Highly Parallel Mode Decision Method for HEVC

Jun Zhang\textsuperscript{1,2}, Feng Dai\textsuperscript{1}, Yike Ma\textsuperscript{3}, and Yongdong Zhang\textsuperscript{1}
\textsuperscript{1}Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China
\{zhangjun01, fdai, ykma, zhyd\}@ict.ac.cn

Abstract—High Efficiency Video Coding (HEVC) standard achieves double compression efficiency compared to H.264/AVC with the adoption of more flexible coding structure and advanced coding tools. On the other hand, the coding mode space is too large and it’s very time consuming for an HEVC encoder to search for the best coding mode. With the development of multi-core or many-core computing architecture, parallelizing HEVC encoding on such platforms is an efficient solution to fulfill the high computational requirement. In this paper, we exploit the potential parallelism in HEVC mode decision (MD) process and propose a highly parallel MD method which works in a motion estimation region (MER). Specifically, we analyze and remove data dependencies that hinder parallel MD, including motion estimation (ME) dependencies and entropy coding dependencies, and then the MD computation for different blocks within the same MER can be computed concurrently. Experimental results show that our proposed parallel MD method gets an overall speed up of more than 14x with negligible quality loss (1.79\% bit rate increasing), compared with the non-parallel baseline.

Keywords—HEVC; parallel mode decision; many-core

I. INTRODUCTION

HEVC is developed [1, 2] with the target of outperforming all existing standards, specifically doubling the compression efficiency than H.264/AVC. Although still conforming to the hybrid video coding framework, HEVC adopts enhancements in each stage. One of the most important innovations is the adoption of flexible coding structure. As shown in Fig. 1(a), a Coding Tree Unit (CTU) is split recursively into four equally sized units in a quad-tree manner and a leaf node of the resulting recursive quad-tree (Coding Tree) is called a Coding Unit (CU). For a certain CU, it is also recursively split in a quad-tree manner with leaf nodes of this tree (Transform Tree) called Transform Units (TU). Furthermore, there are many prediction unit (PU) partition modes available for a CU to flexibly perform video content prediction, as shown in Fig. 1(b). Due to the flexible block splitting in HEVC, the coding mode search space is very large for an encoder to seek for the best mode combination. Many research works have been done to accelerate HEVC MD. Proposals [3-5] propose efficient fast algorithms including early detection and mode skipping and all of these proposals have been incorporated in the HEVC reference software HM [6]. These methods reduce HEVC encoding complexity to some extent, but the speed up they provide is not high enough.

With the increasing development of multi-core and many-core computing architectures, parallelizing HEVC encoding on such platforms is an efficient solution to fulfill the high computational requirement of HEVC encoder [7-9]. Compared with its predecessors, HEVC standard itself adopts several features that facilitate parallel codec such as wavefront parallel processing (WPP) [10], tile [11] and motion estimation region (MER) [12]. WPP has low impact on coding quality but with relatively low parallelism. [13] proposes a method namely overlapped wavefront (OWF) based on WPP which can improve the parallelism. Partitioning a picture into many tiles can provide relatively a high parallelism but this will severely drop compression efficiency. While WPP and tile facilitate parallel processing among large regions, MER defines a local region to allow parallel ME within this region, but MER only applies to ME and other parts of MD still has to computed sequentially which results in low overall throughput.

This paper utilizes and extends the benefits of MER and proposes a parallel HEVC MD method inside it. Besides ME dependencies, additional data dependencies are removed to achieve high overall throughput. From MER to MER, MD is performed sequentially in z scan order. We focus on MD for inter CUs here and parallel intra MD will be our further work.

The rest of this paper is organized as follows. In section II we briefly cover the background including MD in HM and MER in HEVC. In section III we analyze and remove the data dependencies hindering parallel MD and our proposed parallelization strategies are then introduced. In section V the experimental results are shown. At last, this paper is concluded.

II. BACKGROUND

A. Mode Decision for HEVC Encoding

Take a CTU or MER for example, its mode means the CU splitting manner within itself, PU partition mode of each CU, prediction information of each PU, and the TU splitting manner of each CU. Typical implementations of HEVC encoding, like HM encoder, perform a depth-first traversal over the whole coding tree as depicted in Fig. 2. For every node in this quad-tree, firstly the rate-distortion (RD) cost of it being coded as a complete CU, i.e. without further splitting, is computed, denoted by Cost\textsubscript{r}. Then if maximum depth is not reached, the encoder splits current node into four child CUs and the same routine applies to them in z-scan order. When all the four child nodes have completed MD, the encoder compares the total cost

Figure 1. Flexible block partitioning in HEVC. (a) CU and TU, (b) PU partition.
of them \((\text{Cost}_{0} + \text{Cost}_{1} + \text{Cost}_{2} + \text{Cost}_{4})\) with \(\text{Cost}_{4}\) to
determine whether current node should be split or not. This
traversing process provides the best coding efficiency but
consumes too much time.

![CTU or MER](image)

**Figure 2.** Full search mode decision method adopted in HM.

B. MER in HEVC

The merge mode design in HEVC is highly sequential
because strong dependencies exist among adjacent PUs which
hinders parallel ME. HEVC adopts a high-level syntax element
[12] that divides a CTU into a number of MERs. A MER has
square shape whose size can be the same as CTU, a quarter of
the CTU and so on. When constructing the merge candidate list
(MCL) of current PU, neighboring PUs belonging to the same
MER are considered unavailable and not added into the MCL.
In this way, the merge mode computation for all PUs in the
same MER can be performed concurrently. In this paper, our
proposed parallel MD method works in a MER.

![Current PU](image)

**Figure 3.** The effect of MER on AMVPCL construction.

III. PROPOSED PARALLEL MD METHOD

In this section, we firstly analyze and remove kinds of data
dependencies which go against parallel processing within a
MER, and then our proposed parallel MD method is presented.

A. Data Dependencies Analysis

As depicted in Fig. 2, MD of a CU in the coding tree can
start only after CUs with earlier z-scan address has finished.
This sequential processing order is due to data dependencies
such as ME dependencies and entropy coding dependencies.

1) Regular ME dependencies: MER delimits a region in
which all PUs can conduct merge mode computation
concurrently, but regular ME within the same MER cannot be
conducted concurrently. For regular motion representation,
HEVC adopts a tool called advanced motion vector prediction
(AMVP) to encode MV. Similar to the merge mode, AMVP
also constructs a candidate list (AMVPCL) by referring to
neighboring PUs which leads to strong dependencies among
neighboring PUs. Because MER only decouples PU
dependencies during MCL construction, parallel ME in a
MER is still restrained by AMVP dependencies during regular
ME. [14] proposes methods to solve this problem but their
approaches are not standard compliant.

2) Context model (CM) derivation dependencies: TU
splitting computation for different CUs in a MER is not
straightforward to be parallelized. HEVC uses only one
entropy coding method, i.e. context adaptive binary arithmetic
coding (CABAC). In order to guarantee coding quality, CMs
used for MD of a CU are inherited from those of previous CUs
which have been trained for a period of time and have been
adapted to the specificities of the local image content. So if
parallel MD in a MER containing multiple CUs is expected,
the encoder must solve the problem of CM absence for each
CU because the encoding of neighboring CUs is also being
performed and the accumulated CMs are not available yet.

3) CM selection dependencies: In order to improve coding
efficiency of CABAC, neighboring mode information
is needed to do CM selection or context modeling for a bin,
which produces additional dependencies among CUs.

B. Dependencies Removing

1) Regular ME dependencies: As has been explained,
regular ME for different PUs are still interdependent because
of AMVP. We propose that if the referenced PU is in the same
MER with current PU, we replace it with an alternate which
lies in a previously coded MER, as depicted in Fig. 3. Let \((x_P,
y_P)\) be the coordinate of the top-left luma pixel of current PU,
\(\text{nPbW}\) and \(\text{nPbH}\) indicate the width and height of it, the PU
covering position \((x_N, y_N)\) is considered to be in the same
MER with current PU if the criterion (1) is met, in which \(S_{M}\)
means the size of MER. \((x_N, y_N)\) is \((x_P - 1, y_P + \text{nPbH})\), \((x_P
- 1, y_P + \text{nPbH} - 1)\), \((x_P + \text{nPbW}, y_P - 1)\), \((x + \text{nPbW} - 1,y_P
- 1)\) and \((x_P - 1, y_P - 1)\) respectively for the five spatially
neighboring PUs A0, A1, B0, B1 and B2.

\[
x_N/S_M == x_P/S_M && y_N/S_M == y_P/S_M \quad (1)
\]

In this way, the AMVP dependencies of neighboring PUs
within the same MER are removed and the ME, including
regular ME and merge mode, can be performed concurrently.

This modification on AMVPCL construction applies only
in MD stage. In order to make sure that the generated bit
stream is standard compliant, when MD of a MER is completed,
the MVP and motion vector difference (MVD) information
need revising. Given the motion parameters that have been
obtained after parallel MD, firstly the standard compliant
AMVPCL is constructed for each non-merge mode PU as the
HEVC standard specifies, and then we select a candidate from
this AMVPCL which gives the minimum MVP index and
MVD total bits cost. The corresponding index and MVD are
recorded and written to the bit stream in encoding stage.

2) CM derivation dependencies: In order to remove the
CM dependencies of current CU on previously coded CUs in
the same MER, we propose that MD for all CUs in the same
MER share a same set of CMs that have been trained up to the
last MER. As shown in Fig. 4 in which only four CUs are
shown in current MER for simplici

encoder. (b) depicts our proposed CM derivation method which means the CMs for all CUs in a MER can be derivated concurrently and so the MD for each CU can be performed independently.

![Figure 4. Context model derivation methods. (a) Method adopted in HM and (b) our proposed method for parallel MD](image)

3) **CM selection dependencies:** In order to improve CABAC coding efficiency, context models are selected adaptively according to spatial information of the current unit. This process is called CM selection or context modeling. For example, in HEVC, context modeling of syntax element `split_cu_flag` of a node in the coding tree depends on current splitting depth and that of the left and up CUs, as shown in figure 5 (a). This kind of CM selection dependencies improves CABAC coding performance but hinders parallel MD in a region because of coding mode dependencies. In order to remove this dependency, similar technique as that used in AMVPCL construction dependencies removing, as depicted in Fig. 5(b). Suppose that the left-top luma position of current CU is `(xC, yC)` and the size of current CU is `nChS x nChS`, the criterion of whether it’s in the same MER with the CU covering coordinate `(xN, yN)` is the same as (1). `(xN, yN)` is `(xC - 1, yC)` and `(xC, yC - 1)` for the left and up CU respectively.

![Figure 5. Our proposed methods to remove CM selection dependencies. (a) dependencies of current CU on the left and up CUs. (b) replace CUs in the same MER with those nearest ones in previously coded MERs.](image)

C. **Highly Parallel MD Method**

With data dependencies that hinders parallel MD removed, our proposed parallel MD method can be presented. The proposed parallel MD method is based on MER and MD for all CUs in the same MER can be fully parallelized. As shown in Fig. 6, suppose that the CTU is 64×64, and without loss of generality, MER is set to 32×32 and the minimum allowed CU is 8×8, each MER residing in current CTU needs mode decision to determine the best coding mode. Our proposed parallelization strategies can be summarized as follows:

1) **Parallel processing among CUs:** For all potential CUs within the same MER, including CUs of same and different splitting depth are computed concurrently, i.e. all nodes in the quadtree perform parallel MD. As depicted in Fig. 6, there are totally $4^0 + 4^1 + 4^2 = 21$ CUs in each MER, and if the MER is set the same size as CTU, this number can increase to 85. Computing so many CUs concurrently will give a high throughput.

2) **Parallel processing within a CU:** For a certain CU, many PU partition modes can be used and each one will give a RD cost with the corresponding coding information after ME and TU splitting computation. There are no explicit dependencies between these PU partition modes thus they can be conducted concurrently and independently.

3) **Parallel ME among PUs:** We propose that all PUs in a CU perform ME concurrently, including merge mode estimation and regular motion estimation. Because all CUs in the same MER are conducting MD concurrently, so actually all PUs in the same MER are run in parallel.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

Based on the HEVC reference software HM10.1 [6], we implement and evaluate our proposed parallel MD method using multithread techniques. We modify the MD module (`xCompressCU` function in HM) to make it parallelized and other encoding stages remain unchanged. The computing platform in our experiments is Tile64 which is a member of TILERA many-core processor family and contains 64 processing units.

A. Implementation

In order to ease the overhead costs of frequent thread creating and destroying, we construct a thread pool at the beginning of encoding process. There is a work queue maintained in which works are placed. An idle thread repeatedly fetches a work from the queue and processes it. In our implementation, we define three kinds of works: MD (xCompressCU in HM) for a particular CU, cost computation (xCheckRDCostInter in HM) for a particular PU mode of a CU, and ME for a particular PU (predInterSearch in HM). Initially the MD work for the topmost CU in current MER is put into the queue. During running process, a work itself may generate and put another work into the queue, as depicted in Fig. 7. Specifically, MD work for a CU may generate four MD works for its four child CUs and may generate several cost computing works for all available PU partition modes of current CU. A cost computing work may generate several ME works for all PUs of current PU partition mode.
B. Experimental Results

To evaluate the efficiency of our proposed parallel MD method, including encoding speed improvement and coding quality degradation, eight standard test sequences are encoded using the HM 10.1 encoder as baseline and our parallelized version. In both these two encoders, intra CU is disallowed in P or B slices. In the parallelized encoder, AMP_MRG function is disabled. The configuration file randomaccess_main is used and MER is set to 64x64. For each video sequence four quantization parameter values are used: 22, 27, 32 and 37, according to [15]. The speed up is derived as (2) in which $\text{Enc\_Time}_b$ and $\text{Enc\_Time}_p$ mean the encoding time consumed by the baseline and our parallelized encoder. Speed up of both MD module and overall encoding process of inter slices are taken into account. BD-rate [16] is used to measure the coding quality degradation. The experimental results are shown in Table I, in which the speed up is obtained by using 36 cores and averaging that of four QPs.

\[ \text{speed up} = \frac{\text{Enc\_Time}_b}{\text{Enc\_Time}_p} \]  

Figure 8 shows the average MD speed up of all sequences using different number of cores (4, 8, 16, 24, 32, 36) with QP 27. Similar results are observed for other QPs. It can be seen that our proposed parallel method behaves good when less than 32 cores are used but the curve gradually becomes flat from this point. This situation may results from inter-core synchronization costs. Because our proposed method works in a low parallel level, it can be used jointly with other high level parallel methods such as WPP, to gain much higher speed up.

<table>
<thead>
<tr>
<th>Table I. Experimental results of our proposed parallel MD method.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>1080p</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>1600p</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

V. CONCLUSION

In this paper, a parallel MD method for HEVC encoding is proposed which works in a MER. We propose efficient algorithms to remove data dependencies that hinder parallel computing in a MER and then we present our proposed highly parallel MD method. Tiny bit-rate increasing and high encoding speed up demonstrate their efficiency.

ACKNOWLEDGEMENT

This work is supported by National Nature Science Foundation of China (61102101, 61272323), National Key Technology Research and Development Program of China (2012BAH06B01), Co-building Program of Beijing Municipal Education Commission.

REFERENCES


Figure 8. MD speed up using different number of cores (QP=27).

Figure 7. The thread pool and work queue in our implementation.