The International Journal of Engineering and Science (IJES)
|| Volume || 7 || Issue || 11 Ver.II || Pages || PP 70-76 || 2018 ||
ISSN (e): 2319 – 1813 ISSN (p): 23-19 – 1805

Fast Motion Estimation For High Efficiency Video Coding Based On A Graphics Processing Unit

Yu-Kai Lin, Song-Hui Hong, Tung-Hung Hsieh, Chou-Chen Wang
Department of Electronic Engineering, I-Shou University, Kaohsiung, Taiwan
Corresponding Author: Chou-Chen Wang

----------ABSTRACT----------
In the newest high efficiency video coding (HEVC) standard, the motion estimation (ME) takes around 70% of the encoding time in HM encoder. In order to reduce the complexity of the ME module in HEVC, this paper proposes a flexible coding tree unit (CTU)-level parallel ME method using a graphics processing unit (GPU). The proposed method can be combined with fast CTU-level multiple reference frame motion estimation (MRF-ME) to further reduce the encoding time. Firstly, we decompose ME algorithm into three kernels to achieve a highly parallel computation with a low external memory on GPU. Secondly, the kernel 1 executes a GPU program of calculating the sum of absolute differences (SAD) of small coding unit (SCU 8×8). Thirdly, the kernel 2 merges the variable block size from SCU (8×8) to large coding unit (LCU 64×64). Finally, the kernel 3 compares minimum SAD to find the best matching block. Simulation results show that the proposed method can achieve an average time improving ratio of MRF-ME module about 96.68% and 97.78% when compared to HM16.7 under MRF=4 and MRF=8, respectively.

KEYWORDS: *Video coding standard, HEVC, GPU, Motion estimation.*

Date of Submission: 10-12-2018 Date of acceptance: 31-12-2018

I. INTRODUCTION

With the advancement of technology, the video panels of 4K×2K and 8K×2K high-resolution had become the main specification on digital TV. However, the early video coding standard H.264/AVC [1] is difficult to support the video applications of high definition (HD) and ultrahigh definition (UHD) resolution. Therefore, the Joint Collaborative Team on Video Coding (JCT-VC) consists of ITU-T and ISO/IEC which developed a newest high efficiency video coding (HEVC) for satisfying the UHD requirement in 2010, and the first version of HEVC was approved as ITU-T H.265 and ISO/IEC 23008-2 by JCT-VC in Jan. 2013 [2]. HEVC coding standard is better than H.264/AVC which can reduce 50% bitrate in the almost same video quality under high profile (HP). HEVC adopts new coding structure that includes coding unit (CU), prediction unit (PU) and transform unit (TU) in a quadtree-structured coding tree unit (CTU) and each CTU can be split into four different depth sub-CUs to do intra prediction and inter prediction. In inter prediction, the well-known most time-consuming part is the motion estimation (ME) module, it needs to compute thousands of times with the sum of absolute difference (SAD) algorithm in search window (SW) to achieve the best motion vector (MV). In order to obtain more accurate prediction and better image quality, HEVC also allows using multiple reference frame (MRF) in ME module (MRF-ME), but the calculation is getting more and more complexity.

Recently, lots of paper propose hardware methods using graphic process unit (GPU) which developed by NVIDIA to parallel accelerate the huge calculation of MRF-ME module [3-7]. In these studies, Khemiri *et al.* [3] proposed a fast CTU-level parallel ME method to accelerate SAD and sum of square difference (SSD) calculation process through CPU and GPU collaboration based on functions repeated thousands of times called the SAD and SSD. They found that there is a same calculation item existing in the SAD and SSD for rate-distortion cost (RDcost) of ME and RDcost in mode decision (RDmode). Therefore, they proposed parallel difference (PD) and parallel reduction (PR) algorithm to accelerate SAD and SSD calculation process through CPU and GPU parallel architecture. But Khemiri*et al*. didn't consider that HEVC coding structure will call GPU kernel function most frequently by CPU and transfer data most frequently from CPU to GPU. This leads to their method occurs an obstacle to further speed up when HEVC takes MRF-ME module. On the other hand, Lin et al. [4] directly applied the fast ME algorithm default in H.264 test platform (JM) to H.265 video encoding. They utilized GPU to perform the computation and merging of SAD for different prediction unit (PU) mode. However, their method encounter that the shared memory is not enough in GPU when the range of searching window is larger than 32. And, it is inflexible for HEVC encoder due to frame-based structure.

In order to solve above-mentioned problem, we proposed a fast ME algorithm based on flexible CTU-level. The proposed method decomposes ME algorithm into three steps, including SAD calculation, SAD merging

and SAD comparing kernels, to achieve a highly parallel computation with a low external memory on GPU. On the other hand, the proposed method can be combined with fast CTU-level multiple reference frame motion estimation (MRF-ME) or fast mode decision algorithms to further reduce the encoding time.

II. BRIEF OVERVIEW OF MRF-ME MODULE AND GPU ARCHITECTURE

ME is a key module in the HEVC encoder. In other words, it is one of the critical and the most time-consuming module in video coding.ME is performed by using block matching algorithm (BMA) to find MV at the encoder and supports variable block sizes in HEVC. Generally, there are two kinds of search methods for ME to get a best matched predicted block. The first one is full search by checking all points in a SW, which is simple but time-consuming. The second one is fast motion search by checking several points in several iterations. Therefore, it is faster than the full search way and is more common used in software encoder. However, the full search is more suitable for hardware implementation or some heterogeneous computing (e.g. CPU+GPU) due to its regularity.

3.1 MRF-ME module

HEVC adopts some new coding structures including CU, PU and TU. The CU is the basic unit of region splitting used for inter/intra prediction, which allows recursive subdividing into four equally sized blocks. The CU can be split by coding quadtree structure of 4 level depths, which CU size ranges from largest CU size of 64×64 pixels to the smallest CU size of 8×8 pixels. At each depth level (CU size), HEVC performs ME with different size. In general, inter-coded CUs have eight PU types including symmetric blocks (2N×2N,2N×N, N×2N, N×N) and asymmetric blocks (2N×nU, 2N×nD, nL×2N, nR×2N) [1]. The RDcost needs be calculated by performing the PUs and TUs to select the optimal partition mode under all partition modes for each CU size. In the PU structure, HEVC adopts ME module to choose the optimal inter prediction mode. In order to improve the accuracy of PU prediction, MRF interframe prediction is performed in the ME module for HEVC.

Figure 1 shows the computational process for a CU to find the optimal partition mode from MRF-ME module in HEVC. Although the MRF-ME can enhance the PU performance and allow the encoder to search a better reference frame from several previous pictures, the computational complexity of the MRF-ME dramatically increases. Table 1 shows the complexity of MRF-ME in HM 16.7 [11] test platform using 3,840×2,160 video sequence and four reference frames. From Table1, we can find the total number of ME calculations required in MRF-ME is up to 27,450,240 times. Therefore, the very high computational complexity becomes a main bottleneck for the real-time applications of HEVC in UHD videos

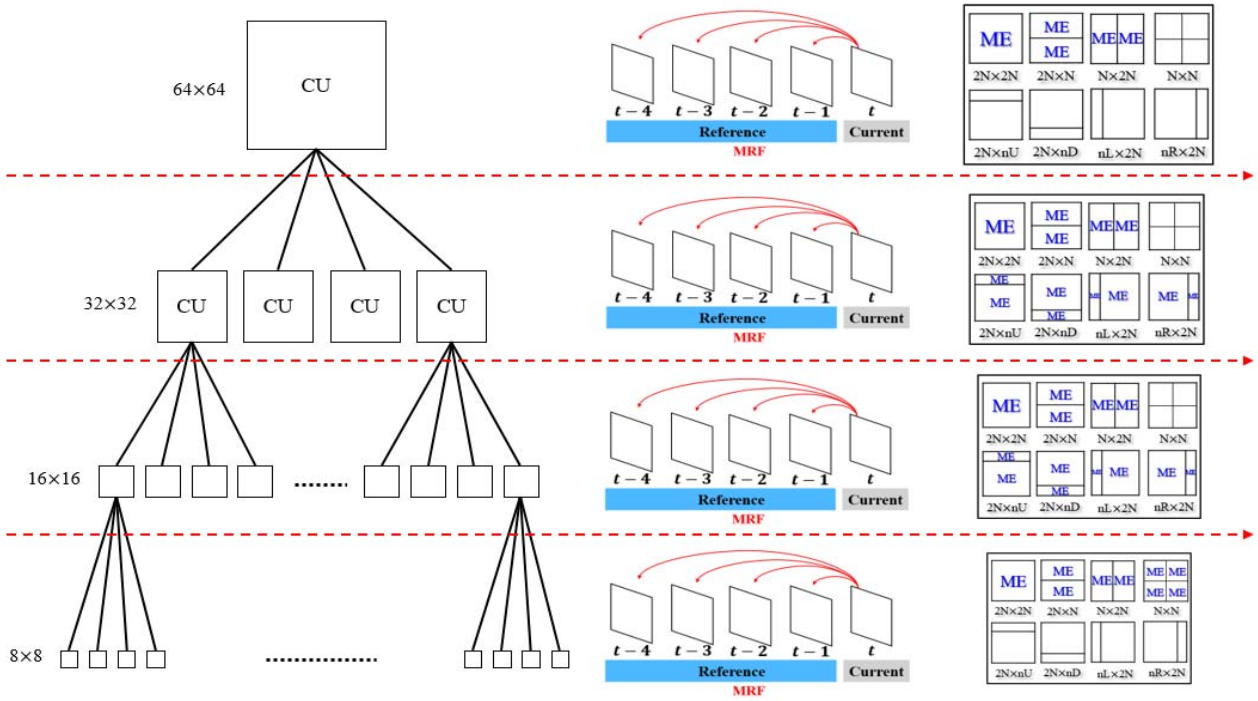

Fig. 1. MRF-ME module in HEVC.

Table 1. Complexity of MRF-ME in HM 16.7.

Depth	CU/CTU	modes/CU	MEs/modes	CTUs /frame	MRF (frame)	ME calculation times
0	1	3	5	8,160	4	163,200
1	4	7	13	8,160	4	1,697,280
2	16	7	13	8,160	4	6,789,120
3	64	4	9	8,160	4	18,800,640
Total calculations of MRF-ME						27,450,240

On the other hand, we can find that the ME module takes around 70% encoding time of inter frame prediction from the study report of [8], as shown in Fig. 2. Specially, inter prediction requires a high heavy computational burden in the entire HEVC encoding procedure, so it is a big challenge in the real-time ME by software encoder in CPU. Therefore, it is a very important issue how to speed up the searching process of ME module.

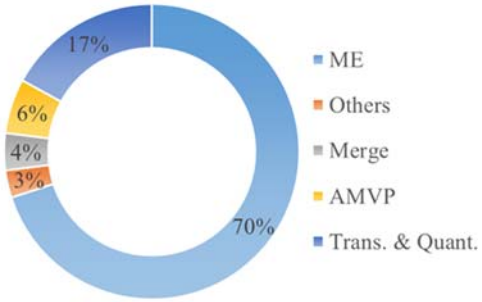

Fig. 2. Encoding time distribution of inter frame prediction.

3.2 GPU architecture

Nowadays, due to the rapid development of GPU processing capability, there has been a strong demand of using GPU as a co-processor to assist CPU to deal with data-intensive application [9]. GPU consists of hundreds of streaming process (SP) which is designed for achieving high performance. These SP are grouped into streaming multiprocessor (SM) in order to perform single instruction multiple data (SIMD) which is suitable for arithmetic intensive application. The SP offers special function unit for float-point and integer operations, the SM includes shared memory and register.

Fortunately, NVIDIA has announced a programming friendly GPU architecture called "Compute Unified Device Architecture" (CUDA) [10] to make massive data parallel processing easier. CUDA is an extension application with C/C++ languages that it can allows programmer to develop parallel process codes for GPU. In the GPU product, there are some architecture proposed by NVIDIA: Tesla, Fermi, Kepler, Maxwell, Pascal and Volta. Those architectures depend on different number of core compute capability and produce technique.

For a 64×64 CTU, the number of total combinations of ME calculations is very huge, as shown in Table 1. It is impossible to evaluate every possible CU and PU combination using RDO. This is the main reason why the HEVC encoding is so slow. Therefore, the cooperation mechanism for the CPU and GPU is a very efficient operation for hardware encoder of HEVC, as shown in Fig. 3. One CPU thread controls the GPU for data exchange, while the other CPU threads perform normal module is implemented in the GPU, and the GPU returns the best MV and the lowest MC cost of every possible PU to the CPU memory through PCI-Express port. When the CPU thread needs the ME data for the mode decision, it can get them directly from the memory and derive the PU and CU cost data to make the final decision. However, if the ME search is a full search for UHD resolution, even with a powerful GPU, the searching speed is still slow. Therefore, how to develop a ME paralleling computation algorithm on CPU plus GPU platform has been widely studied to realize real-time applications in HEVC. In this paper, we adopt the Pascal architecture to implement the proposed fast ME-MRF module.

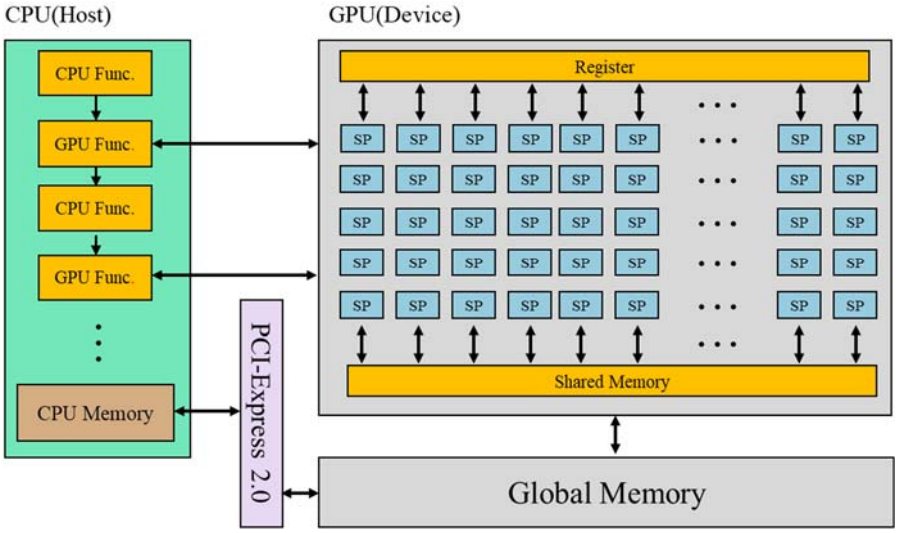

Fig. 3. CPU and GPU cooperation mechanism.

III. PROPOSED PARALLEL ME MODULE

In order to solve some problems including transfer data most frequently from CPU to GPU [3] and insufficient shared memory in GPU [4], we proposed a fast MRF-ME parallel algorithm based on flexible CTU-level. In the proposed algorithm, an image is divided into lots of CTU, and each CTU will call the GPU function to do 8×8 block SAD calculation, and then merge all different mode and find out minimum SAD for all mode. After finishing GPU parallel process, we will return minimum SAD and MV with all different mode to different depth CU. Figure 4 shows the proposed Fast CTU-level MRF-ME architecture on GPU.

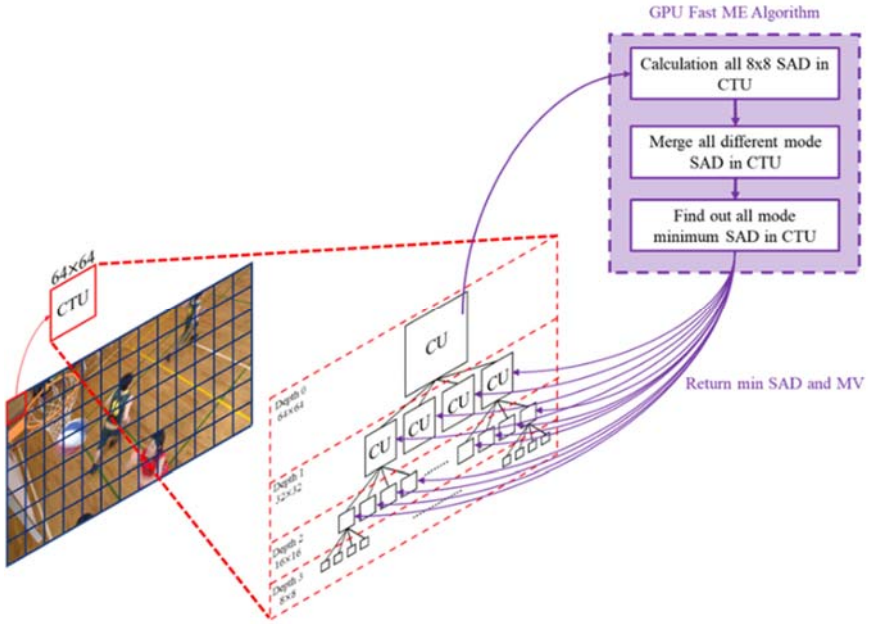

Fig. 4. Fast CTU-level MRF-ME architecture on GPU.

To further speed up the hardware encoder of HEVC, we decompose ME module into three kernels to achieve a highly parallel computation with a low external memory on GPU. The proposed three GPU kernel functions can be described as follows:

(1) Kernel 1 computes all 8×8 block SAD calculation in CTU
(2) Kernel 2 merges all different mode in CTU
(3) Kernel 3 finds out the minimum SADs for all mode

The flowchart of the proposed parallel computation is shown in Fig.5. At first, it needs to transfer reference frame pixel data and current frame pixel data from CPU to GPU's texture memory, and then start to launch Kernel 1, it will transfer all 8×8 block in CTU to shared memory in order to accelerate GPU computation time. After that it computes all 8×8 block's SAD within SW, and then saves all the SAD within SW to global memory. Secondly, launching Kernel 2 and fetching SAD which are obtained by Kernel 1, and then start to merge in 8 kinds of mode and save them to global memory. Thirdly, launching Kernel 3 and fetching the SAD which are merged by Kernel 2, and find out the minimum SAD from each mode and then save them to global memory. Finally, returning 8 kinds of minimum SAD with different mode to CPU. So, it will take a loop for all CTU by using the three kernel functions to perform parallel MRF-ME in HEVC structure of different reference frame.

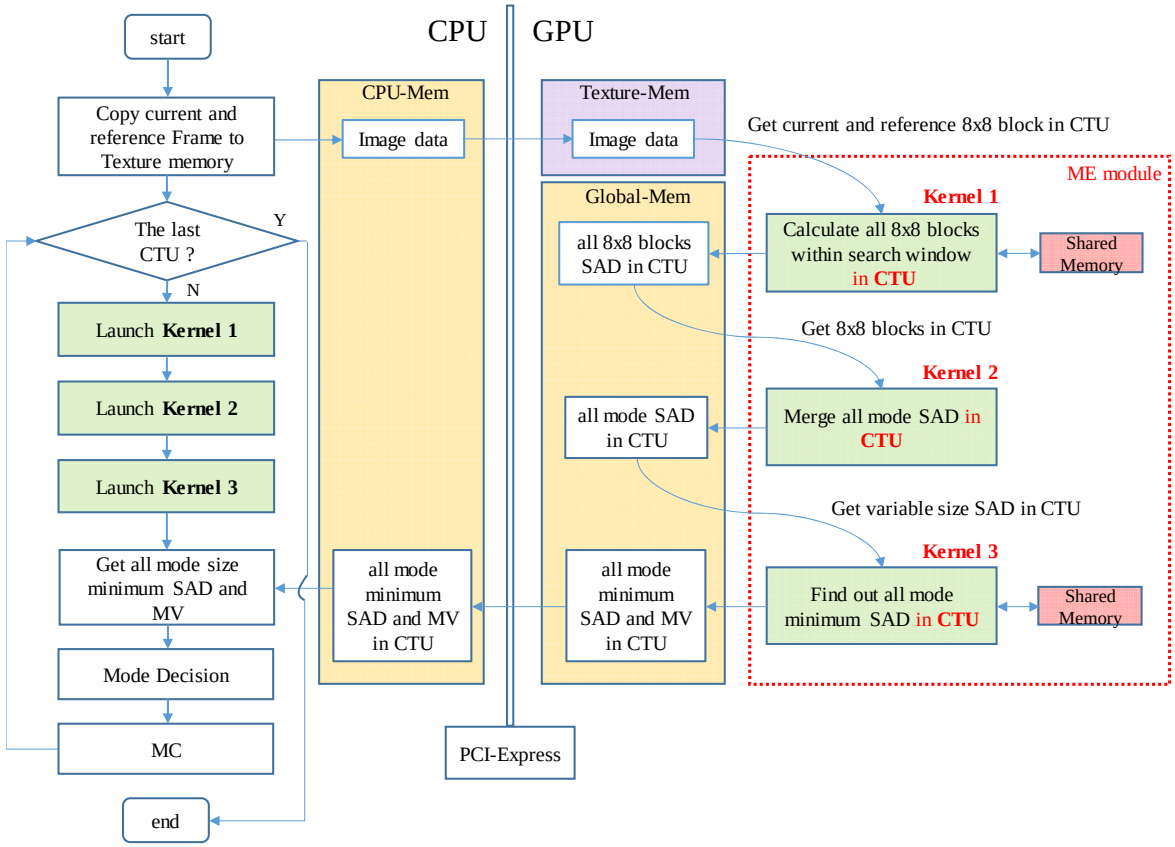

Fig. 5. The flowchart of proposed parallel ME algorithm.

IV. EXPERIMENTAL RESULTS

In this paper, we have implemented a CTU-level fast MRF-ME algorithm based on GPU in HM16.7[11] encoder test model, the encoding configuration is low delay P with (1) QP = 22, 27, 32, 37, (2) Max partition depth equals 3, (3) Fast search equals 0 (full search), (4) Motion search range equals 16 and 32, (5)MRF equals 4 and 8, (6) Intra period equals -1 (IPPPP...), (7) AMP equals 0, (8) Frame to be encoded equals 24 frames. In addition, the standard sequences are used as test frames including Traffic (2560×1600), Kimono (1920×1080) and ParkScene (1920×1080).

Simulations are conducted on a desktop with (1) Intel Core i5-3470, (2) NVIDIA GeForce GTX 1060-6GB, (3) Window 10-64bit, (4) CUDA Toolkit 9.1, (5) CUDA Compute Capability 6.1. The coding performance is evaluated based on speedup and ME time improving ratio (TIR), which are defined as follows:

$$Speedup = \frac{ME_{HM16.7}\ time}{ME_{method}\ time} \quad (1)$$

$$TIR = \frac{ME_{HM16.7} time - ME_{method}\ time}{ME_{HM16.7}\ time} \times 100\% \quad (2)$$

Table 2 and Table 3 show the accelerating performance obtained by Khemiri's [3], Lin's [4] and proposed method when performing 4 and 8 multiple reference frames (MRF=4 and MRF=8), respectively. Simulation results show that the proposed method can achieve an average speedup and TIR about 45.73% and 96.68% as MRF=4, and 44.81% and 97.76% as MRF=8, respectively. From Tables 2 and 3, we can observe that the proposed method far surpasses Khemiri's method in speedup and TIR using MRF=4 and MRF=8. In addition, we also can find that the accelerating performances of proposed algorithm are lower than those of Lin's method. However,

Lin's method will not be performed when the search window is larger than SW=32, this is because the shared memory is insufficient. On the other hand, our proposed fast MRF-ME module can operate smoothly at the same conditions.

Table 4compares the total global memory size (GMS) and shared memory size (SMS) are required by Lin's and proposed algorithm as SW=16. From Table4, we can find that the required GMS and SMS of our proposed is 1.85 MB and 4 KB, respectively. However, there are a very high memory which is used by Lin's method. It is obvious the required GMS and SMS in our method only occupy0.16% and 12.5% of Lin's, respectively. In addition, it is clear the proposed GPU-based fast ME algorithm can highly increase the speed of HEVC encoder with insignificant loss of image quality.

Table 2. The accelerating performance as MRF=4, SW=16.

Sequence	QP	Speedup			TIR(%)		
		Khemiri [3]	Lin [4]	Proposed	Khemiri [3]	Lin [4]	Proposed
Traffic	22	11.66	68.85	48.43	23.59	98.55	97.94
	27	11.61	66.79	48.16	23.61	98.50	97.92
	32	11.81	66.78	48.96	23.62	98.50	97.96
	37	12.55	74.12	52.03	23.65	98.65	98.08
Kimono	22	10.37	51.84	42.99	23.11	98.07	95.81
	27	10.16	50.29	42.11	23.10	98.01	95.74
	32	10.55	50.28	43.73	23.11	98.01	95.80
	37	9.89	55.81	40.98	23.18	98.21	96.08
ParkScene	22	11.70	56.76	45.92	24.49	98.24	96.15
	27	11.47	55.06	44.98	24.51	98.18	96.09
	32	11.91	55.06	46.71	24.52	98.18	96.14
	37	11.91	61.11	43.77	24.56	98.36	96.40
Average		11.24	59.40	45.73	23.75	98.29	96.68

Table 3. The accelerating performance as MRF=8, SW=32.

Sequence	QP	Speedup			TIR(%)		
		Khemiri [3]	Lin [4]	Proposed	Khemiri [3]	Lin [4]	Proposed
Traffic	22	10.84	64.01	47.21	23.58	99.17	97.88
	27	10.89	62.62	46.25	23.59	99.14	97.84
	32	11.59	65.51	48.02	23.61	99.16	97.92
	37	10.31	60.92	45.00	23.58	99.14	97.78
Kimono	22	10.16	50.81	42.99	23.56	98.03	97.67
	27	10.04	49.71	42.11	23.56	97.99	97.63
	32	10.91	52.00	43.73	23.57	98.08	97.71
	37	8.57	48.36	40.98	23.54	97.93	97.56
ParkScene	22	11.25	54.56	45.92	24.92	98.17	97.82
	27	11.12	53.38	44.98	24.94	98.13	97.78
	32	12.08	55.84	46.71	24.96	98.21	97.86
	37	10.12	51.93	43.77	24.90	98.07	97.72
Average		10.66	55.80	44.81	24.02	98.44	97.76

Table 4. Compare GMS and SMS required by Lin's and proposed algorithm.

Method / Memory	Lin [4]	Proposed
total GMS	1.1 GB	1.85 MB
total SMS	32KB	4KB

V. CONCLUSION

In this paper, we proposed a fast CTU-level MRF-ME algorithm based on GPU to reduce the calculation complexity of HEVC encoder. Performance evaluations show that the proposed method can efficiently reduce computation time and calculation complexity in MRF-ME module of HEVC. The TIR is 96.68% and 97.76% when MRF=4 and MRF=8, separately. In addition, the proposed method is a flexible CTU-level parallel ME by CPU and GPU pipeline. Therefore, we can combine any fast CTU-level ME or fast mode decision to further reduce the encoding time of HEVC.

REFERENCES

[1]. T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560-576, July 2003.

[2]. High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2, Jan. 2013.

[3]. R. Khemiri, H. Kibeya, F. E. Sayadi, N. Bahri, M. Atri, and N. Masmoudi, "Optimization of HEVC motion estimation exploiting SAD and SSD GPU-based implementation," *IET Image Process Institution of Engineering and Technology*, vol. 12, lss. 2, pp. 243-253, Jan. 2018.

[4]. Y. C. Lin, and S. C. Wu, "Parallel motion estimation and GPU-based fast coding unit splitting mechanism for HEVC," *IEEE High Performance Extreme Computing Conference*, pp. 1-7, Dec. 2016.

[5]. X.W. Wang, et al., "Paralleling variable block size motion estimation of HEVC on multi-core CPU plus GPU platform," *IEEE International Conference on Image Processing*, Melbourne, VIC, pp. 1836-1839, Sept. 2013.

[6]. D. Lee, D. Sim, and S.J. Oh, "Integer-pel motion estimation for HEVC on compute unified device architecture (CUDA)," *IEEE Transactions on Smart Processing and Computing*, vol. 3, no. 6, pp. 397-403, Dec. 2014.

[7]. S. Radicke, J.-U Hahn, Q. Wang, and C. Grecos, "Bi-predictive motion estimation for HEVC on a graphics processing unit (GPU)," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 4, pp. 728-736, Feb. 2015.

[8]. J. Kim, D.S. Jun, S. Jeong, et al. "A SAD-based selective bi-prediction method for fast motion estimation in high efficiency video coding," Electronics and Telecommunication Research Institute Journal, vol. 34, no. 5, pp. 753–758, Oct. 2012.

[9]. GPGPU. [Online]. Available: http://www.gpgpu.org/.

[10]. NVIDIA (2018). CUDA Toolkit Documentation [v9.2], https://docs.nvidia.com/cuda/

[11]. HEVC Test Model documentation [HM16.7]. https://hevc.hhi.fraunhofer.de/HM-doc/

Chou-Chen Wang, "Fast Motion Estimation For High Efficiency Video Coding Based On A Graphics Processing Unit " The International Journal of Engineering and Science (IJES), 7.11 (2018): 70-76