A parallel HEVC encoder scheme based on Multi-core platform

Shu Jun[1,2,3,a], Hu Dong[1,2,3,b]

[1]Education Ministry's Key Lab of Broadband Wireless Communication and Sensor Network Technology

[2]Education Ministry's Engineering Research Center of Ubiquitous Network and Health Service System

[3]Jiangsu Province's Key Lab of Image Procession and Image Communications, Nanjing University of Post and Telecommunications, Nanjing, 210003, China

[a]email: 1013010606@njupt.edu.cn, [b]email: hud@njupt.edu.cn

Keywords: HEVC; multi-core platform; parallel processing; frame-level; CTB-level

Abstract. In this paper, we propose a parallel HEVC encoder scheme based on multi-core platform, which provides maximized parallel scalability by exploiting two-level parallelism, namely, the frame level parallelism and the CTB level parallelism. Inspired by the intra-CTB row level parallelism of WPP in HEVC, we investigate the inter-frame CTB prediction dependency to its reference CTBs, and find the inter-CTB correlation. Using this inter-correlation, we divide a frame into CTB units and create CTB-row level coding threads when their corresponding reference CTBs are available. Each thread is bonded to a processing core, therefore, both intra- and inter-CTB rows can be encoded in parallel. Moreover, we introduce a priority scheduling mechanism to control the coding threads. Experiments on Tilera-Gx36 multi-core platform show that, compared with serial execution, the proposed method achieves 3.6 and 4.3 times speedup for 1080P and 720P video sequences, respectively.

1. Introduction

Recent increasing demands on video coding support for higher resolutions in consumer devices are driving the video coding development to higher compression rates. To meet these demands, the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T and ISO/IEC Moving Pictures Experts Group has developed a new video coding standard, the High Efficiency Video Coding (HEVC) [1]. The HEVC project aims at reducing the bitrate compared to the previous H.264/AVC by another 50%. However, the cost paid for higher coding efficiency is much higher computational complexity. Then, the HEVC encoders are expected to be more complex than the H.264/AVC encoders [2], and parallelism has to be considered in real-time HEVC encoding. With the development of multi-core Digital Signal Processor (DSP) platform, parallelizing HEVC encoding on such platforms is of extreme importance to deal with this problem [3].

In the video coding layer of HEVC, the same hybrid approach (intra-/inter- picture prediction and 2-D transform coding) as that of H.264/AVC [4] is employed. Fig.1 depicts the block diagram of a hybrid video encoder, which could create a bitstream conforming to the HEVC standard. As Fig.1 shows the main modules in HEVC encoder includes intra-/inter-prediction, transformation and quantization, inverse transformation and inverse quantization, entropy coding and loop filtering, etc. Compared with H.264/AVC, the complexity of HEVC is mainly reflected as follows: In HEVC the motion compensation uses the same quarter pixel motion resolution, but the derivation of interpolated pixels is generalized by using a larger 8-tap and 4-tap interpolation filter for luma chroma respectively. Intra-prediction is generalized as well by parameterization of the predicted angle, allowing 33 possible different angles. The transform is still an integer transform but allows more block sizes, ranging from 4×4 to 32×32 with higher internal processing precision. HEVC also defines a more efficient block structure, called Coding Tree Block (CTB). The sequence is coded by CTB of size 16×16, 32×32, or 64×64 pixels. Each CTB can be recursively subdivided using quad-tree segmentation in coding units (CUs), which can in turn be further subdivided into prediction units (PUs) and transform units (TUs). Coding units can be subdivided down to a minimum CU of size 8×8. The minimum prediction unit size is 4×8 and 8×4, and minimum TU size is 4×4 pixels [5].

As we can see, the computational complexity of HEVC encoding is huge. It is unlikely that single core processor can encode a 1080P or higher resolution HEVC video in real-time. This paper will present a new parallel scheme on the multi-core platform. In particular, our contributions can be summarized as follows:

1) Inspired by Wavefront Parallel Processing (WPP) of HEVC and the intra-CTB row level parallelism, we investigate the inter-frame CTB prediction dependency to its reference CTB, and find the inter-CTB correlation.

2) Based on intra-CTB and inter-CTB correlation, we divide a frame into CTB units and create CTB-row level coding threads when their corresponding reference CTBs are available. Each thread is bonding to a single processing core, so intra- and inter-CTB rows can be encoded in parallel. Moreover, we introduce a priority scheduling mechanism to control these coding threads. Hence, the frames and CTBs are processed in parallel, both frame-level and CTB-row-level parallelism are realized.

3) We test the proposed parallel encoding scheme on a multi-core platform, the Tilera-Gx36 system with 36 cores running at 1.2GHz, for 1080P and 720P video sequence respectively. We compare the proposed approach with serial execution of x265 reference software both in terms of speedup and PSNR to prove the efficiency of the proposed parallel scheme.

The rest of this paper is organized as follows. Section 2 introduces the parallelization strategies of HEVC and analyzes the dependencies of inter-frame CTBs. In Section 3, the proposed parallel scheme is described in details. Experimental results and their analysis are presented in Section 4, followed by a short conclusion in section 5.

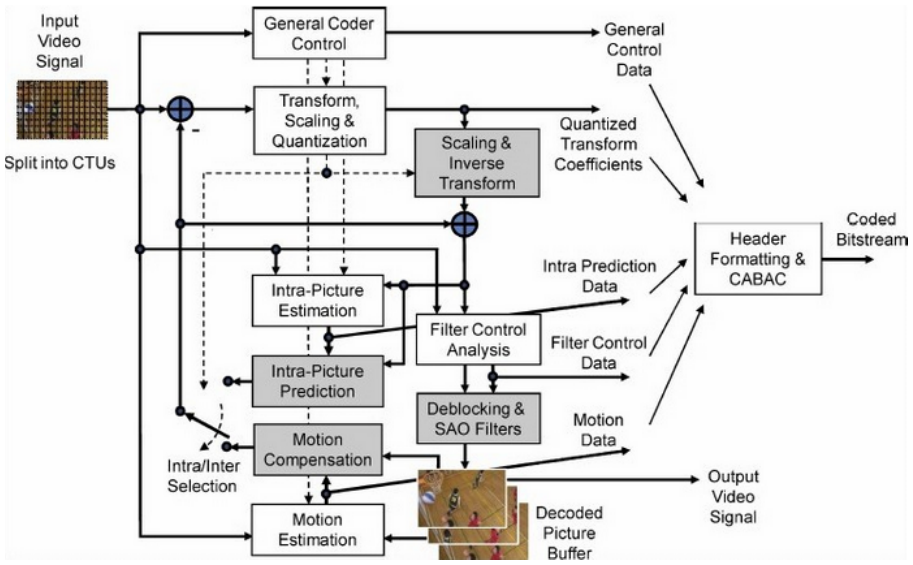

Fig.1. General diagram of the HEVC encoder

2. Parallelization Strategies in HEVC

The current HEVC standard contains several strategies aiming at better parallel processing. In H.264/AVC, there are frame-level, slice-level or macroblock-level parallelism [6]. Take slice-level parallelism for example, a picture can be partitioned in multiple arbitrarily sized slices for independent processing, having multiple slices in a picture, however, degrades objective and subjective quality due to slice boundary discontinuities and increases significant coding losses. In order to overcome the shortage of the parallelization strategies employed in H.264/AVC, two tools have been included in the HEVC standard: Wavefront Parallel Processing (WPP) and Tiles. Both of these tools allow subdivision of each picture into multiple partitions that can be processed in parallel.

2.1 Wavefront Parallel Processing

When WPP is enabled, a picture is divided into several CTB-rows and every row can be assigned to a core [7]. The first row is processed in an ordinary way, the second row can begin to be processed after two CTBs have been encoded in the first row, the third row can begin to be processed after two CTBs have been encoded in the second row, etc. Compared to slices, no coding dependences are broken at row boundaries. Additionally, CABAC probabilities are propagated from the second CTB of the previous row, to further reduce the coding losses (Fig.2). Also, WPP does not change the regular raster scan order. Furthermore, a WPP bitstream can be losslessly transcoded to/from a nonparallel bitstream with only an entropy-level conversion [8].

2.2 Investigation on Inter-Frame Dependency

Although WPP makes the intra-frame parallelism easier, the parallelism is limited. Due to the intra-frame dependency, WPP does not allow all the rows to start being encoded simultaneously, so rows cannot be finished at the same time, which will make parallelization inefficiency.

As is discussed above, the wavefront parallelism still can be improved. To maximize parallel scalability, we combine inter-frame level parallelism with intra-CTB row level parallelism. So analyzing the CTB dependency to its reference CTBs inter-mode frame, the inter-frame dependency, is necessary.

In order to improve the encoding performance, HEVC requires the reconstructed frames as reference pictures to deal with time redundancy, thus achieving inter-CTB row level parallelism must consider inter-frame dependency. Before a CTB in current frame can be encoded, all of its reference CTBs in the searching range must be available. Inter-CTB row level parallel approach is shown in Fig.3, it can not immediately be encoded when the first row in Image 0 has been encoded, because the motion estimation search range is larger than a CTB-row. In order to predict accurately, we have to wait the coding unit being reconstructed. If the motion estimation search range is twice as big as a CTB unit, the first CTB in upper left corner of image 1 can not be encoded unless the third row in image 0 is reconstructed [9].

As shown in Fig.3, the current coding CTB in Image 1 can be encoded with the CTB in Image 0 simultaneously. Therefore, as long as there no inter-frame CTB prediction dependency exists, some CTBs in Image 1 and Image 0 can be encoded simultaneously. By introducing the inter-parallel, we improve the parallel speedup significantly.

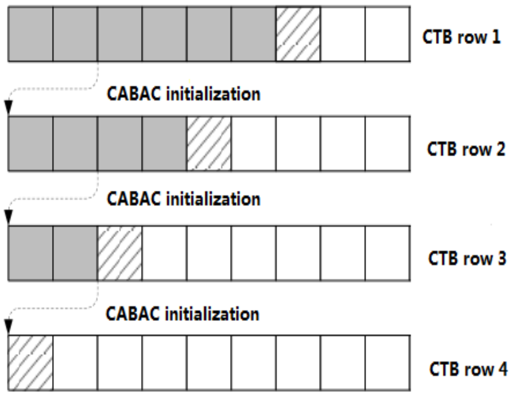

Fig.2 WPP processes rows of CTBs in parallel

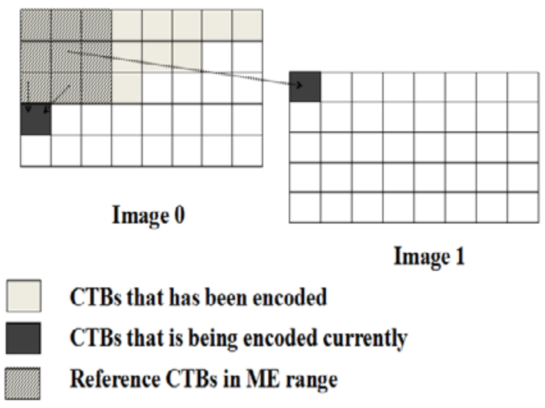
Image 0
Image 1
CTBs that has been encoded
CTBs that is being encoded currently
Reference CTBs in ME range

Fig.3 inter-CTB level parallelism

3. Proposed Parallel Encoding Method

According to the data dependence analysis of the inter-frame parallelism, this section we will present the proposed parallel method, we divide our work into two parts, the first part describes the proposed method in details and the second part, the priority scheduling mechanism, is presented to control the coding threads, which is essential to our scheme.

3.1 Combine intra-CTB row Level Parallelism With Inter-frame Parallelism

Our proposed scheme combines intra-CTB row level parallel processing with inter-frame parallel processing. Compared to single granularity, multi-granularity division can achieve higher speedup. In fact, the inter-frame level parallelism is not at the level of the frame, but calls inter-CTB row level parallelism to complete the final coding process.

The proposed scheme is based on multi-core platform. Using the parallel programming technology like task pool and thread pool, we assign tasks to frames that can be coded, each task has a separate memory space to store parameter information of each frame, a plurality of frame-level tasks share a common thread pool. Therefore, several frames can be encoded in parallel, which called inter-frame level parallelism. Then according to the received parameters, we create multiple CTB-row level tasks which call threads to complete the coding, and bind each thread to the corresponding CPU core. The proposed scheme further uses homogeneous multi-core platform's shared memory model to achieve multi-thread synchronization. It can parallel encoding intra- and inter-CTB rows only when the dependencies are eliminated, by checking intra- and inter-dependency flag, we can achieve CTB-row level parallelism. The specific process is shown in Fig.4.

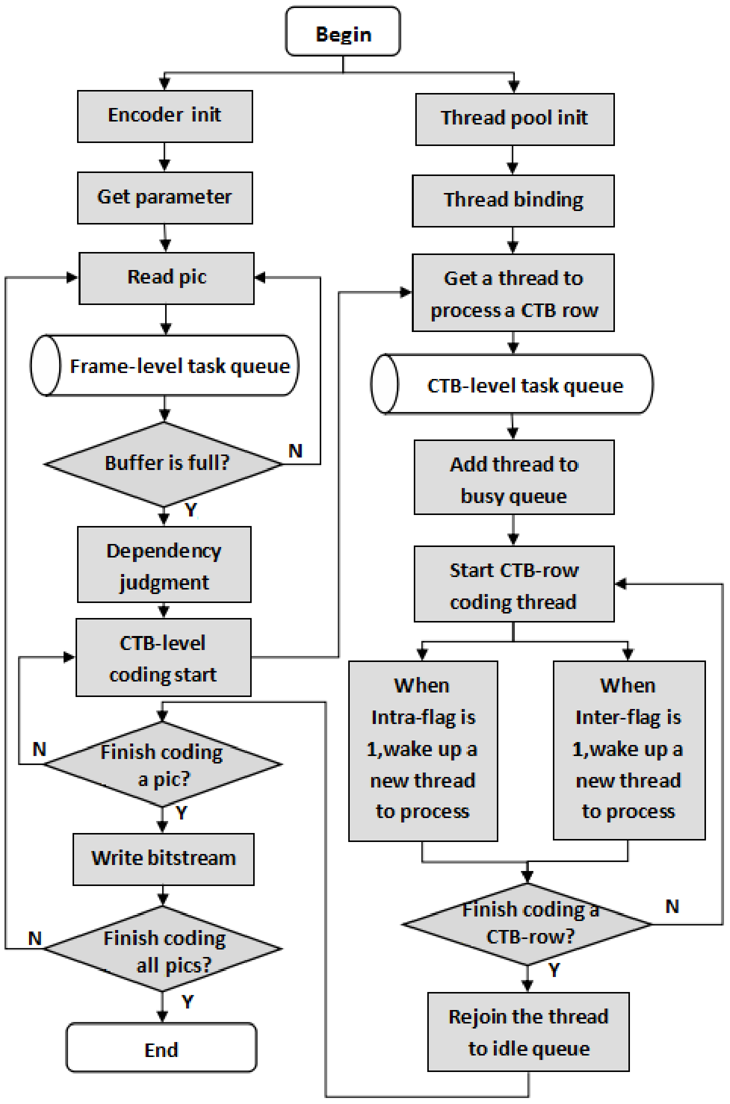

Fig.4 the scheme of combine intra-CTB level parallelism with inter-frame level parallelism

3.2 Priority Scheduling Mechanism

As Fig.4 shows to us, two priority queue structures are introduced to implement the proposed scheme. In the main coding thread, first reading a picture, adding it to frame-level task queue and then continue reading frames if the queue is not full. When the buffer is full, by using task pool technology, we assign tasks to frames that can be encoded in the queue, each task has a separate memory space to store parameter information of each frame, a plurality of frame-level tasks share a common thread pool. But in fact, the inter-frame level parallelism is not at the level of the frame, but by calling inter-CTB row unit to complete the final coding process.

In this paper, as the smallest parallel granularity, a CTB-row calls an idle thread in thread pool to encode it and add it to CTB-row level task queue. To determine the CTB-row coding order, we define two-level priority of CTB-row task queue. The first one is the inter-level and the second is intra-level, inter-level of priority level is higher than intra-level, that means if both inter-CTB row and intra-CTB row are well prepared be encoded, the inter-CTB row enter the queue first. Specifically, the inter-level of priority specifies while in the task queue, if several CTB-rows in different frames are ready to be encoded, the CTB-row which has the smallest frame number in frame-level task queue joins the queue first, similarly, the intra-level specifies if several CTB-rows in the same frame are prepared be encoded, the CTB-row with the smallest line number adds to the queue first.

Two-level task queue is depicted in Fig.5, each CTB-row task in the CTB-row level task queue will call an idle thread to encode it, until there are no threads available in thread pool, then the CTB-row task in the task queue have to wait for a new idle thread, once there is a CTB-row finish being encoded, the coding thread rejoins the thread pool for other CTB row-level task calls. It is worth noting that the CTB-row coding work is carried out serial, one CTB by one CTB. For CTB-rows in I frame, multiple CTB-rows can be parallel encoded when intra-frame data dependency eliminates, and for CTB-rows in non-I frames, we need to consider inter-frame data dependency, due to the need of reconstructed pixels to deal with time redundancy, therefore, only the corresponding CTB-rows in reference region are reconstructed, can CTB-row thread be opened to achieve intra- and inter-CTB parallel processing.

4. Experimental results

In order to compare our proposed parallel method with serial execution, we adopt HEVC reference software x265, which supports WPP and includes all feature of the main profile [10]. The experimental platform is Tilera-Gx36, which is a member of TILERA multi-core processor family with 36 cores. In order to avoid the impact of special platform, we do not use any Tilera-Gx36 platform-dependent optimizations. The test sequences are Kimono of 1080P resolution and Fourpeople of 720P resolution. In our work, we implement the proposed parallel method on x265, CTU size set 32×32, so there are more CTB-rows to be parallel encoded, QP unified set 27, similar results are observed for other QPs. More detail experimental environments and conditions are written in Table 1.

Table 1. Experimental Platform and Test Conditions

Processor	TILE-Gx8036
Architecture	TILE-Gx
Number of cores	36
Frequency (single core)	1.2GHz
Operating system	Tilera MDE-4.0.3.1415127
Compiler	GCC 4.6.3
Test sequences	Kimono , Fourpeople
Reference software	x265
Encoding Conditions	QP: 27

To evaluate the efficiency of our proposed parallel method, we use four index to compare our proposed method to original method in x265. The fps denotes coding rate, the PSNR and Bitrate are used to measure the change of picture quality and the speedup of our proposed method can be calculated as follows:

$$Speedup = \frac{T_{serial}}{T_{proposed}} \tag{1}$$

Where T_{serial} and $T_{proposed}$ are respectively the coding time of serial execution and our proposed method. Table 2 shows the parallel performance of our proposed parallel method compare to serial execution. And Fig.6 shows the speedup of two sequences.

Table 2. Experimental Results of Our Proposed Parallel Method

Class	Threads number (cores)	Original method			Proposed method		
		fps	PSNR (dB)	Bitrate(kbps)	fps	PSNR (dB)	Bitrate(kbps)
Kimono (1080P)	2	2.65	38.403	1771.60	3.03	38.394	1792.63
	4				4.44	38.394	1792.63
	8				6.31	38.394	1792.63
	16				7.84	38.394	1792.63
	24				8.52	38.394	1792.63
	32				9.46	38.394	1792.63
	36				9.31	38.394	1792.63
Fourpeople (720P)	2	3.94	27.870	25006.86	5.76	30.112	25117.76
	4				8.32	27.857	25117.76
	8				10.71	27.857	25117.76
	16				13.55	27.857	25117.76
	24				15.01	27.857	25117.76
	32				16.75	27.857	25117.76
	36				16.69	27.857	25117.76

From Fig.6 and Table 2, we get three major observations:

1) Compare to original method in x265, our proposed method has a little change in PSNR and Bitrate, this may results from parallel processing more CTB-rows, weaken the relevance of inter-frames, so that the image quality has declined.

2) The speedup of our proposed method behaves good when less than 32 cores but the upward trend gradually become flat from this point, this result may due to inter-core synchronization and communication costs.

3) Compare with serial execution, our proposed method achieves 3.6 and 4.3 times speedup for 1080P and 720P video sequences, respectively.

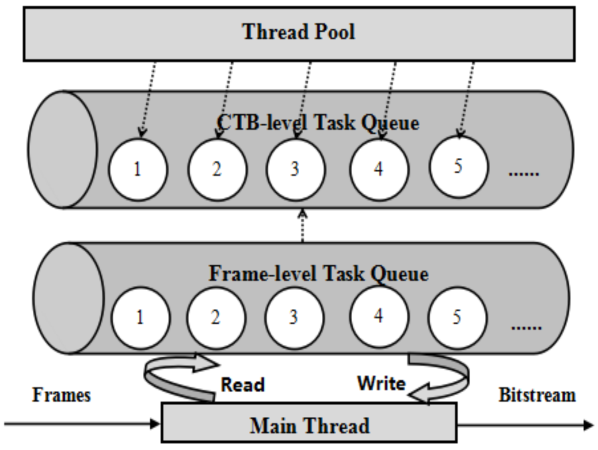

Fig.5 the two-level task queue model

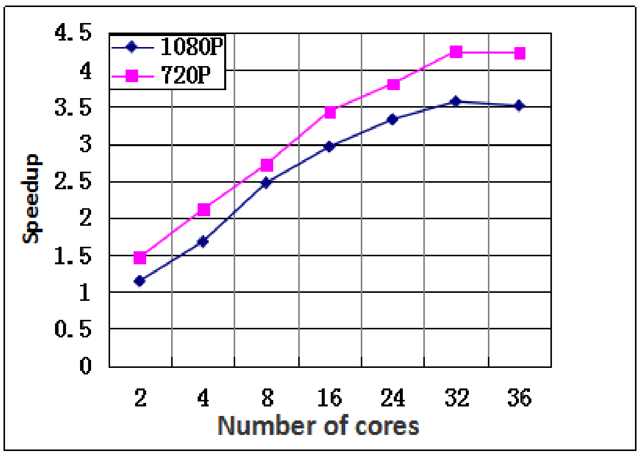

Fig.6 Speedup of proposed method using different number of cores

5. Conclusion

In this paper, in order to improve the speedup of HEVC encoder, a parallel scheme based on multi-core platform is proposed. On the basis of intra-frame CTB-row level parallelism, we exploit inter-frame parallelism to achieve intra- and inter-CTB multigrain parallelization. Meanwhile, we introduce a priority scheduling mechanism to control these coding threads. Experimental results shows the new scheme improve the parallel speedup with a little change in PSNR and Bitrate, but we only test CTU size 32×32 and do not consider the effect of other CTU size like 64×64. How to choose suitable CTU size of different video sequences is the direction of our future research.

References

[1] G.J.Sullivan and J.-R.Ohm. Recent developments in standardization of high efficiency video coding (HEVC). Proc.SPIE, Aug.2010, p.77980V.

[2] F.Bossen, B.Bross, K.Sühring and D.Flynn. HEVC complexity and implementation analysis. IEEE Trans.Circuits Syst.Video Technol, vol.22, no.12, pp.1684-1695, Dec.2012.

[3] S.Borkar and A.A.Chien. The future of microprocessors. Commun.ACM, vol.54, pp.67-77, May 2011.

[4] G.J.Sullivan, J.-R.Ohm, W.-J.Han and T.Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. IEEE Trans.Circuits Syst.Video Technol, vol.22, no.12, pp.1648-1667, Dec.2012.

[5] C.C.Chi, M.Alvarez-Mesa, B.Juurlink, G.Clare, F.Henry, S.Pateux, and T.Schierl. Parallel scalability and efficiency of HEVC parallelization approaches. IEEE Trans.Circuits Syst.Video Technol, vol.22, no.12, pp.1826-1837, Dec.2012.

[6] B.Juurlink, M.Alvarez-Mesa, C.C.Chi, A.Azevedo, C.Meenderinck and A.Ramirez. Scalable Parallel Programming Applied to H.264/AVC Decoding. Berlin, Germany: Springer, 2012.

[7] F.Henry and S.Pateux. Wavefront parallel processing. Tech.Rep.JCTVC-E196, Mar.2011.

[8] G.Clare and F.Henry. An HEVC transcoder converting non-parallel bitstreams to/from WPP. Tech.Rep.JCTVC-J0032, May 2012.

[9] WEI Fei-fei, LIANG Jiu-zhen, HAN Jun. A Parallel X264 Encoder Algorithm Based on the Inter-Frame and Intra-Frame Macroblock-Level. Computer Engineering & Science, vol.33, No.7, 2011.

[10] x265 project, multicoreware, https://bitbucket.org/multicoreware/x265/src/e7424e0cb60f4bb08 e7d519a49ff9ab77d6fe713/source/common/vec/dct-sse3.cpp? at=default.