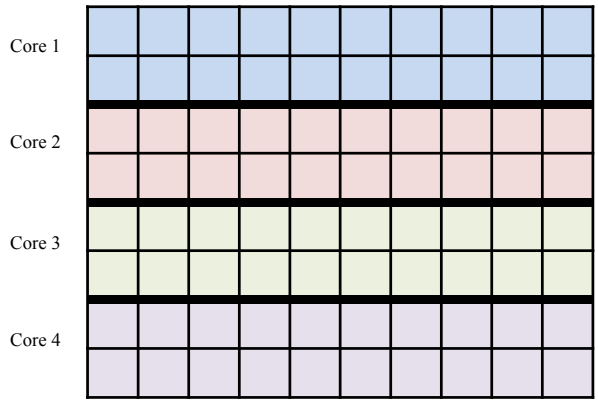

Fig. 3. Parallelization method with picture partitioning.

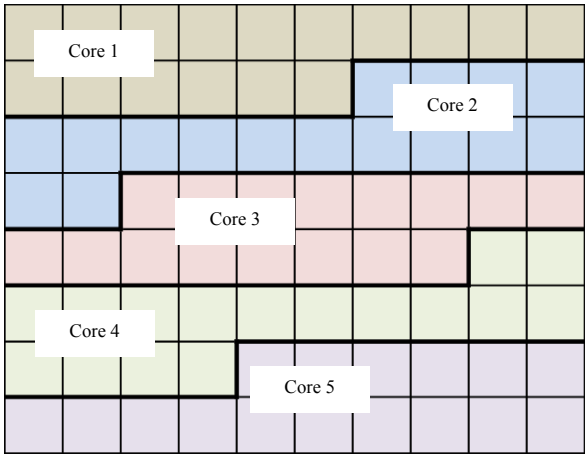

Fig. 4. Parallelization method by allocating the same number of CTUs to each core.

Furthermore, an algorithm was proposed for allocating the same number of CTUs to each core [11]. It divides the deblocking filtering process by separating the horizontal and vertical filtering at the picture level and employing CTU-level DLP to each filtering process, as shown in Fig. 4.

In another algorithm [12], each CTU row is processed by a core. These algorithms both suffer from unequal workloads, depending on the video characteristics. To alleviate this problem, a parallelization method with a load balancing algorithm was proposed with CU partitioning information [13]. However, the algorithm [13] cannot deal with an accurate deblocking filter complexity with CU partition information because the deblocking filter is performed on TU and PU boundaries. In a CU block, the computation complexity of the deblocking filter could thus differ from the CU partition information only.

III. Proposed Deblocking Filter Parallelization Based on Complexity Estimation

HEVC employs a deblocking filter and SAO to improve

ETRI Journal, Volume 39, Number 3, June 2017
https://doi.org/10.4218/etrij.17.0116.0638

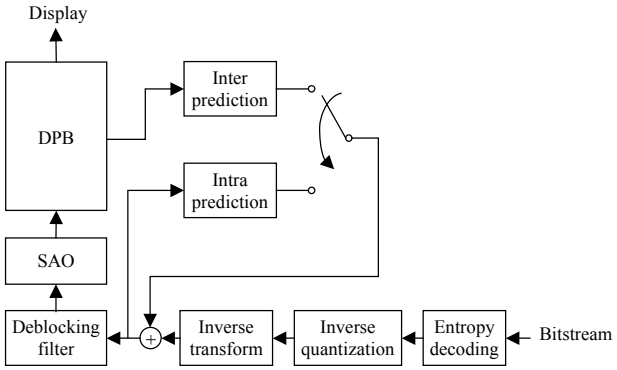

Fig. 5. Block diagram of the HEVC decoder.

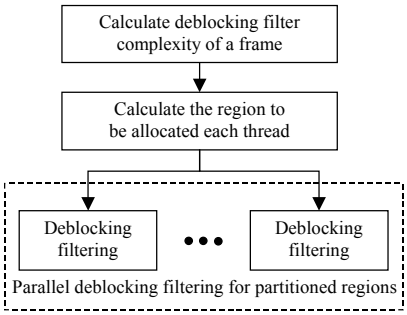

Fig. 6. Deblocking filter parallelization method based on complexity estimation.

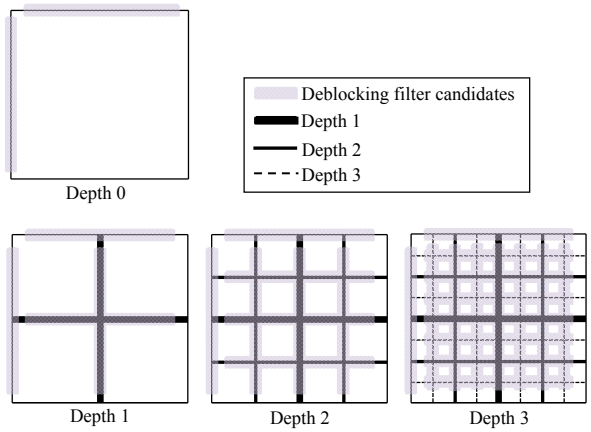

Fig. 7. Deblocking filter candidates according to the CU depth.

Table 2. Complexity factors of deblocking filter complexity according to the CU depth.

CU depth	Complexity factor	
	TU is not split	TU is split
0	8	16
1	4	8
2	2	4
3	1	1

compression performance. The deblocking filter and SAO are applied to the motion-compensated picture after inverse quantization and the inverse transform, as shown in Fig. 5. Several studies have focused on the parallelization of pixel decoding in HEVC decoders [17]. However, an HEVC decoder decodes each picture based on the previously filtered pictures. Thus, the running time performance of the HEVC decoder cannot be maximized by parallelizing only the pixel decoding aspect. In-loop filter parallelization is required to maximize the overall performance of the HEVC decoder. In addition, several studies have focused on SAO parallelization [11], [18]. In this paper, we propose deblocking filter parallelization and an efficient method of applying DLP to the HEVC deblocking filter. As described above, the existing parallelization method for the deblocking filter cannot maximize parallelization efficiency because of the workload imbalance. To address this problem, we propose a parallelization efficiency maximization method for predicting the computational complexity of the deblocking filter with CU and TU partition information and then equally distributing the workload across multiple cores.

Figure 6 shows the deblocking filter parallelization method, which is based on the estimation of computational complexity. First, we calculate the complexity of the deblocking filtering task for a specific frame. Based on the estimated complexity, we calculate the region to be assigned to a core or thread, and we assign the region to a core or thread for parallel deblocking filtering.

In the proposed method, we calculate the complexity of the deblocking filtering using not only CU but also TU partitioning information. Because the deblocking filter determines whether filtering will be performed, and it is applied to several selected PU or TU boundaries, as shown in Fig. 7, the CU partitioning information is closely related to the block boundary to which the deblocking filter is applied. To accurately evaluate the complexity, we use TU partitioning information from the CU to determine whether the TU is split. As shown in Fig. 7, we determine the number of 8 × 8 block boundaries based on the CU depth. When the CU depth is 0, the number of block boundaries is 16. We can have 8, 4, and 2 block boundaries for CU depths of 1, 2, and 3, respectively. In addition, if a CU is split into multiple TUs, the respective numbers of block boundaries are 32, 16, 8, and 2.

Table 2 calculates the deblocking filter complexity in a frame, which shows the ratio of the number of 8 × 8 block boundaries. Based on the CU depth, we use 8, 4, 2, or 1 to calculate the complexity of the deblocking filter. If the CU is split in a TU level, the complexity factor is doubled when the CU depth is not 3. Because the deblocking filter is not applied on PU or TU boundaries smaller than 8 × 8, TU split information is not

available in small blocks. In a frame, the complexity of the deblocking filter can be expressed by:

$$DF_{\text{complexity}} = \sum_{d=0}^{3} SCF_d, \quad (1)$$

where SCF_d is the sum of the complexity factors of CUs, where d represents the CU depth. We estimate the total complexity of the deblocking filter for each frame and then divide the total complexity by the number of threads or cores. In this way, we can determine the ideal equal workload to be processed by each thread or core. Next, we compute the accumulated complexity from the first to the last CTU, and we determine the group of CTUs in the raster scanning order to realize almost equal workloads for all the threads or cores.

IV. Experimental Results

To evaluate the performance of the proposed deblocking filter parallelization method, we implemented the proposed algorithm using HM 12.0 reference software [8]. For this implementation, we used OpenMP 2.0 [19] for parallelization and standard HEVC sequences for our evaluation [20]. We encoded each sequence in 22, 27, 32, and 37 QP to produce the bitstreams. Table 3 shows the test conditions for this evaluation. To reduce testing errors, we performed decoding five times and then calculated the average decoding time. To measure the degree of performance improvement achieved by the parallelization method, we computed the speed-up factor and average time saving (ATS) as follows:

$$Speed-up = \frac{DT_{\text{ref.}}}{DT_{\text{prop.}}}, \quad (2)$$

$$ATS\,(\%) = \frac{Speed-up_{\text{prop.}} - Speed-up_{\text{ref.}}}{Speed-up_{\text{ref.}}}, \quad (3)$$

where $DT_{\text{prop.}}$ is the decoding time when the proposed method is used, and $DT_{\text{ref.}}$ is the decoding time of the HM 12.0 decoder.

In Table 4, the P1 values are the speed-up factors when the same number of CTUs is allocated to each core for deblocking filtering [10]. The P2 column shows the speed-up factors when each CTU row is allocated to each core [11]. The P3 values are the speed-up factors by the proposed algorithm. It is evident that, on average, the speed-up factor in P3 is slightly higher than the others. With two threads, the proposed algorithm is 3% faster on average than P1 and P2. With four threads, the proposed algorithm is 2% and 6% faster on average than P1 and P2, respectively. However, the gain of the proposed algorithm varies significantly depending on the sequences. The proposed algorithm is 17% faster than P1 for the "BasketballDrive" sequence with two threads. For the "ParkScene" sequence, we found the proposed algorithm to be 10% better than P2 with a QP of 27. With four threads, the proposed algorithm is 17% and 19% faster than P2 for two sequences, respectively. We consider the proposed algorithm to be effective for several sequences that have different local characteristics. We found the estimated and actual complexities to be quite different depending on the CTU locations. In addition, more threads/cores can improve load-balancing accuracy for not only the proposed algorithm, but also existing algorithms. However, we found that the number of threads is not always proportional to acceleration factors on account of the context switching overhead.

Table 3. Experimental conditions.

Reference software	HM 12.0
Encoding conditions	Low-delay, Random-access, All-intra
QP	22, 27, 32, 37
Sequences	BasketballDrive, BQTerrace, Cactus, Kimono, ParkScene
Resolution	Full-HD (1,920 × 1,080)
CPU	Intel core i7 X990 3.47GHz
OS	Windows 7 (64-bit)
OpenMP version	2.0
Compiler	MSVC 2012

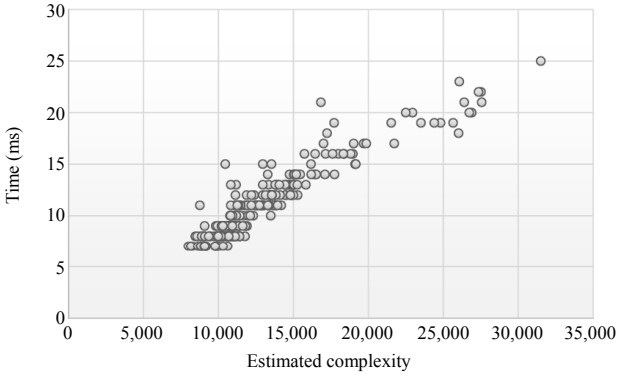

Fig. 8. Relationship between the actual run-time and the estimated complexity by the proposed algorithm.

Figure 8 shows the relationship between the actual deblocking filtering time and the estimated complexity, as calculated by the proposed method. As the estimated complexity increases, the deblocking filter time also increases. As shown in the figure, the correlation is quite high and the estimated complexity of the deblocking filter by the proposed algorithm is sufficiently accurate to be applied to load balancing. We obtained a correlation number of 0.84 for Fig. 8.

The estimated complexity with only CU split information [12] could differ according to actual complexity. For confirmation of the argument, we generated bitstreams by

Table 4. Comparison of decoder performance for different deblocking filter parallelization methods.

QP	Sequence	Decoding time (s) of deblocking filter according to the number of threads							Speed-up factors according to the number of threads					
		HM 12.0	2 threads			4 threads			2 threads			4 threads		
			*P*1[10]	*P*2[11]	*P*3	*P*1[10]	*P*2[11]	*P*3	*P*1[10]	*P*2[11]	*P*3	*P*1[10]	*P*2[11]	*P*3
22	BasketballDrive	6.93	4.07	3.89	3.96	2.45	2.44	2.19	1.70	1.78	1.75	2.83	2.84	3.16
	BQTerrace	11.65	6.69	6.68	6.66	3.69	3.87	3.81	1.74	1.74	1.75	3.16	3.01	3.05
	Cactus	9.18	5.06	5.12	4.95	2.85	2.96	2.98	1.81	1.79	1.85	3.22	3.10	3.05
	Kimono	3.71	2.02	2.07	2.05	1.12	1.16	1.14	1.84	1.79	1.81	3.32	3.19	3.24
	ParkScene	4.39	2.46	2.57	2.37	1.41	1.55	1.37	1.78	1.71	1.85	3.10	2.83	3.20
	Average	7.17	4.06	4.07	4.00	2.30	2.40	2.30	1.78	1.76	1.80	3.13	3.00	3.15
27	BasketballDrive	5.79	3.76	3.34	3.22	2.05	1.99	1.75	1.54	1.73	1.80	2.83	2.91	3.30
	BQTerrace	8.97	5.26	5.35	5.16	3.03	3.13	2.90	1.70	1.68	1.74	2.96	2.87	3.10
	Cactus	7.61	4.21	4.29	4.21	2.32	2.51	2.49	1.81	1.77	1.81	3.28	3.04	3.06
	Kimono	3.33	1.90	1.90	1.82	1.08	1.12	1.04	1.75	1.75	1.83	3.09	2.98	3.21
	ParkScene	3.81	2.20	2.32	2.11	1.19	1.35	1.28	1.73	1.64	1.80	3.19	2.82	2.98
	Average	5.90	3.47	3.44	3.30	1.93	2.02	1.89	1.71	1.71	1.79	3.07	2.92	3.13
32	BasketballDrive	5.04	3.15	3.13	2.86	1.96	1.68	1.76	1.60	1.61	1.76	2.57	2.99	2.87
	BQTerrace	7.76	4.49	4.60	4.34	2.60	2.73	2.46	1.73	1.69	1.79	2.98	2.84	3.15
	Cactus	7.05	3.93	3.96	3.92	2.22	2.37	2.37	1.79	1.78	1.80	3.17	2.97	2.97
	Kimono	2.99	1.77	1.74	1.69	0.96	1.02	0.93	1.69	1.72	1.77	3.12	2.93	3.22
	ParkScene	3.45	2.00	2.07	1.95	1.08	1.27	1.07	1.73	1.67	1.77	3.21	2.73	3.24
	Average	5.26	3.07	3.10	2.95	1.76	1.81	1.72	1.71	1.69	1.78	3.01	2.89	3.09
37	BasketballDrive	4.48	2.95	2.66	2.82	1.65	1.59	1.53	1.52	1.69	1.59	2.71	2.81	2.94
	BQTerrace	6.87	4.08	4.15	3.90	2.37	2.49	2.37	1.68	1.66	1.76	2.90	2.75	2.90
	Cactus	6.37	3.53	3.59	3.49	2.04	2.16	2.10	1.80	1.77	1.82	3.13	2.95	3.04
	Kimono	2.70	1.54	1.51	1.50	0.87	0.88	0.88	1.73	1.79	1.76	3.14	3.06	3.02
	ParkScene	3.03	1.71	1.78	1.70	0.99	1.04	0.94	1.77	1.70	1.78	3.05	2.91	3.23
	Average	4.69	2.76	2.74	2.68	1.58	1.63	1.56	1.70	1.72	1.74	2.99	2.90	3.02
Total average		5.75	3.34	3.34	3.23	1.90	1.97	1.87	1.72	1.72	1.78	3.05	2.93	3.10

coding half of the picture with CU and TU partition information and the remaining half of the picture with TU partition information. We found that the proposed algorithm with CU and TU split information is approximately 1.48 times better than the conventional algorithm using CU split information. We do not contend that the proposed algorithm is always significantly better than the conventional algorithms. However, the proposed algorithm can further accelerate the parallel performance for several videos that are coded with various TU splits.

V. Conclusion

In this paper, we proposed an efficient parallelization method for the HEVC decoder deblocking filter. To alleviate the workload imbalance problem and better ensure the efficient parallelization of the deblocking filter, we estimate the complexity to enable the distribution of equal workloads across multiple cores. The proposed deblocking filter based on complexity estimation improves the average time saving by 2% and 6% over those of uniform distribution deblocking filter parallelism and CTU row distribution parallelism, respectively. In addition, we determined that the speed-up factor of the proposed method in the percentage run-time was up to 3.1 compared to single core decoding with the deblocking filter. Further work is recommended to parallelize all parts of the HEVC decoder with a focus on workload balancing.

References

[1] *High Efficiency Video Coding*, Rec. ITU-T H.265 and ISO/IEC

23008-2, Jan. 2013.

[2] *High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS & Consent)*, JCTVC-L1003, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2013.

[3] Y. Kim et al., "A Fast Intra-Prediction Method in HEVC Using Rate-Distortion Estimation Based on Hadamard Transform," *ETRI J.*, vol. 35, no. 2, Apr. 2013, pp. 270–280.

[4] A. Lee et al., "Efficient Inter Prediction Mode Decision Method for Fast Motion Estimation in High Efficiency Video Coding," *ETRI J.*, vol. 36, no. 4, Aug. 2014, pp. 528–536.

[5] K. Goswami et al., "Early Coding Unit–Splitting Termination Algorithm for High Efficiency Video Coding (HEVC)," *ETRI J.*, vol. 36, no. 3, June 2014, pp. 407–417.

[6] A. Norkin et al., "HEVC Deblocking Filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1746–1754.

[7] C. Fu et al., "Sample Adaptive Offset in the HEVC Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1755–1764.

[8] HM-12.0, Accessed, 2016. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-12.0

[9] L. Yan et al., "Implementation of HEVC Decoder on x86 Processors with SIMD Optimization," *IEEE Vis. Commun. Image Process.*, San Diego, CA, USA, Nov. 27–30, 2012, pp. 1–6.

[10] Y. Ahn et al., "Implementation of Fast HEVC Encoder Based on SIMD and Data-Level Parallelism," *EURASIP J. Image Video Process.*, vol. 2014, Mar. 2014.

[11] H. Jo, D. Sim, and B. Jeon, "Hybrid Parallelization for HEVC Decoder," *Int. Congr. Image Signal Process.*, Hangzhou, China, Dec. 16–18, 2013, pp. 170–175.

[12] CE12 Subset2: Parallel Deblocking Filter, *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC)*, JCTVC-E181, Mar. 2011.

[13] H. Jo, S. Park, and D. Sim, "Parallelized Deblocking Filtering of HEVC Decoders Based on Complexity Estimation," *J. Real-Time Image Process.*, vol. 12, no. 2, Dec. 2015, pp. 369–382.

[14] Parallel Deblocking Filter, *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC)*, JCTVC-D263, Mar. 2011.

[15] S. Vijay, C. Chakrabarti, and L.J. Karam, "Parallel Deblocking Filter for H.264 AVC/SVC," *IEEE Workshop Signal Process. Syst.*, San Francisco, CA, USA, Oct. 6–8, 2010, pp. 116–121.

[16] B. Pieters et al., "Parallel Deblocking Filtering in MPEG-4 AVC/H. 264 on Massively Parallel Architectures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 1, Jan. 2011, pp. 96–100.

[17] H. Ryu et al., "Performance Analysis of HEVC Parallelization Methods for High-Resolution Videos," *IEIE Trans. Smart Process. Comput.*, vol. 4, no. 1, Feb. 2015, pp. 28–33.

[18] E. Ryu et al., "Sample Adaptive Offset Parallelism in HEVC," *Multimedia Ubiquitous Eng. Lecture Notes Electr. Eng.*, vol. 240, May 2013, pp. 1113–1119.

[19] L. Dagum and R. Menon, "OpenMP: an Industry Standard API for Shared-Memory Programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, Jan. 1998, pp. 46–55.

[20] Common Conditions and Software Reference Configurations, *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC)*, JCTVC-H1100, Feb. 2012.

Hochan Ryu received BS and MS degrees in Computer Engineering from Kwangwoon University, Seoul, Rep. of Korea, in 2013 and 2015, respectively. Since 2015, he has been an associate research engineer at Digital Insights Inc., Seoul, Rep. of Korea. His current research interests are video coding, video processing, parallel processing for video coding, and multimedia systems.

Seanae Park received BS, MS, and PhD degrees in Computer Engineering from Kwangwoon University, Seoul, Rep. of Korea, in 2004, 2006, and 2011, respectively. She is a senior research engineer at the Image Processing Systems Lab (IPSL) of Kwangwoon University. Her current research interests are high-efficiency video compression, standardization of future video coding, and multimedia systems.

Eun-Kyung Ryu received BS and MS degrees in Computer Engineering from Kwangwoon University, Seoul, Rep. of Korea, in 2011 and 2013, respectively. Since 2013, she has been an associate research engineer at LG Electronics Inc., Rep. of Korea. Her current research interests are image processing, video compression, and entropy coding.

Donggyu Sim received BS and MS degrees in Electronic Engineering from Sogang University, Seoul, Rep. of Korea, in 1993 and 1995, respectively. He earned a PhD degree at the same university in 1999. From 1999 to 2000, he was with Hyundai Electronics Co., Ltd., Icheon, Rep. of Korea, where he was involved in MPEG-7 standardization. From 2000 to 2002, he was a senior research engineer at Varo Vision Co., Ltd., Seoul, Rep. of Korea. where he worked on MPEG-4 wireless applications. He worked for the Image Computing Systems Lab (ICSL) at the University of Washington, Seattle, USA, as a senior research engineer from 2002 to 2005. He researched ultrasound image analysis and parametric video coding. Since 2005, he has been with the Department of Computer Engineering at Kwangwoon University, Seoul, Rep. of Korea. In 2011, he joined Simon Frasier University, Vancouver, Canada as a visiting scholar. He was elevated to an IEEE Senior Member in 2004. He is one of the primary inventors with many essential patents licensed to MPEG-LA for the HEVC standard. His current research interests are video coding, video processing, computer vision, and video communication.