Parallel deblocking filter for HEVC on many-core processor

Chenggang Yan, Yongdong Zhang, Feng Dai, Xi Wang, Liang Li and Qionghai Dai

High-efficiency video coding (HEVC) is the next generation standard of video coding. The deblocking filter (DF) constitutes a significant part of the HEVC decoder complexity. A three-step parallel framework (TPF) is proposed for the H.264/AVC DF, which is also suitable for HEVC except the third step. The third step of the TPF is replaced with a directed acyclic graph-based order. Experiments show that the proposed method dramatically accelerates more than the state-of-the-art parallel method.

Introduction: At the beginning of high-efficiency video coding (HEVC) standardisation, the processing order of the deblocking filter (DF) is similar to that of the H.264/AVC DF [1]. As shown in Fig. 1, each frame in HEVC is divided into coding tree units (CTUs), which can be recursively split into smaller coding units (CUs) by using a generic quadtree segmentation structure. As shown in Figs. 1*a* and *b*, the DF follows the processing order of the CTU and the CU, the numbers of which indicate the coding order. CUs can be further split into prediction units (PUs) and transform units (TUs). Deblocking filtering takes place in vertical and horizontal edges of PUs and TUs. Each edge consists of one or several parts, whereas a part is of the size of 8×8 samples for the luma component and of 4×4 samples for the chroma component. As shown in Fig. 1*c*, if one CU has the size of 16×16 samples for the luma component, this CU will have four vertical edges v_1, v_2, v_3 and v_4 and four horizontal edges h_1, h_2, h_3 and h_4. Within each CU, the vertical edges are processed before the horizontal edges. The order of filtering the vertical and horizontal edges is from top to bottom and from left to right.

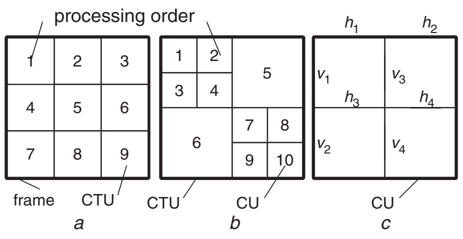

Fig. 1 *Processing order of DF*

a Processing order of CTU within example frame
b Processing order of CUs within example CTU
c Processing order of edges within example 16 × 16 CU

The data dependencies of the DF are caused by the DF's three subtasks: edge discrimination (ED), boundary strength computation (BSC) and filtering. ED determines whether or not to adopt filtering. BSC decides between a strong and weak filtering. The result of each filtering step may be used as an input to subsequent ED and filtering. To make HEVC deblocking filtering more 'parallel-friendly', [2] proposed an efficient order-changed parallel method (OCPM), which changes the order of filtering. The filtering does not follow the processing order of the CTU and CUs as before. Within each frame, all the vertical edges are filtered in parallel before all the horizontal edges. However, the OCPM has not alleviated the data dependencies between BSC and filtering. BSC cannot be processed in parallel before filtering. Meanwhile, the OCPM has the load imbalance problem because the DF of every edge is not the same.

On the premise of keeping the order of filtering unchanged, our previous work on a three-step parallel framework (TPF) for H.264/AVC DF [3] is also suitable for that of HEVC except the third step. We consider ED and filtering as EDF. In the first step, we divide the entire DF process into two parts and parallelise BSC before EDF, which increase the parallelism. In the second step, we use the Markov empirical transition probability matrix and the Huffman Tree to alleviate the load imbalance problem of BSC. However, in the last step, the independent pixel connected area parallelisation is not suitable for HEVC EDF because the unit representation of HEVC is greatly different from that of H.264/AVC. In this Letter, we propose a directed acyclic graph (DAG)-based order to parallelise HEVC EDF.

Proposed DAG-based order to parallelise EDF

Classify filtered pixels into two sets: The details of the filtered pixels and their respective evaluated pixels can be found in [1]. Fig. 2 shows an example of the classified filtered pixels for a 16×16 CU. The white pixels will not be filtered neither by vertical filtering nor by horizontal filtering. The once filtered pixels are located in the middle of edge filtering. The twice filtered pixels are located in the margin of edge filtering. We mark the filtered pixel sets as follows:

$$D_c = D_{\text{once}} \cup D_{\text{twice}} \tag{1}$$

where D_c is the complete set of filtered pixels. D_{once} and D_{twice} are the subsets of D_c. Set D_{once} consists of pixels that will be filtered only once by the vertical or the horizontal edge. Set D_{twice} consists of pixels that will be filtered by both the vertical and horizontal edges. The EDF of D_{once} has no data dependency on D_{twice}, so D_{once} can be processed in parallel before D_{twice}.

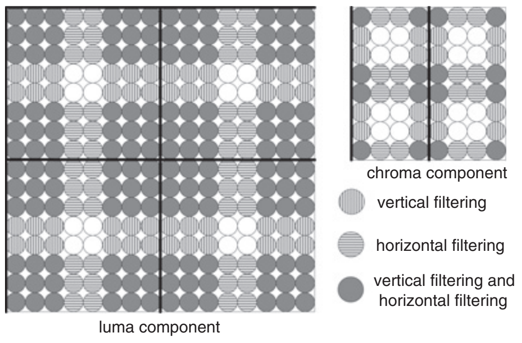

Fig. 2 *Example of classified filtered pixels for 16 × 16 CU*

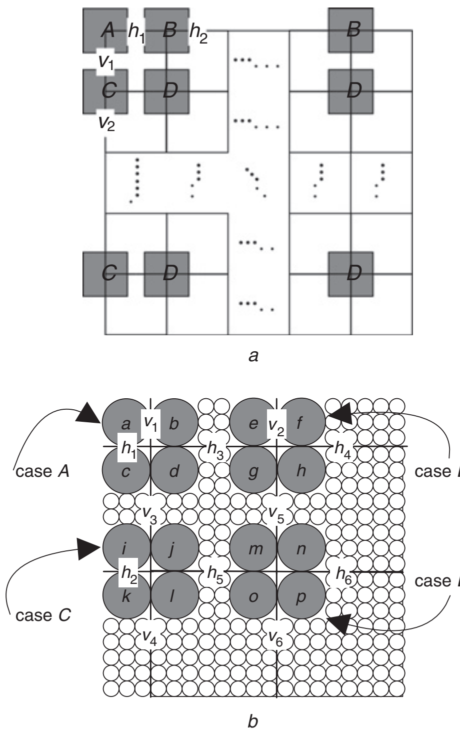

Fig. 3 *Dividing D_{twice} into four cases marked as A, B, C and D*

a Dividing D_{twice} into four cases according to filter conditions
b Pixels sets are marked as *a*, *b*, …, *n*, *p* for luma component

Classify D_{twice} *into four cases:* Fig. 3*a* shows an example of a CU with the size $N*N$. N indicates horizontal and vertical numbers of eight consecutive pixels for the luma component and four consecutive pixels for the chroma component. v_1, v_2, …, $v_{(n*n)}$ are vertical filtering and h_1, h_2, …, $h_{(n*n)}$ are horizontal filtering. The grey areas represent the twice filtered pixels which belong to D_{twice}. We divide those areas into four cases marked as A, B, C and D according to the filtering conditions. Those pixel sets are 6×6 samples for the luma component and 2×2 samples for the chroma component. Cases A, B and C, respectively, are located in the upper-left, upper and left borders of the CTU. The pixels in these cases have similar filtering conditions.

Parallelise D*twice* *with DAGs:* The dependencies analysis of D_{twice} between the luma and the chroma components is similar. To facilitate the subsequent analysis, we just analyse the luma component and mark the pixel sets in four cases A, B, C and D as a, b, …, n, p (Fig. 3*b*). Each case has four pixel sets and each pixel set is 3×3 samples. For example, case A has the pixel sets a, b, c and d. The dependencies among pixel sets are represented as DAGs (Fig. 4). Each vertex designates a task that a specific pixel set is filtered by a particular edge. For example, the vertex 'av_1' means that the pixel set 'a' is filtered by the edge 'v_1'. The data dependencies in four cases are different from each other. The dependencies among the EDF tasks correspond to the edges in the DAGs. For example, task 'ah_1' has a data dependency on task 'av_1'. The DAGs for the chroma component are the same as those for the luma component. The only difference is that the pixel sets a, b, …, n, p represent a 1×1 sample for the chroma component. We decompose the process of tasks in DAGs into four stages. When the in-degrees of some vertices in the DAGs are zero, those vertices are processed in parallel. In the following stage, those processed vertices will be removed from DAGs, and the data dependencies will be updated. We can obtain the similar process of D_{twice} for the chroma component. We parallelised the D_{twice} with DAGs. The dependencies among the filtered pixels did not change.

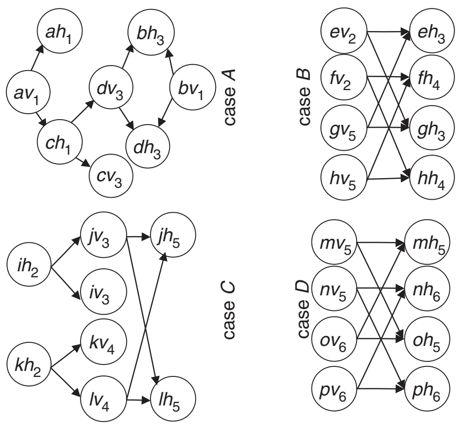

Fig. 4 *Dependencies among pixel sets are represented as DAGs*

Results: Experiments were carried out on HM7.0 under the profile 'randomaccess_main'. The experiment platform was Tile64, which contains 64 processing cores [3]. Ten standard video test sequences with 64 frames were used. Fig. 5 shows the speedup of all the methods compared to serial execution. 'Proposed' means the improved TPF, the third step of which is replaced with a DAG-based order. Compared with the OCPM, our proposed method achieves on average more than two times speedup. Meanwhile, the proposed method improves the coding efficiency, which achieves an average BD-rate reduction of 0.31% for the Y component.

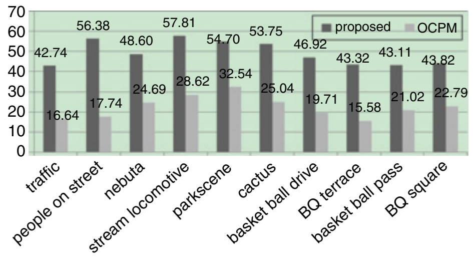

Fig. 5 *Speedup of OCPM and proposed method compared to serial execution using 64 cores, QP = 32*

Conclusion: The OCPM is proposed to change the order of DF and to make DF 'parallel-friendly'. On the premise of keeping the order of DF unchanged, we replace the third step of the TPF with a DAG-based order. The performance of the proposed scheme is confirmed by the experiments.

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

© The Institution of Engineering and Technology 2014
8 October 2013
doi: 10.1049/el.2013.3235
One or more of the Figures in this Letter are available in colour online.

Chenggang Yan, Yongdong Zhang, Feng Dai, Xi Wang and Liang Li (*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, People's Republic of China*)

E-mail: zhyd@ict.ac.cn

Qionghai Dai (*Tsinghua University, Beijing, People's Republic of China*)

Chenggang Yan: Also with Tsinghua University, Beijing, People's Republic of China

References

1 Sullivan, G.J., and Ohm, J.-R.: 'Recent developments in standardization of high efficiency video coding (HEVC)'. Proc. SPIE, San Diego, CA, USA, August 2010, vol. 7798, pp. 30–36
2 Ikeda, M., Tanaka, J., and Suzuki, T.: CE12 Subset2, 'Parallel deblocking filter'. JCTVC 5th meeting JCTVC-E181, Geneva, CH, March 2011
3 Zhang, Y., Yan, C., and Dai, F., *et al.*: 'Efficient parallel framework for H.264/AVC deblocking filter on many-core platform', *IEEE Trans. Multimed.*, 2012, **14**, (3), pp. 510–524