PARALLEL AMVP CANDIDATE LIST CONSTRUCTION FOR HEVC

Qin Yu[1], *Liang Zhao*[2], *Siwei Ma*[1]
[1]Institute of Digital Media, Peking University, Beijing 100871，China
[2]Key Lab of Information Processing, Institute of Computing Technology, Chinese Academy of Sciences
{qyu, swma}@pku.edu.cn, lzhao@jdl.ac.cn

ABSTRACT

Advanced motion vector prediction (AMVP) is one of the most important inter prediction coding tools adopted in the state-of-the-art HEVC coding standard, which does great effect on the coding efficiency. However, the current AMVP design is highly sequential and thus restricts the throughput both on the encoder and the decoder sides. To facilitate the parallel processing and enlarge the throughput, a parallel AMVP candidate list (AMVPCL) construction solution is proposed. The proposed parallel scheme consists of a three level fine granularity solutions. The first level is a CU-based approach and it constructs AMVPCL of all PUs in the same CU in parallel. The second level is also at CU level but it generates a single set of AMVPCL for all PUs inside a CU. Specifically, we only apply this method to 8x8 CU to balance the parallelism degree and rate-distortion performance. The third level is a CU-group based approach, in which AMVPCL of all PUs in the same CU-group are constructed in parallel. Experimental results show the proposed algorithm can efficiently achieve parallel motion estimation with negligible 0.0%~1.3% BD-rate loss at different degree of parallelism.

Index Terms**—**video coding, AMVP, parallel, HEVC

1. INTRODUCTION

High Efficiency Video Coding (HEVC) is the upcoming new video coding standard under development of JCT-VC(Joint Collaborative Team-Video Coding), which is a joint team of ISO/IEC MPEG and ITU-T VCEG. By adopting a series of new coding tools and strategies, the compression efficiency is nearly doubled compared to its predecessor, H.264/AVC. The significant coding efficiency improvement is achieved by a series of new coding tools. Advanced motion vector prediction (AMVP) technique is one of these new coding tools used in inter-prediction.

As we know, there exists redundancy among the motion vectors of neighboring blocks. If we encode one motion vector for each block directly, it may cost large numbers of bits especially for smaller block size. And the proportion of bits for motion vector and bits for entire stream is significant especially for large QP values. Since the motion vectors of the neighboring blocks are correlated with each other, the motion vectors of the neighboring blocks can be utilized to predict the motion vector of the current block, which is called motion vector prediction (MVP). For MVP, only the difference between the current motion vector and its predictor needs to be transmitted, thus bits for motion vector are reduced largely.

MVP is widely used in the existing video coding standards, e.g. H.263 and H.264/AVC. In H.264/AVC [1], the median of motion vectors of the neighboring blocks A, B, and C are used as the motion vector predictor of the current block. The position of A, B, and C are located as shown in Fig.1.

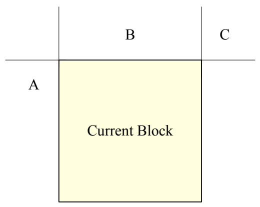


Fig.1. The blocks used for MVP in H.264/AVC

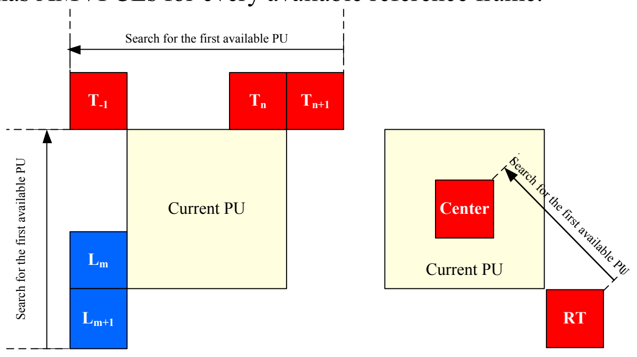
In order to further improve the coding efficiency, the emerging HEVC standard employs a motion vector competition mechanism, which is called advanced motion vector prediction (AMVP). For AMVP, the best motion vector predictor for the current block is selected from a set of predictors and the index of the predictor is transmitted to the decoder. In JCT-VC A124 [2], an improved AMVP method is proposed to adapt to the large block and flexible temporal structure. The encoder selects the best predictor from a given AMVP candidate list (AMVPCL), which is composed of three spatial motion vectors, a median motion vector and a temporal motion vector. These three spatial motion vectors are chosen from the above, left and from each applicable corner. And the temporal motion predictor is given by the nearest reference frame and is scaled according to the temporal distance. To optimize this technique, many proposals such as JCT-VC D231 [3], JCT-VC E481 [4] and JCT-VC F470 [5] are proposed. In the current HEVC [6], the length of the candidate list is fixed to three and the final best motion vector is chosen from first two of them.

With the current AMVP technique, the coding efficiency has been improved to some extent. However, the construction of AMVPCL needs the motion information of neighboring blocks. This dependency of blocks makes the motion estimation in the encoding process which is the most time consuming module and motion vector derivation in the decoding process difficult to do in parallel. The prediction of the neighboring blocks must be conducted sequentially in raster scan order. In this paper, we proposed three efficient AMVPCL construction methods to remove this dependence and implement parallelism at different granularities. With the proposed methods, the motion estimation process of the blocks in the same parallel region can be conducted concurrently with negligible loss.

The remainder of this paper is organized as follows. Section 2 presents the parallel implementation problem of AMVP. Section 3 gives a detailed description of the proposed AMVPCL parallel construction algorithm. The complexity of proposed algorithm is analyzed in Section 4. Experimental results are shown in Section 5. Finally, we make a conclusion of this paper.

2. AMVP IN THE CURRENT HEVC

In HEVC, CU is basic coding unit similar to macroblock, which can have various sizes and allows recursive quad-tree splitting. PU is the basic unit for prediction and it allows multiple different shapes to encode irregular image pattern. PU is limited to that of CU with symmetrical partition (SMP) and asymmetrical partition (AMP). Within the current HM, there are 8 PU types, which are 2Nx2N, 2NxN, Nx2N, 2NxnD, 2NxnU, nDx2N, nUx2N and NxN. Among these PU types, the 2Nx2N PU type divides the CU into one PU while the others divide the CU into multiple PUs. Each PU has AMVPCLs for every available reference frame.


Search for the first available PU
T_{-1}
T_n
T_{n+1}
Search for the first available PU
Current PU
L_m
L_{m+1}
Center
Current PU
Search for the first available PU
RT

(a) Spatial Candidates (b) Temporal Candidates

Fig. 2. Illustration of the AMVPCL construction of HM6.0

In HM6.0, the AMVPCL is composed of both spatial candidates and temporal candidates. Spatial candidates are classified into 2 categories, i.e. top (T_{-1}, T_n and T_{n+1}) and left category (L_m and L_{m+1}) [7], as shown in Fig. 2(a). In each category, the first existed and non-intra coded candidate in the search order is added to the AMVPCL. After the spatial candidates are derived, a temporal candidate from the collocated frame is added to the AMVPCL, as shown in Fig. 2(b). Therefore, the candidate list contains 2 spatial candidates and 1 temporal candidate at most.

Within the current HEVC test model HM6.0, on the encoder side, the AMVPCL for the current PU is first derived, and then a temporary best MVP is selected as the start point for motion estimation (ME). When the ME process is finished, the final optimal MVP is reselected according to the obtained motion vector. On the decoder side, the AMVPCL is firstly constructed, and then the MVP can be derived according to the decoded index. When encoding a CU, only the first PU in it can immediately derive its AMVPCL while other PUs have to wait until its preceding PUs are encoded, as shown in Fig. 3. Fig. 3 illustrates the different kinds of candidates highlighted in different colors for SMP cases, and the same way is also applied to AMP cases. The top ones are in red, and the left ones are in blue. The gray ones indicate the candidates are available until they are coded, while the green ones are candidates that are not encoded yet when the current PU is coding.

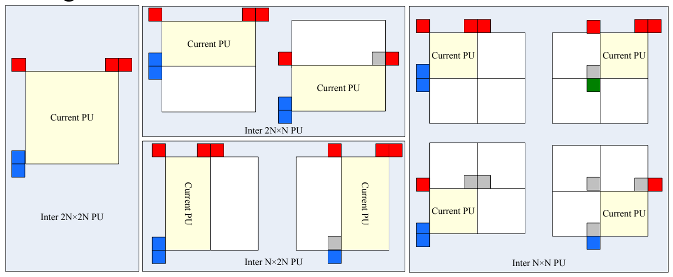
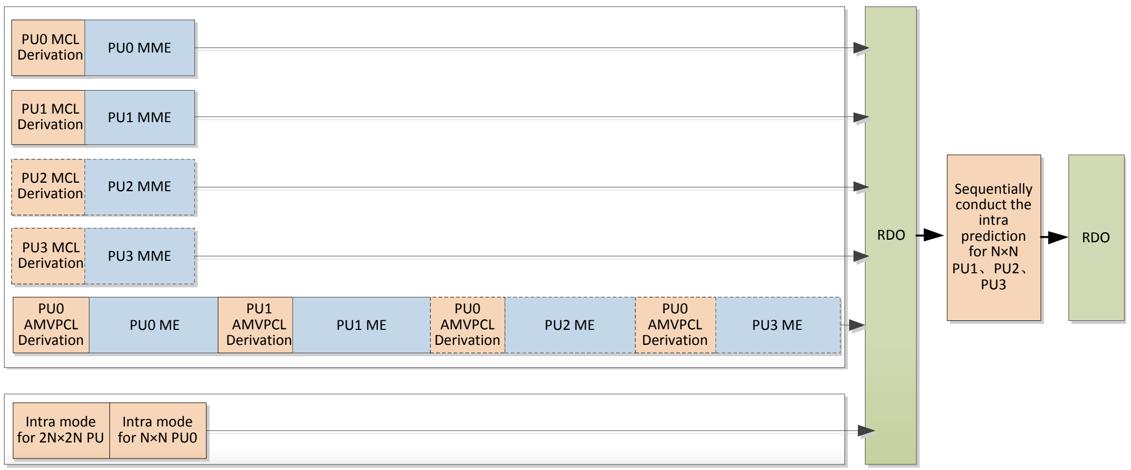


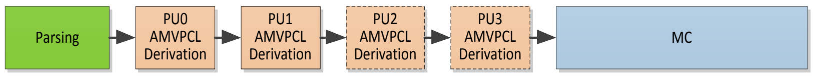
Fig. 3. AMVPCL of PUs in a CU

So we can conclude that the spatial candidates are highly dependent on its neighboring PUs, and consequently AMVPCL derivation process has to be done sequentially on both the encoder and the decoder sides. This makes parallel processing of multiple inter PUs difficult for both the encoder and decoder in inter modes. This sequential behavior directly limits the throughput of the encoder and decoder. Fig. 4 provides an example to further elaborate the problem. PU0, PU1, PU2 and PU3 represent the different PU in a CU. As can be seen, the merge candidate list (MCL) derivation process for different PUs can be derived in parallel as well as the merge mode motion estimation (MME), but the AMVPCL derivation and regular motion estimation (ME) process must be carried out sequentially.

From early 1990s to now, a series of video coding standards have been established. In order to adapt to the new application requirements, the encoder and decoder are becoming more and more complex. What's more, our requirement on the resolution of video sequence has been changing from SD, HD to ultra-high-definition (UHD). Although the computation power and hardware techniques have been improved significantly, real time encoding is still challenging for the emerging HEVC standard, especially for UHD HEVC coding. Moreover, with the rise of video sites, IP-based video playback places higher and higher demands on speed of the codec. One important way to enhance encoding and decoding speed is parallel processing techniques. To realize parallel motion estimation, three solutions with different parallel granularities are proposed in the following section.



(a) Encoder



(b) Decoder

Fig. 4. An example of parallel implement on motion estimation in HM6.0

3. PROPOSED AMVP CANDIDATE LIST CONSTRUCTION ALGORITHM

To enlarge the throughput both on the encoder and the decoder sides, three parallel solutions for AMVPCL with different granulites are introduced in this section, called Solution I, II, III respectively. Solution I is a CU-based approach and it constructs AMVPCL of all PUs in the same CU in parallel. Solution II is also a CU-based approach but it generates a single set of AMVPCLs the same with that of the inter 2Nx2N PU for all PUs inside a CU. Specifically, we only apply this method to 8x8 CU to balance the parallelism degree and rate-distortion performance. Solution III is a CU-group based approach, in which AMVPCL of all PUs in the same CU-group are constructed in parallel.

Solution I: CU based parallel AMVPCL construction for all PUs in a CU

In the parallel AMVPCL construction process, the candidates within the CU, which is called inner candidates, are unavailable. Since spatial dependency exists among neighboring PUs, we can find an alternative for the unavailable candidate. A parallel AMVPCL construction method for all PUs in a CU is proposed in solution I. We replace these unavailable candidates with the corresponding ones outside the CU. Thus the spatial dependency among PUs in the same CU is removed. Fig. 5 illustrates our proposed method for symmetric motion partition (SMP) cases, and the same method is applied to asymmetric motion partition (AMP) cases. As can be seen from Fig. 3, inner candidates (in gray and green) of the PU are replaced by the corresponding candidates pointed by the arrows in Fig. 5, which are outside of the current CU.

Solution II: CU based parallel AMVPCL construction with all PUs sharing the AMVPCL of Inter 2Nx2N PU

From the knowledge of section 2, we know that we need to traverse all PU types to find the best PU partition for a CU. Each PU type has one or multiple PUs and a PU has AMVPCLs for every reference frame. Then the total number of AMVPCL for a CU is very large. In solution II, no matter what kind of partition mode a CU uses, all PUs in it use the same set of AMVPCL with that of inter 2Nx2N partition mode. The proposed solution reduces the construction rounds of AMVPCL significantly. The simplification will be analyzed in the next section. Fig. 6 illustrates proposed method for SMP cases, and the same way is also applied to AMP cases.

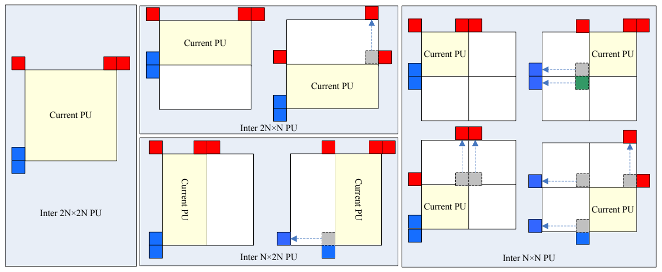


Fig. 5. Proposed Solution I: a CU-based approach for AMVPCL construction

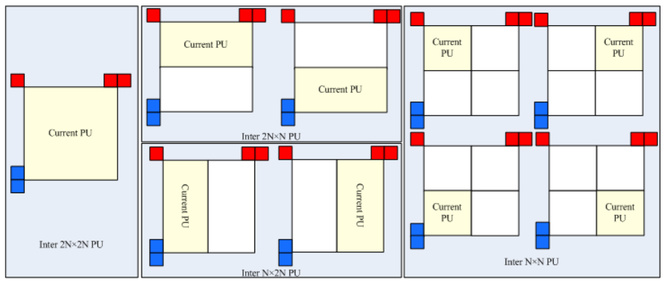


Fig. 6. Proposed Solution II: all PUs in the CU share one set of AMVPCL

Solution III: CU group based parallel AMVPCL construction for all PUs inside the same CU group

In solution III, all PUs inside a CU group construct their AMVPCL in parallel.

To specify the size of CU group, a syntax element, log2_parallel_amvp_level_minus2, is defined. The relationship between log2_parallel_amvp_level_minus2 and the size of CU group is tabulated in Table 1. The value of log2_parallel_amvp_level_minus2 varies between 0 and 4. If the size of CU group is NxN, then the value of N can be specified as follow:

$$N = 2^{\log2\_parallel\_amvp\_level\_minus2+2} \tag{1}$$

Table 1. Partition of CU group

log2_parallel_ amvp_level_m inus2	Size of CU group	Remark
4	64x64	Parallel AMVPCL derivation for all PUs inside a LCU
3	32x32	Parallel AMVPCL derivation for all PUs inside a 32x32 block
2	16x16	Parallel AMVPCL derivation for all PUs inside a 16x16 block
1	8x8	Parallel AMVPCL derivation for all PUs inside a 8x8 block
0	4x4	Sequential AMVPCL derivation for all PUs as the smallest PU is 4x4

If the involved candidate and the current PU are within the same CU group, this candidate is disabled. Fig. 7 shows the CU group partition and the AMVPCL construction. In Fig. 8, a LCU is quad-tree divided into four CU groups, and the size of each group is 32x32. For PU$_2$, all its candidates are in the same CU group with it, so its AMVPCL has only temporal candidate and additional candidates.

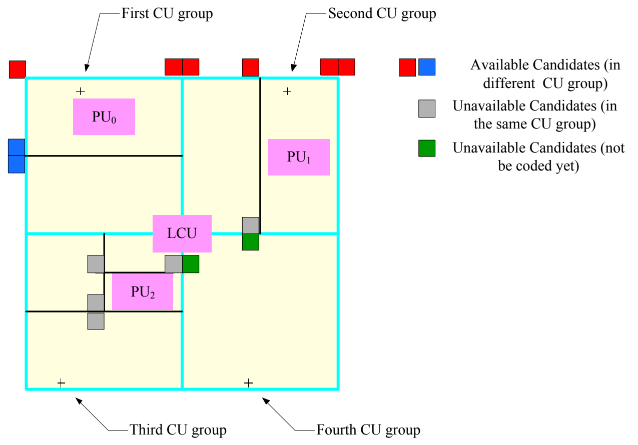

First CU group
Second CU group
PU$_0$
PU$_1$
LCU
PU$_2$
Third CU group
Fourth CU group
🟥🟦 Available Candidates (in different CU group)
⬜ Unavailable Candidates (in the same CU group)
🟩 Unavailable Candidates (not be coded yet)

Fig. 7. CU group partition and AMVPCL construction

4. COMPLEXITY AND PARALLELISM ANALYSIS OF THE PROPOSED AMVPCL ALGORITHM

This section provides complexity and parallelism analysis of the proposed parallel AMVPCL construction algorithm.

Complexity analysis

All three proposed solutions are analyzed one by one below. Let's start from Solution I. From Fig.5, it can be seen that the proposed approach only needs some judgments to decide whether the candidate and the current PU belong to the same CU. The number of judgments is one for two-partition type and seven for four-partition type (the number of gray PUs in Fig.5). This additional operation is negligible when compared to the parallelism it brings.

For Solution II, in the current HM, each PU has its own set of AMVPCLs, whose size is the number of reference frame. On the encoder side, the AMVPCLs to be constructed for motion estimation could reach a very large number as shown in Table 2. In Table 2, the numbers in first and second brackets refer to the number of CUs in a 64x64 block and PUs in the CU respectively; N is the number of reference frame in all reference lists. For CU larger than 8x8, there are one 2Nx2N PU, four SMP PUs and eight AMP PUs, and for an 8x8 CU, there are one 2Nx2N PU, four SMP PUs. For a 64x64 block, the number is 593xN. The larger the number, the more occurrence chance of different motion candidates increasing the memory contention possibility. Note that even when the AMVPCLs could be constructed in parallel, the memory can be accessed only in a sequential manner. Thus for high-throughput encoder design, it is desirable to reduce the number of different motion candidates as much as possible. Note that the proposed solution significantly reduces the number of AMVPCLs that should be constructed for motion estimation. Table 2 shows that the number for a 64x64 block is reduced by more than 80%.

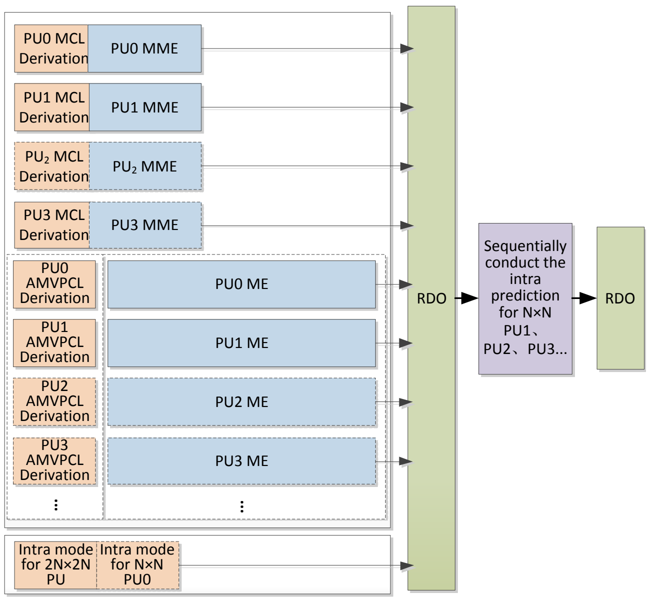
Table 2. Number of AMVPCLs constructed for 64x64 block motion estimation

CU Size	AMVPCL construction for a 64x64 block		
	HM6.0	Proposed	Rounds reduction
64x64	(1)* (13)* N	(1)*(1)*N	92%
32x32	(4)* (13)* N	(4)* (1)* N	92%
16x16	(4*4)* (13)*N	(4*4)* (1)* N	92%
8x8	(4*4*4)*(5)*N	(4*4*4)*(1)*N	80%
Sum	593*N	85*N	86%

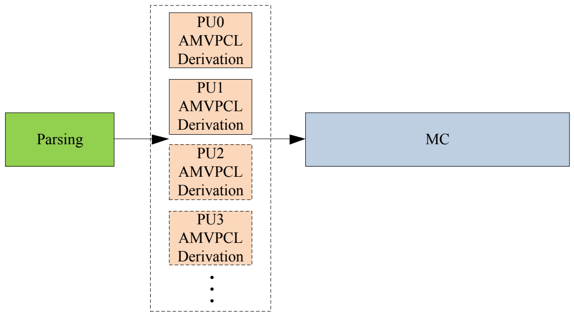
For solution III, the proposed parallel AMVPCL construction approach needs to check whether the current PU and its neighboring PU belong to the same CU group during the spatial MVP derivation process. However, the complexity increase is negligible compared to the amount of availability checks already needed in the spatial MVP derivation process of HM6.0.

Parallelism analysis

With the proposed solutions, the throughput analysis on the encoder and decoder sides is shown in Fig. 8. On the encoder side, all PUs inside a given parallel region can derive their AMVPCLs concurrently. Specially, for solution II, only one set of AMVPCL should be derived before motion estimation; and for solution III, if the size of CU group is larger than 8x8, the PUs from different CU depth can also conduct AMVPCL and motion estimation in parallel. On the decoder side, the AMVPCLs of different PUs which locate in the same parallel region can be derived in parallel. For solution II, if the PUs in one CU have the same reference frame, they can share the same AMVPCL.



(a) Parallel AMVPCL in Encoder



(b) Parallel AMVPCL in Decoder

Fig. 8. Throughput analysis

5. EXPERIMENT RESULT

To verify the effectiveness of the proposed methods, they are implemented into HM6.0 software. As the proposed algorithm focus on the AMVPCL parallel construction, experiments are only conducted on six test conditions, which are random access high-efficiency setting (Random Access HE10), random access main setting (Random Access Main), low delay high-efficiency setting (Low delay B HE10), low delay main setting (Low delay B Main), low delay P high-efficiency setting (Low delay P HE10), and low delay P main setting (Low delay P Main) respectively.

The test platform used is Intel (R) Xeon (R) CPU X5660-2.80GHZ cluster 23.9G RAM. A group of experiments were carried out on the common test sequences with quantization parameters 22, 27, 32 and 37 as specified by [8]. For solution II, to balance the parallelism and the rate-distortion performance, it is only applied to 8x8 CU. Table 3, 4 and 5 show the summary results of the proposed solutions against HM6.0.

From Table 3, it can be seen that if we implement parallel motion estimation for all PUs in a CU using Solution I, the average BD-rate is increased by 0.2%. And if we conduct motion estimation in parallel for all PUs in an 8x8 CU, the coding complexity is deceased with negligible average BD-rate increase of 0.1%. From Table 5, conclusion can be made that larger parallel region leads to more loss of coding performance. When the size of parallel CU group varies from 8x8 to 64x64, the average bit rate increase varies from 0.0% to 1.3%. All the three proposed solutions can facilitate parallelism with negligible loss, but we suppose to adopt the third solution as we can change the parallel degree to balance the speed up requirement and the coding performance.

Table 3. Summary results of Solution I

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
0.30%	0.40%	0.40%	0.30%	0.30%	0.30%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
0.20%	0.10%	0.40%	0.20%	0.20%	0.00%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.10%	0.10%	-0.10%	0.10%	0.20%	0.10%

Table 4. Summary results of Solution II

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
0.10%	0.20%	0.20%	0.10%	0.10%	0.00%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
0.10%	0.00%	0.10%	0.00%	0.00%	0.00%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.10%	-0.10%	-0.30%	0.00%	0.30%	0.00%

Table 5. Summary results of solution III for different size of CU group

(a) log2_parallel_amvp_level_minus2=4

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
2.40%	2.40%	2.40%	2.40%	2.30%	2.30%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
1.00%	1.10%	1.30%	1.00%	0.8%	1.00%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.60%	0.30%	0.20%	0.60%	0.60%	0.40%

(b) log2_parallel_amvp_level_minus2=3

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
1.50%	1.40%	1.60%	1.40%	1.40%	1.40%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
0.70%	0.60%	0.70%	0.60%	0.50%	0.50%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.40%	0.20%	0.20%	0.40%	0.60%	0.20%

(c) log2_parallel_amvp_level_minus2=2

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
0.60%	0.60%	0.60%	0.50%	0.50%	0.50%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
0.30%	0.20%	0.50%	0.20%	0.10%	0.10%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.20%	0.10%	-0.20%	0.20%	0.20%	0.00%

(d) log2_parallel_amvp_level_minus2=1

Random Access Main			**Random Access HE10**		
Y	U	V	Y	U	V
0.10%	0.20%	0.10%	0.10%	0.10%	0.10%
Low delay B Main			**Low delay B HE10**		
Y	U	V	Y	U	V
0.00%	0.10%	0.10%	0.00%	-0.10%	-0.10%
Low delay P Main			**Low delay P HE10**		
Y	U	V	Y	U	V
0.00%	-0.10%	-0.10%	0.00%	0.30%	-0.40%

6. CONCLUSION

In the current HEVC framework, there exists significant dependency among neighboring PUs. It makes parallel processing of multiple inter PUs difficult for both the encoder and decoder. In this paper, we propose three solutions to solve this problem at different parallelization levels, called Solution I, II, III respectively. Solution I is a CU-based approach and it constructs AMVPCL of all PUs in the same CU in parallel. Solution II is also a CU-based approach but it generates a single AMVPCL for all PUs inside a CU. Specifically, we only apply this method to 8x8 CU to balance the parallelism degree and rate-distortion performance. Solution III is a CU-group based approach, in which AMVPCL of all PUs in the same CU-group are constructed in parallel. And we can change the size of parallel region to adapt to different applications. The proposed approaches improve parallelism of all inter modes excluding merge/skip mode hence make the HEVC design more friendly to high-throughput implementation, at the cost of negligible loss in RD performance.

ACKNOWLEDGEMENT

This research is supported by the 973 program (2009CB320903), the 863 program (2012AA011505) and the National Science Foundation of China (61121002, 61103088) of China, which are gratefully acknowledged.

REFERENCES

[1] ITU-T and ISO/IEC JTC 1. Advanced video coding for generic audiovisual service. ITU-T and ISO/IEC JTC 1 Recommendation H.264 and ISO/IEC 14 496-10(MPEG-4) AVC, 2003.

[2] K. McCann, "Samsung's Response to the Call for Proposals on Video Compression Technology", JCT-VC A124, 1th JCT-VC Meeting, Dresden, Germany, 15-23 April, 2010.

[3] A. Fujibayashi, "Simplified Motion vector prediction" JCT-VC D231, 4th Meeting: Daegu, Korea, 20-28 January, 2011.

[4] B. Bross, " MV Coding and Skip/Merge operations", JCT-VC E481, 5th Meeting: Geneva, CH, 16-23 March, 2011.

[5] T. Sugio , "Parsing Robustness for Merge/AMVP", JCT-VC F470, 6th Meeting: Torino, IT, 14-22 July, 2011.

[6] B. Bross, "High efficiency video coding (HEVC) text specification draft 6", JCTVC-H1003, 8th JCT-VC Meeting, San Jose, CA, USA, 1-10 February, 2012.

[7] L. Zhao, X. Guo, S. Lei, S. Ma, D. Zhao and W. Gao, "Non-CE9: Simplification of AMVP", JCTVC-H0316, 8th JCT-VC Meeting, San Jose, CA, USA, 1-10 February, 2012.

[8] F. Bossen, "Common test conditions and software reference configurations", JCTVC-H1100, 8th JCT-VC Meeting, San Jose, CA, USA, 1-10 February, 2012.