Efficient parallel HEVC intra-prediction on many-core processor

C. Yan, Y. Zhang, F. Dai, J. Zhang, L. Li and Q. Dai

High-efficiency video coding (HEVC) is the state-of-the-art video coding standard, which adopts more complicated and time-consuming intra-prediction (IP) modes. Many-core processors are good candidates for speeding up HEVC IP in the case that HEVC IP can provide sufficient parallelism. Proposed is an efficient parallel framework for HEVC IP. Experiments show that the proposed method dramatically accelerates more than the state-of-the-art parallel method.

Introduction: Intra-prediction (IP) is used to remove the spatial redundancies within one image. High-efficiency video coding (HEVC) provides a highly flexible block splitting manner for IP, which includes three unit concepts [1]: the coding tree unit (CTU), the coding unit (CU) and the prediction unit (PU) (Fig. 1). Each frame is first divided uniformly into non-overlapped square CTUs, which can be recursively split into smaller CUs using a generic quadtree segmentation structure. The CU also has a square shape and can be further split into PUs. PUs are the basic units used for carrying IP information. There are only nine intramodes available for 4×4 luma blocks in H.264/AVC. To enhance the coding efficiency of HEVC, HEVC provides as many as 35 prediction modes for different PUs. If a neighbouring PU is coded, it will be available for the current PU. The current PU may have data dependencies on its neighbouring left, left-down, upper, upper-left and upper-right PUs, whose prediction information may be available for the current PU.

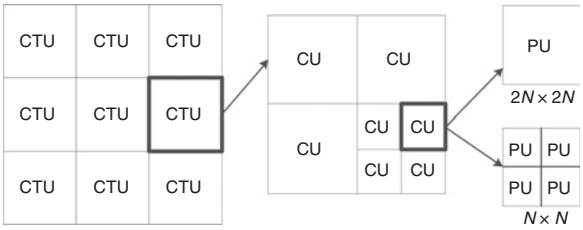

Fig. 1 *Flexible block splitting manner for HEVC IP*

Parallel intra-coding (PIC) [2] is the state-of-the-art parallel proposal for HEVC IP. PIC introduces the concept of the parallel IP unit (PPU), which defines the size of a block that can be coded in parallel. The maximum parallelism of PIC reaches 2, which is not adequate for many-core processors. It is urgently demanded to provide sufficient parallelism for HEVC IP.

Proposed efficient parallel framework for HEVC IP: PIC eliminates the data dependencies among blocks within the same PPU. However, the PPUs and CTUs have to be processed sequentially. We propose an efficient parallel framework for HEVC IP. We first analyse the data dependencies among neighbouring CTUs and use the directed acyclic graph (DAG)-based order to parallelise CTUs (DAGCTU), which exploits the implicit CTU-level parallelism. Then we find that the CTU size influences the parallelism and coding efficiency. On the premise of having satisfying coding efficiency, we select the optimal CTU size by using a support vector machine (SVM) classifier (SVMCTU).

DAGCTU**:** The CTUs are processed in row scanning order, where k denotes the time stamp of the CTU. The data dependencies among neighbouring CTUs are caused by the PUs. When processing the current CTU, the PUs within the current CTU's neighbouring left-down CTU are not coded yet, which will be unavailable for the PUs in the current CTU. Therefore the current CTU has no data dependency on its adjacent left-down CTU. Meanwhile, the PUs in the current CTU's neighbouring left, upper, upper-left and upper-right CTUs are coded. The current CTU has data dependencies on its neighbouring left, upper, upper-left and upper-right CTUs.

We generate a DAG to capture the dependency relationships of CTUs. We first map each CTU in the frame into a point in a two-dimensional (2D) coordinate plane as follows:

$$i = \mathrm{ceil}\left(\frac{k}{W}\right) \tag{1}$$

$$j = k \bmod W \tag{2}$$

where i and j are, respectively, coordinate values of the horizontal axis and vertical axis, W is the horizontal CTU number of the frame and the ceil function returns the value of a number rounded upwards to the nearest integer.

Then we use a DAG to represent the execution flow of the CTUs and the precedence constraints among the CTUs. We mark the DAG as $G = (V, E)$, which consists of a set of vertices V and edges E. Vertices are numbered according to the coordinate value of CTUs in the 2D coordinate plane. For example, vertex $v_{i,j}$ represents the CTU with the coordinate value (i, j). If vertex $v_{i,j}$ is a parent of vertex $v_{m,n}$, vertex $v_{m,n}$ will have data dependency on $v_{i,j}$ and there will exist an edge $(v_{i,j}, v_{m,n}) \in E$. When vertex $v_{i,j}$ is processed, vertex $v_{i,j}$ and edge $(v_{i,j}, v_{m,n})$ will be removed from the DAG. Precedence constraint means that when the in-degrees of some vertices are zero, these vertices can be processed in parallel. To parallelise the vertices, it is important to record and update the in-degrees of all the vertices. We obtain the initial value of the in-degrees by the adjacency matrix. We generate the adjacency matrix $\boldsymbol{A}$ of the DAG as follows:

$$\boldsymbol{A}_{(i,j),(m,n)} = \begin{cases} 1, & (v_{i,j}, v_{m,n}) \in E \\ 0, & \text{otherwise} \end{cases}$$
$$\text{s.t.} \quad 1 \le i, m \le H \quad 1 \le j, n \le W \tag{3}$$

where H is the vertical CTU number of each frame, and $\boldsymbol{A}$ is a 2D matrix.

The initial in-degree $\boldsymbol{D}_{m,n}$ of vertex $v_{m,n}$ in the DAG can be summarised as follows:

$$\boldsymbol{D}_{m,n} = \sum_{i=1}^{H} \sum_{j=1}^{W} A_{(i,j),(m,n)}$$
$$\text{s.t.} \quad 1 \le m \le H, \quad 1 \le n \le W \tag{4}$$

where $\boldsymbol{D}$ is a 2D matrix, which represents the initial state of the in-degrees of the DAG.

SVMCTU**:** The processing times of CTUs are different from each other. To analyse the maximum parallelism conveniently, we suppose all the processing times of CTUs are the same. The maximum parallelism of CTU ($\mathrm{MP}_{\mathrm{CTU}}$) can be calculated as follows:

$$\mathrm{MP}_{\mathrm{DAGCTU}} = \min\left(\mathrm{ceil}\left(\frac{W}{2}\right), H\right) \tag{5}$$

$$W = \frac{W_{\mathrm{f}}}{C} \tag{6}$$

$$H = \frac{H_{\mathrm{f}}}{C} \tag{7}$$

where W_{f} and H_{f} are, respectively, the horizontal and vertical pixel numbers of each frame. C is the length of the CTU.

Fig. 2 *Select optimal CTU size by using SVM classifier (SVMCTU)*

The maximum parallelism of our method is much more than that of PIC. When the resolution of CTU is 16×16, the maximum parallelism of our proposed method reaches 60 and 80 for 1920×1080 and 2560×1600 video sequences, respectively. We also find that the CTU size influences the parallelism and coding efficiency greatly. The smaller the CTU size is, the more the parallelism is, whereas at the same time the CTU size also affects the coding efficiency significantly quite the opposite way. On the premise of having satisfying coding efficiency, we select the optimal CTU size using a SVM classifier (Fig. 2). In particular, we extract useful coding information of current video sequences to compose feature vectors (FVs). We form the FV as follows:

$$\mathrm{FV} = (W_{\mathrm{f}}, H_{\mathrm{f}}, \mathrm{QP}, C) \tag{8}$$

Besides the CTU size, the coding efficiency is also closely related to other coding information, such as QP and the resolution of the frame [1]. Large CTU size can be very efficient for a high resolution, large QP video sequence. We obtain W_f, H_f, QP and C from the pre-encoded HEVC files, which are generated to the FV. For a video sequence, W_f, H_f, QP and C are specified in the configuration profile. We change the CTU size and form different FVs. Then we send the FVs to a pre-trained classifier. The output of the classifier is a class label 0 or 1 corresponding to whether we obtain the satisfying coding efficiency. 1 indicates we obtain the satisfying coding efficiency. The classifier is trained as follows. For each frame, the FV is recorded along with the corresponding class label, which is jointly called a training instance. Each instance is normalised to be in a range between 0 and 1, which is used as an input feature to the linear SVM classifier. Extensive instances are input into the classifier to train it. We mark the FVs as follows:

$$S_{FV} = S_{FV0} \cup S_{FV1} \tag{9}$$

where S_{FV} is the complete set of FVs. Set S_{FV0} and S_{FV1} are the subsets of S_{FV}. Set S_{FV0} is the FVs that do not have satisfying coding efficiency. Set S_{FV1} is the FVs that have satisfying coding efficiency. Among all the FVs that belong to set S_{FV1}, we select the minimum CTU size. According to (5)–(7), we know that if we select the smaller CTU size, the parallelism will be higher. On the premise of having satisfying coding efficiency, we obtain the maximum parallelism using the SVM classifier.

Table 1: BD-rate performance compared with HM12.0

Sequences	PIC (%)	DAGCTU $C = 16$ (%)	Proposed (%)
Vidyo1	2.0	7.4	1.2
Vidyo3	1.8	8.7	1.6
SlideShow	1.4	17.9	1.8
ParkScene	1.2	2.8	1.6
BasketballDrive	1.3	11.2	1.0
BQTerrace	0.6	4.1	2.2
Traffic	1.2	3.5	0.9
SteamLocomotive	0.2	4.7	1.1
Nebuta	0.2	6.7	1.1
Average	**1.1**	**7.4**	**1.4**

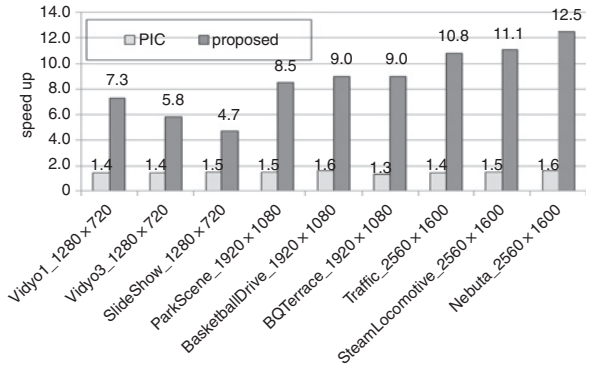

Fig. 3 *Speed up of PIC and proposed method compared to serial execution using 64 cores, QP = 37*

Results: The experiment platform is Tile64, which contains 64 processing cores [3]. Experiments were carried out on the HEVC reference software HM12.0 and a well-known library libsvm3.12 [4] for the SVM. Nine standard test sequences 'PeopleOnStreet', 'Kimono', 'Cactus', 'BasketballDrill', 'BasketballPass', 'BasketballDrillText', 'ChinaSpeed', 'Vidyo4' and 'SlideEditing' are used for training the classifier. Compared with HM12.0, if the BD-rate loss of a frame is more than 1.5%, the class label will be set as 0 (unsatisfying quality). The FV of a frame and the corresponding class label is jointly called a training instance. We randomly select 2000 instances which are input into an SVM to train it. The BD-rate performances of all the methods compared with HM12.0 are shown in Table 1. The positive number means the coding efficiency loss. From (5) to (7), if we directly select the smallest CTU size (equal to 16), the parallelism of DAGCTU will be the highest. However, DAGCTU ($C = 16$) leads the BD-rates to a serious increase of 7.4% on average, whereas PIC and our proposed method have little effect on the coding efficiency. Fig. 3 shows the speed up of PIC and the proposed method compared with serial execution. Our proposed method accelerates a lot more than PIC. Compared with serial execution, our proposed method achieves averagely more than eight times speed up.

Conclusion: On the premise of having satisfying coding efficiency, we propose an efficient parallel framework for HEVC IP. The performance of the proposed scheme is confirmed by experiments.

Acknowledgments: This work is supported by the National Key Technology Research and Development Program of China (2012BAH06B01) and the National Nature Science Foundation of China (61102101, 61272323, 61379084).

20 February 2014
doi: 10.1049/el.2014.0611
One or more of the Figures in this Letter are available in colour online.

C. Yan, Y. Zhang, F. Dai, J. Zhang and L. Li (*Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, People's Republic of China*)

E-mail: zhyd@ict.ac.cn

Chenggang Yan and Q. Dai: Also with the Department of Automation, Tsinghua University, Beijing, China

References

1 Sullivan, G.J., *et al.*: 'Overview of the high efficiency video coding (HEVC) standard', *IEEE Trans. Circuits Syst. Video Technol.*, 2012, **22**, (12), pp. 1649–1668
2 Zhao, J., *et al.*: 'CE6.d: Parallel Intra Coding'. JCTVC-F605, July, 2011
3 Zhang, Y., *et al.*: 'Efficient parallel framework for H.264/AVC deblocking filter on many-core platform', *IEEE Trans. Multimedia*, 2012, **14**, (3), pp. 510–524
4 Libsvm Version 3.12. Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/