Tile-level and Frame-level Parallel Encoding for HEVC

Younhee Kim², Jinwuk Seok³, Soon-heung Jung⁴, Huiyong Kim⁵, and Jin Soo Choi⁶

Abstract

High Efficiency Video Coding (HEVC)/H.265 is a new video coding standard which is known as high compression ratio compared to the previous standard, Advanced Video Coding (AVC)/H.264. Due to achievement of high efficiency, HEVC sacrifices the time complexity. To apply HEVC to the market applications, one of the key requirements is the fast encoding. To achieve the fast encoding, exploiting thread-level parallelism is widely chosen mechanism since multi-threading is commonly supported based on the multi-core computer architecture. In this paper, we implement both the Tile-level parallelism and the Frame-level parallelism for HEVC encoding on multi-core platform. Based on the implementation, we present two approaches in combining the Tile-level parallelism with Frame-level parallelism. The first approach creates the fixed number of tile per frame while the second approach creates the number of tile per frame adaptively according to the number of frame in parallel and the number of available worker threads. Experimental results show that both improves the parallel scalability compared to the one that use only tile-level parallelism and the second approach achieves good trade-off between parallel scalability and coding efficiency for both Full-HD (1080 x 1920) and 4K UHD (3840 x 2160) sequences.

Keyword: HEVC encoding, tile-level parallelism, frame-level parallelism, Real-time encoder
I. Introduction

High Efficiency Video Coding (HEVC)\textsuperscript{1,2} is a new video codec which was finalized in January 2013. HEVC is known to provide two times higher compression ratio than the previous video coding standard, which is promising to the multimedia industry. As the electronics industry pushes the large display products and consumers pursue the high resolution video contents, broadcast community and other key players in the video market have taken the fast action to applying the HEVC codec to deliver advanced services. Due to achievement of high efficiency, HEVC sacrifices the time complexity. To apply HEVC to the market applications, one of the key requirements is the fast encoding\textsuperscript{3}. To achieve the fast encoding, exploiting thread-level parallelism is widely chosen mechanism since multi-threading is commonly supported based on the multi-core computer architecture. In HEVC, there are several picture partition schemes such as slices, tiles\textsuperscript{4}, and wavefront parallel processing (WPP)\textsuperscript{5,6,7}. Since the slices result larger coding loss compared to tiles, we consider the tile scheme for picture partitioning parallelism in this paper.

1. Tile-level parallelism

In HEVC the picture partitioning parallelism structure is supported as called Tile; a picture is divided into squares and encoded independently. There is no dependency between the tiles so that the several tiles can be encoded in parallel. However, the neighboring blocks at the tile boundary cannot be referenced while encoding, the encoding efficiency is decreased. We have implemented the Tile-level parallelism based on HEVC test model (HM) encoder. Since the HM encoder is single-core codec, the multiple tiles are encoded in serial and use a single CABAC engine. We parallelize multiple tile encoding with each independent CABAC engine.

Fig. 1. HEVC random access GOP structure proposed in [10]

2. Frame-level parallelism

Figure 1 illustrates the random access GOP structure defined in [10]. With the GOP structure, the frames in the fourth GOP level can be employed to the frame-level parallelism. The GOP structure shown in Fig. 1 has a shortage, which is that only GOP level 4 frames can be evolved in parallel processing. To improve the frame-level parallel scalability, we propose to change the GOP structure as Fig. 2, where the frames in the third GOP level are not referenced each other. In this way, we can encode two frames in the GOP level 3 in par-
parallel and then four frames in the GOP level 4 in parallel. We call this frame-level parallelism scheme Frame-level parallelism of GOP level 3&4 and Fig. 3 illustrates the scheme.

For the previous video coding standard, frame-level parallelism, slice-level parallelism, and combining approach of two different level parallelisms have been proposed. Since the tile is a newly adopted tool to support the parallel HEVC encoding, we have tried to combine the tile-level parallelism and frame-level parallelism in HEVC encoding. To the best of our knowledge, this is the first report of combining the tile-level parallelism and frame-level parallelism for HEVC. In this paper, we implemented tile-level parallel encoding\cite{8, 9} and frame-level parallel encoding each and proposed an effective combined approach applying the adaptive number of tile as taking the consideration of the number of frames in parallel and the number of available cores.

II. Improvement of parallelism

HEVC allows the tile number up to 25 for full HD video (1920 x 1080) and 110 for 4K UHD video (3840 x 2160). The recent multi-core architecture technique has improved significantly so that many systems have high number of cores.
Employing more than two parallelism schemes improves the parallel scalability. Based on the implementation of tile-level parallelism and frame-level parallelism, we combine the two parallelisms to improve the parallel scalability and to more effectively utilize the multi-core system.

1. Combined Approach 1: Fixed number of tile in a frame

First, we combine the tile-level parallelism and frame-level parallelism as shown in Fig. 4. As an example, we set the number of tile in a frame is four. When the encoder takes the input frame, it creates the thread as many as the number of frames in parallel, and then each frame-encoding-thread creates the worker thread as many as the number of tile.

The combined approach 1 illustrated in Fig. 4 has some shortage. When the frame-level parallelism is employed, the number of tiles in parallel is increased by factor of the number of frames in parallel. The unbalanced number of worker threads through the encoding timeline results in difficulty in

그림 4. 조합 1 방식: 타일 레벨 병렬화와 프레임 레벨 병렬화를 조합할 때 프레임 레벨 병렬화가 상관없이 동일한 개수의 타일로 화면을 분할하여 타일 레벨 병렬화를 수행하는 방법
Fig 4. Combined Approach 1: Tile-level parallelism combined with the frame-level parallelism of GOP level 3 & 4. The number of tile is the same for all frames. The number of tile in a frame is four as an example

그림 5. 조합 2 방식: 타일 레벨 병렬화와 프레임 레벨 병렬화를 조합할 때 병렬로 처리되는 프레임의 개수에 따라 적응적으로 타일의 개수를 조정하여 화면 분할 병렬화를 수행하는 방법
Fig. 5. Combined Approach 2: Tile-level parallelism combined with the frame-level parallelism of GOP level 3 & 4. The number of tile is changed taking the frame-level parallelism into consideration. The initial number of tile in a frame is four as an example
effectively running multi-threading.

2. Combined Approach 2: Adaptive number of tile in a frame

As targeting the problem in the previous section, we propose to apply the adaptive number of tile as considering the number of frames in parallel. The proposed method as taking an example with four tiles in a frame in serial as illustrated in Fig. 5. With the second approach having the adaptive number of tile, we expect less loss in coding efficiency and good balanced CPU usage.

III. Experimental Results

1. Test sequences and environments

We implemented our parallel encoding approaches described so far based on HEVC reference software. Multi-threading has been applied using Windows threads APIs. The test sequences used in the experiment are two set of video. The first set has five Full-HD (1920 x 1080) videos of 100 frames (Kimono, Park Scene, Cactus, Basketball Drive, and BQ Terrace), which are from HEVC test sequences. The second set has five 4K UHD (3840 x 2160) videos of 100 frames (Jockey, YachtRide, ReadySteadyGo, ShakeNDry, and HoneyBee), which are from Kvazaar Encoder[11, 12] test sequences. We used the separate platforms according to the test sequence resolution. The platform used for Full-HD test set has one Intel Xeon E5-2690 processor with eight physical cores whereas the platform for 4K UHD test set has two Intel Xeon E5-2690 processors with sixteen physical cores in total. For speedup measurement, the sequences encoded with one tile per frame in serial are used as the anchor. We select Main profile and HEVC common condition random access setting with two modifications: The GOP structure is changed as Fig. 2 to encode the frames in the level 3 in parallel and the encoding tool of AMP is not applied.

2. Coding efficiency analysis

As described in Section II-1, picture partition schemes result coding loss. However as shown in Fig. 6 tiles cause less coding loss compared to slices. As shown in Fig. 7 slices produces large boundary compared to tiles and the coding loss for slices is high.

As designing the encoder with various parallelisms, coding loss and parallel scalability should be carefully considered. The coding efficiency is measured using the Bjøntegaard delta (BD) bitrate as described in [13]. To measure the BD bitrate (BDBR), test sequences are encoded with no parallelization and no picture partitioning. Table 1 - 2 shows the coding losses by encoding with Tile-level parallelism. The coding losses of the tile-level parallelism occur mainly at the tile boundary that the neighboring encoding information cannot be used. The coding loss by tile partitioning increases as the global motion in the sequence is large such as Jockey. Since a race horse runs very fast in Jockey, coding efficiency drops very significantly at tile boundaries. Note that for the Jockey sequence, our proposed Combined Approach 2 decreases the coding loss significantly compared to the fixed tile partitioning method. Table 3 - 4 shows the coding losses when we applied the adaptive picture partitioning according to the GOP level as described in Fig. 5. The results present that coding loss from Combined Approach 2 parallelism produces less coding losses compared to the Tile-level parallelism.
표 1. FHD: 타일 레벨 병렬화 방법의 부호화 효율(Y-BDBR)
Table 1. FHD: Tile-level parallelism coding efficiency (Y-BDBR)

| FHD Video | Tile 4 | Tile 8 | Tile 16 |
|-----------|--------|--------|---------|
| Kimono    | 1.1    | 1.8    | 3.5     |
| ParkScene | 0.2    | 0.7    | 1.4     |
| Cactus    | 0.5    | 0.8    | 1.8     |
| BQTerrace | 0.4    | 0.8    | 1.9     |
| BasketballDrive | 1.5 | 2.4 | 4.2 |
| Average   | 0.74   | 1.30   | 2.56    |

표 2. 4K: 타일 레벨 병렬화 방법의 부호화 효율(Y-BDBR)
Table 2. 4K: Tile-level parallelism coding efficiency (Y-BDBR)

| 4K60P Video | Tile 16 | Tile 24 | Tile 32 |
|-------------|---------|---------|---------|
| Jockey      | 41.5    | 45.4    | 48.7    |
| YachiRide   | 2.4     | 3.3     | 4.0     |
| ReadySteadyGo | 12.5 | 15.6    | 17.4    |
| ShakeNDry   | 0.9     | 1.2     | 1.6     |
| HoneyBee    | 4.4     | 5.2     | 6.3     |
| Average     | 12.3    | 14.1    | 15.6    |

표 3. FHD: 조합 2 방법의 부호화 효율(Y-BDBR)
Table 3. FHD: Combined approach 2 parallelism coding efficiency (Y-BDBR)

| FHD Video | Tile 8/4 | Tile 16/8/4 | Tile 24/12/8 |
|-----------|----------|-------------|--------------|
| Kimono    | 0.7      | 1.3         | 2.5          |
| ParkScene | 0.1      | 0.5         | 0.9          |
| Cactus    | 0.1      | 0.0         | 0.6          |
| BQTerrace | 0.6      | 0.6         | 1.3          |
| BasketballDrive | 1.5 | 2.1 | 3.6 |
| Average   | 0.60     | 0.90        | 1.78         |
3. Parallel scalability analysis

Table 5-10 show the speedup by Tile-level parallelism, Combined Approach 1 parallelism, and Combined Approach 2 parallelism, respectively. From the results of Table 7-10, the combined parallelism improves the parallel scalability significantly.

Table 5. FHD: Tile-level parallelism scalability (Speedup)

| Video      | Tile 4 | Tile 8 | Tile 16 |
|------------|--------|--------|---------|
| Kimono     | 3.08x  | 5.16x  | 6.77x   |
| ParkScene  | 2.86x  | 4.52x  | 5.47x   |
| Cactus     | 2.72x  | 3.81x  | 4.94x   |
| BQTerrace  | 2.84x  | 4.25x  | 5.27x   |
| BasketballDrive | 2.34x | 3.66x  | 5.40x   |
| Average    | 2.77x  | 4.28x  | 5.57x   |

Table 6. 4K: Tile-level parallelism scalability (Speedup)

| 4K60P Video       | Tile 16 | Tile 24 | Tile 32 |
|-------------------|---------|---------|---------|
| Jockey            | 7.12x   | 9.51x   | 9.51x   |
| YachiRide         | 7.82x   | 11.07x  | 11.07x  |
| ReadySteadyGo     | 7.06x   | 9.66x   | 9.66x   |
| ShakeNDry         | 6.48x   | 10.01x  | 10.01x  |
| HoneyBee          | 8.45x   | 10.84x  | 10.84x  |
| Average           | 7.39x   | 9.81x   | 10.22x  |

Table 7. FHD: Combined approach 1 parallelism scalability (Speedup)

| Video      | Tile 4 | Tile 8 | Tile 16 |
|------------|--------|--------|---------|
| Kimono     | 4.40x  | 6.35x  | 7.38x   |
| ParkScene  | 3.96x  | 5.58x  | 6.44x   |
| Cactus     | 3.78x  | 5.01x  | 6.13x   |
| BQTerrace  | 3.66x  | 5.06x  | 5.95x   |
| BasketballDrive | 3.49x | 5.09x  | 6.61x   |
| Average    | 3.86x  | 5.42x  | 6.50x   |

Table 8. 4K: Combined approach 1 parallelism scalability (Speedup)

| 4K60P Video       | Tile 16 | Tile 24 | Tile 32 |
|-------------------|---------|---------|---------|
| Jockey            | 9.07x   | 9.51x   | 10.80x  |
| YachiRide         | 10.73x  | 11.07x  | 13.39x  |
| ReadySteadyGo     | 9.40x   | 9.68x   | 11.64x  |
| ShakeNDry         | 9.61x   | 10.01x  | 12.55x  |
| HoneyBee          | 10.41x  | 10.84x  | 12.24x  |
| Average           | 9.85x   | 11.12x  | 12.12x  |

Table 9. 4K: Combined approach 2 parallelism scalability (Speedup)

| Video      | Tile 4/2/1 | Tile 8/4/2 | Tile 16/8/4 |
|------------|------------|------------|-------------|
| Kimono     | 3.25x      | 5.48x      | 7.05x       |
| ParkScene  | 3.06x      | 4.93x      | 6.22x       |
| Cactus     | 2.87x      | 4.21x      | 5.84x       |
| BQTerrace  | 2.94x      | 4.58x      | 5.80x       |
| BasketballDrive | 2.82x | 4.40x      | 5.70x       |
| Average    | 2.99x      | 4.72x      | 6.14x       |

Table 10. 4K: Combined approach 2 parallelism scalability (Speedup)

| Video      | Tile 16/8/4 | Tile 24/12/6 | Tile 32/16/8 |
|------------|-------------|--------------|--------------|
| Jockey     | 7.53x       | 9.05x        | 10.28x       |
| YachiRide  | 8.33x       | 10.11x       | 11.58x       |
| ReadySteadyGo | 8.12x | 9.20x        | 10.67x       |
| ShakeNDry  | 8.07x       | 9.24x        | 10.84x       |
| HoneyBee   | 9.57x       | 10.57x       | 11.97x       |
| Average    | 8.32x       | 9.63x        | 11.07x       |

When designing the encoding parallelism, we carefully consider the trade-off between the speedup and coding efficiency. From our experimental results, we aggregate the speedup results against coding loss for each parallelization scheme as shown in Fig. 8. Both Combined Approach 1 and 2 are better parallel scalability compared to the Tile-level parallelism. Combined Approach 2 parallelism shows better
speedup against the coding loss for Full-HD and 4K test sequences. In addition, the comparison of speedup between the combined approach 2 and the tile-level parallelism shows the similar pattern as Fig 9 no matter what number of physical core. However, the speedup scalability according to the number of available core is different. The speedup scalability decreases as the number of cores increases. This performance decrease causes from the inefficient parallelization such as frame synchronization.

(Younhee Kim et al.: Tile-level and Frame-level Parallel Encoding for HEVC)}

Fig. 8. Speedup against BD bitrate

Fig. 9. Speedup against number of thread and number of cores
IV. Conclusion

In this paper, we present two effective approaches of combining the tile-level parallelism and frame-level parallelism. Both approaches provide better parallel scalability compared to the tile-level parallelism and the experimental results show that when combining the tile-level parallelism and frame-level parallelism, applying the adaptive number of tile as taking the consideration of the number of frames in parallel and the number of available cores results in better trade-off between coding loss and parallel scalability for Full-HD and 4K UHD video sequences.

참고 문헌 (References)

[1] High Efficiency Video Coding, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, Apr. 2013.
[2] G. J. Sullivan et al., “Overview of the High Efficiency Video Coding (HEVC) standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[3] Y. Kim et al., “A Fast Intra-Prediction Method in HEVC Using Rate-Distortion Estimation Based on Hadamard Transform,” ETRI Journal, vol. 35, no. 2, pp. 270-280, Apr. 2013.
[4] A. Fuldseth et al., “Tiles”, JCT-VC document JCTVC-F335, Torino, Jul. 2011.
[5] C. Gordon, F. Henry, and S. Pateux, “Wavefront Parallel Processing for HEVC Encoding and Decoding,” JCT-VC document JCTVC-F274, Jul. 2011.
[6] J. M. Thiesse and J. Viéron, “On Tiles and Wavefront tools for parallelism,” JCT-VC document JCTVC-I0198, Geneva, Apr., 2012.
[7] S. Zhang, X. Zhang, and Z. Gao, “Implementation and Improvement of Wavefront Parallel Processing for HEVC Encoding on Many-core Platform,” Proceedings of IEEE International Conference on Multimedia and Expo Workshops (ICMEW) 2014, pp. 1-6.
[8] C. Chi et al., “Parallel Scalability and Efficiency of HEVC Parallelization Approaches,” IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1827-1838, Dec. 2012.
[9] C. C. Chi et al., “Parallel Scalability and Efficiency of WPP and Tiles,” JCT-VC document JCTVC-I0520, Geneva, 2012.
[10] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC document JCTVC-I1100, Apr. 2012.
[11] M. Viitanen et al., “Kvazaar HEVC encoder for efficient intra coding,” in Proc. IEEE Int. Symp. Circuits Syst., Lisbon, Portugal, May 2015. Ultra video group [Online]. Available:. .
[12] G. Bjontegaard, “Calculation of Average PSNR Differences between RD-Curves,” Document VCEG-M33, Austin, TX, USA, Apr. 2001.
저 자 소 개

석 진 욥
- 1993년 : 홍익대학교 전기전도공학과 학사
- 1995년 : 홍익대학교 전기전도공학과 석사
- 1998년 : 홍익대학교 전기공학과 박사
- 2000년 ~ 현재 : 한국전자통신연구원 업상미디어연구실 책임연구원
- 주관심분야 : 비선형 확률시스템, 업상, UHDTV 방송 시스템

정 순 홍
- 2001년 : 부산대학교 전기공학과 학사
- 2003년 : 한국과학기술원(KAIST) 전기및전자공학과 석사
- 2003년 ~ 2005년 : LG전자 주임연구원
- 2005년 ~ 현재 : 한국전자통신연구원 업상미디어연구실 선임연구원
- 2010년 ~ 현재 : 한국과학기술원(KAIST) 전기및전자공학과 박사과정
- 주관심분야 : 업상처리, 업상통신, 실감방송

김 휘 용
- 1994년 : 한국과학기술원(KAIST) 전기및전자공학과 학사
- 1998년 : 한국과학기술원(KAIST) 전기및전자공학과 석사
- 2004년 : 한국과학기술원(KAIST) 전기및전자공학과 박사
- 2003년 ~ 2005년 : 캠퍼스네트워크기술연구소 멀티미디어팀장
- 2005년 ~ 현재 : 한국전자통신연구원 업상미디어연구실장
- 주관심분야 : 업상통신, 컴퓨터 비전, 멀티미디어 시스템, 실감미디어서VICES

최 진 수
- 1990년 : 경북대학교 전자공학과 학사
- 1992년 : 경북대학교 전자공학과 석사
- 1996년 : 경북대학교 전자공학과 박사
- 1996년 ~ 현재 : 한국전자통신연구원 실감방송미디어연구부장
- 주관심분야 : 업상통신, UHDTV 방송, 실감미디어서VICES