PARALLEL DEBLOCKING FILTER FOR H.264/AVC IMPLEMENTED ON TILE64 PLATFORM

Chenggang Yan[1,2] , Feng Dai[1], Yongdong Zhang[1], Yike Ma[1,2], Licheng Chen[1,2], Lingjun Fan[1,2], Yasong Zheng[1,2]
(1 Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190)
(2 Graduate University of Chinese Academy of Sciences, Beijing, 100049)
Email:{yanchenggang,fdai,zhyd,ykma,chenlicheng,fanlingjun,zhengyasong}@ict.ac.cn

ABSTRACT

For the purpose of accelerating deblocking filter, which accounts for a significant percentage of H.264/AVC decoding time, some researchers use multi-core platforms to achieve the required performance. We study the problem under the context of many-core systems. Parallelization of deblocking filter on many-core platform is challenging not only because deblocking filter has complicated data dependencies which provides insufficient parallelism for so many cores but also because parallelization may have significant synchronization overhead. We present a new method to exploit the implicit parallelism and reduce the synchronization overhead. We apply our implementation to the deblocking filter of the H.264/AVC reference software JM15.1 on Tile64 platform. The proposed method achieves up to 817%, 604% and 532% speedup for CIF, SD and HD videos compared to the well-known wavefront method using 62 cores, respectively.

Keywords—H.264/AVC, Deblocking filter, Parallel algorithm, Tile64

1. INTRODUCTION

H.264/AVC is the newest coding standard which brings a high efficient video compression method to the multimedia industry [1]. Due to its compression efficiency, it has been applied to several important applications including video telephony, video storage, broadcast, video streaming, HD-DVD. An in-loop deblocking filter has been adopted by H.264/AVC, though provides 10–15% bit rate saving, brings heavy computation [2]. For high definition videos, Chen et al. [3] shows that the deblocking filter contributes 38% computation time. Therefore, it is important to improve the H.264/AVC deblocking performance.

With the development of IC technologies, multi-core processors are commonplace [4-7] and many researchers have tried to parallelize the deblocking filter in different data-level using multi-core platforms. The wavefront method is a commonly used macroblock(MB)-level data partition method [8, 9]. This scheme could achieve good load balancing if a dynamic scheduler assigns MBs to different processors, which needs nonnegligible synchronization. Meanwhile it has little parallelism at the beginning and at the end of processing a frame, which is not believed to be adequate for the increasing number of cores. K.-H. Chen et al. [10] presented an edge-level data partition which is not suitable to software implementation. Sung-Wen Wang et al. [11] observed that each filtering step only affects a limited region of pixels and presented a pixel-level parallelism method (PLPM). It has a sustained and sufficient data parallelism. The main disadvantage is that the performance is easily affected by the synchronization overhead when the number of cores increases. Now we are entering into many-core ages such as TILERA many-core systems and it is urgent demands for a better method.

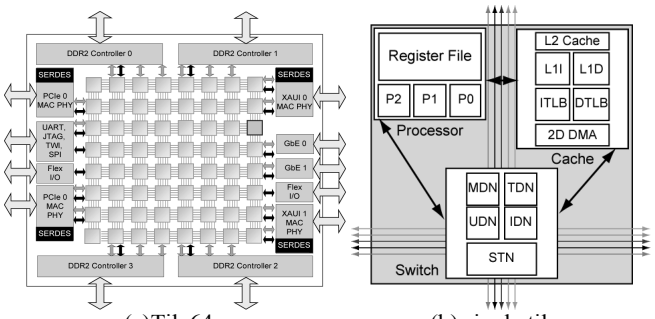
All the methods introduced directly discuss the data partition scheme for the entire filtering process. In this paper we review the deblocking filter process which is composed of three parts [12], BS computation, ED and filtering. We study the dependence between the three parts in edge level and first process BS computation, which increases the parallelism. Then we change the edge order within each MB for ED and filtering, which leads to a significant change of data dependencies between neighboring MBs. The proposed method significantly improves the performance overhead, while the decoded video quality is not changed.

The remainder of this paper is organized as follows. First, we briefly introduce the Tile64 platform in Section 2. An overview of H.264/AVC deblocking filter and the wavefront method are described in Section 3. The proposed deblocking filter speedup techniques are presented in Section 4. Experimental results and analysis are given in Section 5. Finally, we conclude this letter in Section 6.

2. SHORT OVERVIEW OF TILE64 PLATFORM

Centralized monolithic processor designs, such as single core and shared bus structures, are not scaling, leading to multi-core design becoming the norm [5-7]. Research highlights the benefits of mesh networks connecting these cores [13, 14], which can solve the multi-core scalability problem.

978-1-61284-350-6/11/$26.00 ©2011 IEEE



(a)Tile64 (b) single tile

Fig. 1. Tile64 block diagram and Single tile block diagram figure

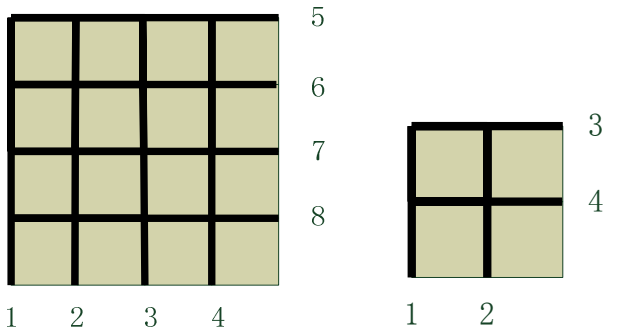
TILERA many-core processor family features devices with 16 to 100 identical processor cores. Fig.1a illustrates a block diagram with Tile64 arranged in an 8×8 array. These cores run at 700MHz and connect through a scalable and high-speed 2D mesh network. Using four 72-bit 800MHz DDR2 interfaces, the chip peak memory bandwidth is over 25GB/s. Two PCI-e ×4 interfaces, two XAUI interfaces and two RGMII interfaces provide over 40Gb/s of I/O bandwidth. The low-speed interfaces provide seamless integration with a variety of systems [15].

Each tile processor (Fig.1b) is a full featured processor and provides a three-way Very Long Instruction Word (VLIW) architecture allowing up to 3 instructions per cycle. A single core contains a 16KB L1 cache, 8KB of data L1 cache, and 16KB of combined L2 cache. Each core also has its own DMA engine and Translation Lookaside Buffer (TLB) which allow memory virtualization and the ability to run a modern O/S on each core [16].

3. DEBLOCKING FILTER AND THE WAVEFRONT METHOD

H.264/AVC video coding standard adopts an in-loop deblocking filter which removes block-edge artifacts that are produced by block transformation and block motion compensation. "In-loop" implies any inappropriate modification of the in-loop deblocking filter causes serious error propagation to succeeding frames.

The deblocking filter stage is applied to each luminance and chrominance 4x4 block edge within one MB (Fig.2). All edges are processed from left to right and from top to bottom. Vertical boundary edges are processed first. Usually MBs in a frame are processed in scan order during deblocking filter which should be followed to ensure both the encoder and the decoder having the same result. The current deblocking MB has dependencies on its adjacent left, upper, and upper-right MBs(Fig.3). When deblocking the current MB, the left, upper, and upper-right MB should have completed processed.



(a) 16x16 Luminance data (b) 8x8 Chrominance data

Fig. 2. Edge filtering order in each MB

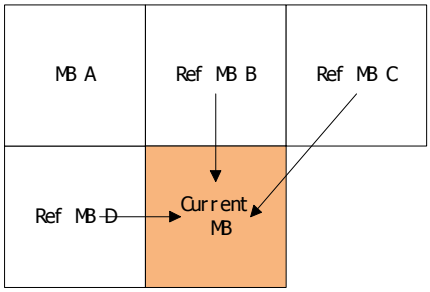


Fig.3. Data dependencies between neighboring MBs in deblocking filter

1	2	3	4	5
3	4	5	6	7
5	6	7	8	9

Fig.4. A wavefront data partition example: each rectangular indicts an MB

The wavefront method is widely used for data partition [8] which processes the data in wavefront order (Fig.4). The MBs are processed according to their numbers which indicate the time stamp. MBs with the same numbers are processed concurrently. Therefore, some researchers adopt this method to parallel deblocking filter. In this paper we call it wavefront MB-level deblocking filter method and wavefront method for short. Fig.5 shows an example of wavefront method for a video frame. At time stamp 4, there are two independent MBs. The maximum number of independent MBs(MNIM) is first available at time stamp 9 and it depends on the resolution. Table1 states the MNIM using the wavefront method. The column "MBs" indicates the horizontal and vertical numbers of MBs in a frame.

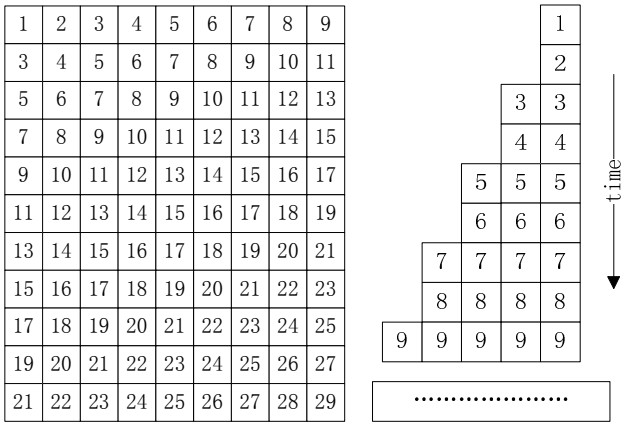


Fig.5. An example of wavefront MB-Level deblocking filter for a video frame

There are some disadvantages for this kind of MB-level parallelism. The first disadvantage is that there are few numbers of independent MBs at the start of deblocking filter. Fig.5 shows that the MNIM increases one at two stamps addition. What's more, when the number of cores is enough, this method has not discovered the potential capacity of the MNIM. The MNIM (Table 1) for every resolution is smaller than the number of cores on Tile64 platform.

Another drawback of the wavefront method is that we should not ignore the communication between the MBs. Before the current MB is processed, it has to communicate with up to three MBs processed on other cores. The amount of synchronization overhead incurred in wavefront method can't be ignored.

Table 1. The MNIM using the wavefront method

Format	Resolution	MBs	MNIM
QCIF	176x144	11x9	5
CIF	352x288	22x18	9
SD	720x576	45x36	18
HD	1280x720	80x45	23
FHD	1920x1088	120x68	34

4. THE PROPOSED DEBLOCKING FILTER SPEEDUP METHOD

4.1. Parallel BSC before ED and filtering

We first review the deblocking filtering of H.264/AVC. For every 4x4 block edge, because of complex "if" instructions we split the deblocking filter into three parts, BS computation (BSC), ED and filtering [12]. Fig. 6 shows the dependence between them for deblocking filter in every edge. BSC is determined according to the coding information, such as coding mode and coded residues, which has nothing to do with the output of other parts. BS value determines the strength of filtering. No filtering is needed with a BS value of 0, whereas the strongest filtering is performed with a BS value of 4 and normal filtering with other values. ED distinguishes the blocking artifacts (BA) from the sharpness of the content (SC), which determines whether we adopt the filtering. Filtering is then applied according to the value of BS. So filtering and ED are closely related to the result of BSC. Meanwhile filtering is influenced by the output of ED.

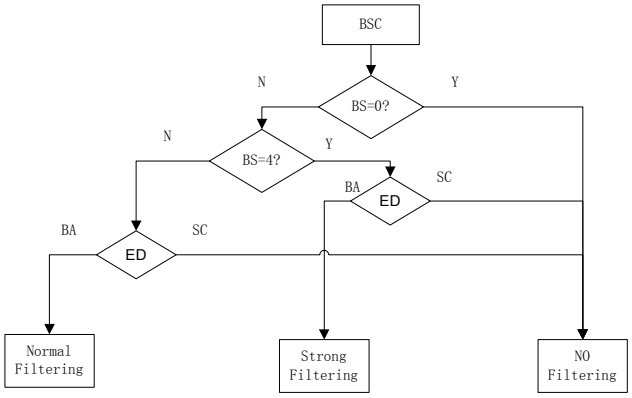


Fig.6. The dependence between the three parts for deblocking filter in every edge

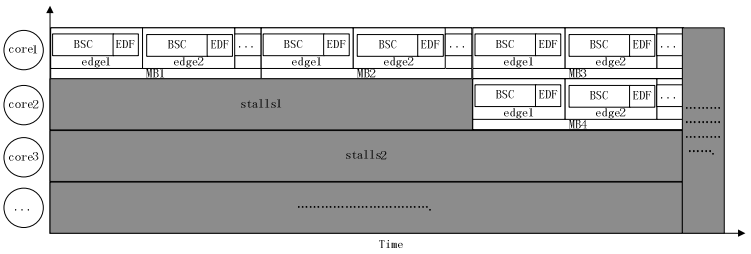


Fig.7. An example of wavefront method considering BSC and EDF in every edge for a video frame

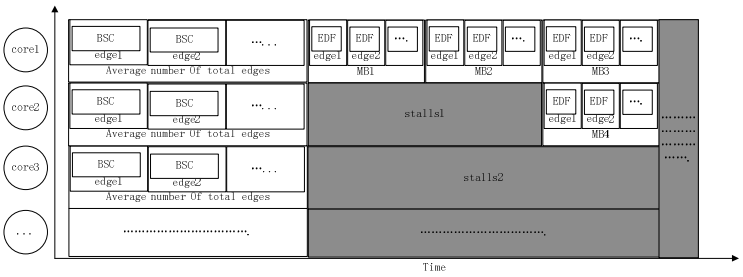


Fig.8. Parallel BSC firstly and apply wavefront method to EDF for a video frame

For each edge, we consider ED and filtering as EDF. Fig.7 shows an example of wavefront method for a video frame considering BSC and EDF. Every MB contains eight edges of deblocking filter for 16x16 luminance data and four edges of deblocking filter for 8x8 chrominance data. At

first, many MBs have to wait for completion of others. For example, MB4 in core2 can be processed when MB1and MB2 have been processed in core1. From [17] we find that most of the filtering computation resources are spent on BSC which is independent from EDF. So we can parallel BSC firstly and apply wavefront method to EDF (Fig.8). To parallel BSC, we calculate the amount of edges and assign them to cores on average. Each core has almost the same amount of BSC and all the cores are at full capacity. Though the number of independent MBs at the start of EDF is the same of wavefront method, but the stalls become smaller. For example, the stalls1 and stalls2 in Fig.8 are smaller than that in Fig.7. But if we apply wavefront method to EDF, the communication\synchronization between the MBs still exists and can't be ignored. This problem will be discussed in next section.

4.2. Change the order of edges in each MB for EDF

As described before, the order of deblocking filter should not change arbitrarily which can influence the results of H.264/AVC decoder. MBs can be processed out of scan order provided these dependencies are satisfied. The wavefront method studies the dependencies in MB-Level for the entire deblocking filter process. In this section we study EDF in edge-level.

Fig.9 shows the data dependencies between the current luminance MB and edges in neighboring MBs for EDF. As shown in Fig. 9, the four left neighbour 4x4 blocks of current MB are named L1 – L4; the four upper neighbour 4x4 blocks of current MB are named T1 – T4; V1-V7 are the vertical edges; H1-H6 are the horizontal edges; V1, V5, V7, H1 and H5 are edges between the MBs. Current MB can be processed for EDF when the neighbor blocks L1-L4 and T1-T4 don't change. So if V1-V5 which influence T1-T4 are processed for EDF, H1-H4 and V6 which affect L1-L4 are conducted as well, then the current MB can be processed.

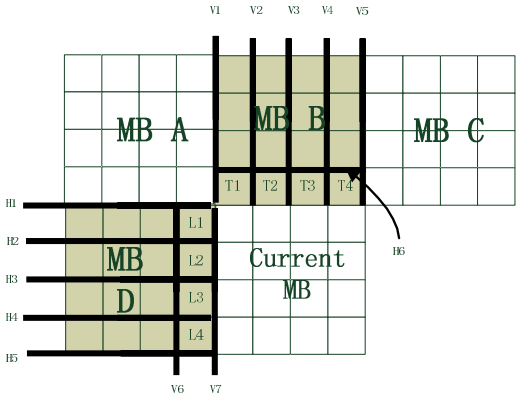
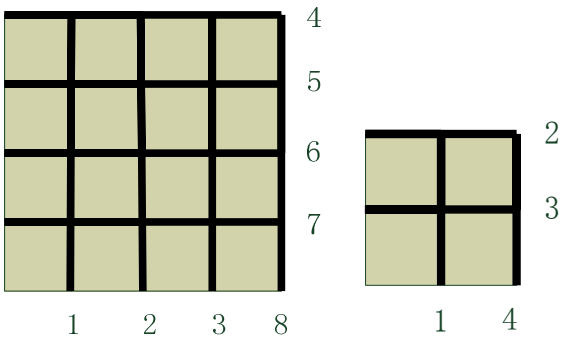


Fig.9. Data dependencies between the current Luminance MB and neighboring edges for EDF



(a) 16x16 Luminance data (b) 8x8 Chrominance data

Fig.10. Edge order for EDF in each MB

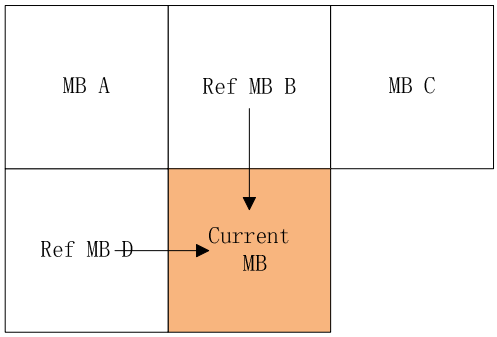


Fig.11. Data dependencies between neighboring MBs for EDF

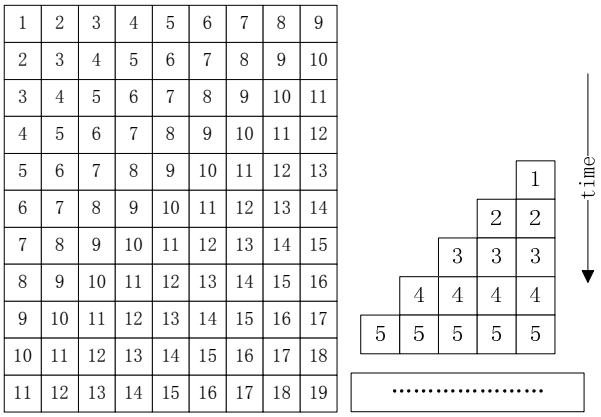


Fig.12. An example of our proposed MB-Level EDF for a video frame

Based on this observation we change the order of edges for EDF within an MB (Fig.10) but not the overall order in one frame. The data dependencies between neighboring MBs for EDF decreases (Fig.11). The current MB only has dependencies on its adjacent left and upper MBs.

Assuming every MB adopts the order of edges as we proposed for EDF, we can have a new scheme of data partition. Fig.12 shows an example of our proposed MB-

level EDF for a video frame. At time stamp 4, there are four independent MBs. The MNIM depends on the resolution as the wavefront method. Table2 states the MNIM using our proposed scheme. Compared with the wavefront method, there are more numbers of independent MBs at the start of EDF. Fig.12 shows that the MNIM increases one at one stamp addition. What's more, when the number of cores is enough, the MNIM is bigger (Table 2).

Table2. Maximum number of independent MBs using our proposed MB-level EDF

Format	Resolution	MBs	MNIM
QCIF	176x144	11x9	9
CIF	352x288	22x18	18
SD	720x576	45x36	36
HD	1280x720	80x45	45
FHD	1920x1088	120x68	68

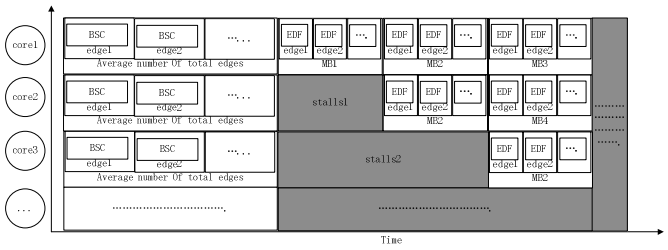


Fig.13. Proposed method for deblocking filter

Fig.13 shows the proposed method for deblocking filter. Paralleling first BSC before EDF exploits the implicit parallelism. Then we change the order of edges in each MB for EDF and proposed a MB-level EDF schedule, which reduces the amount of synchronization and increases the parallelism. In the next section, we will compare the performance of our method with other methods from actual experiments.

5. EXPERIMENTAL RESULTS

In this section, we compare our scheme on deblocking filter with the other methods. To compare them, we adopted a decoder migrated from H.264/AVC reference software JM15.1 without any optimization as baseline profile. The decoder includes the stages of entropy decoding, de-quantization, inverse integer transform, intra prediction and motion compensation. We implemented the parallel algorithm on Tile64 platform which was described in section 2. The input videos in our experiments contain a list of standard test sequences named mobile, highway, hall, mother-daughter, riverbed ,rush_hour pedestrian, container and blue_sky with three resolution levels, 1280x720(HD), 720x576(SD) and 352x288(CIF) which are encoded by the reference software JM15.1.

Fig.14 shows the speedup of wavefront method and our method compared to deblocking filter of JM15.1. The speedup in our method increases more quickly than that in wavefront method because our method has more parallelism and less synchronization. As the number of cores increases to a certain extent, the speedup degrades in wavefront method because of synchronization overhead. The performance of our method is much better than that in wavefront method. We also find that the synchronization in of our method can't be neglected as well (Fig.14c). When the number of cores is larger than 50, the performance decreases a little.

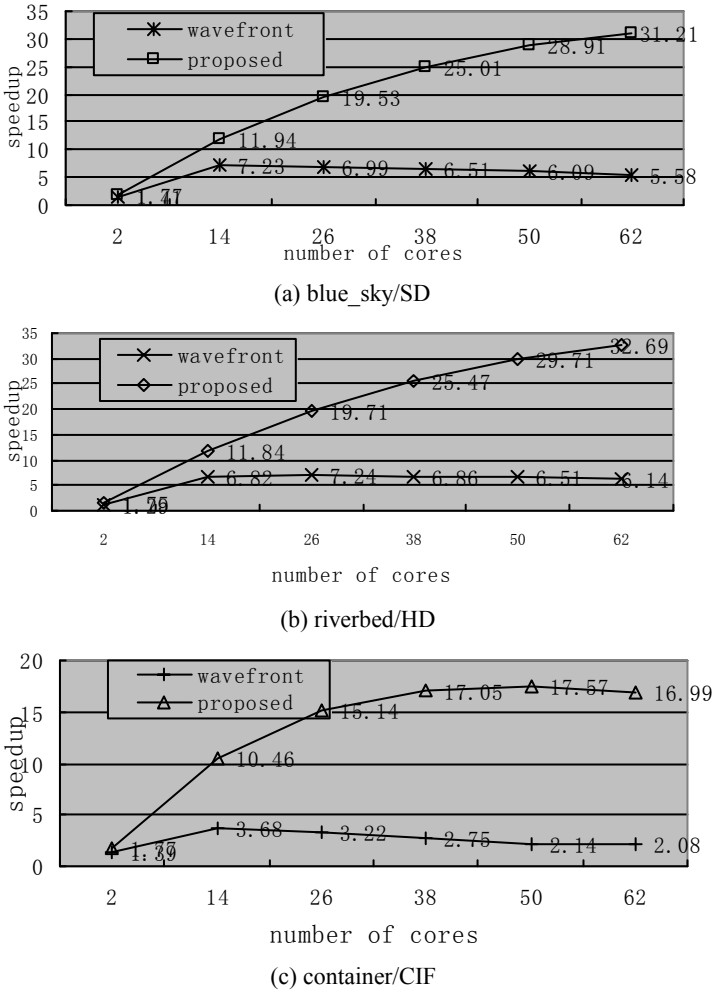


Fig.14. The speedup of wavefront method and our method compared to deblocking filter of JM15.1

Table 3 compares the execution time between wavefront method, PLPM method and our method using 62 cores. The average execution time between them is summarized in Table 4. Our method achieves up to 817%, 604% and 532% speedup for CIF, SD and HD videos compared to wavefront method using 62 cores, respectively. We also observe a significant speedup compared to PLPM method, up to 680%, 502% and 429% for CIF, SD and HD videos using 62 cores, respectively.

Table3. The execution time between different methods using 62 cores

sequences	Format	wavfront(ms)	PLPM(ms)	proposed(ms)
blue_sky	SD	47.68	38.51	8.53
blue_sky	HD	65.09	54.24	15.49
pedestrian	SD	43.82	36.52	7.61
pedestrian	HD	64.68	53.90	14.34
riverbed	SD	36.32	30.27	6.02
riverbed	HD	61.65	49.83	11.59
rush_hour	SD	43.90	36.58	7.41
rush_hour	HD	66.07	55.06	14.18
container	CIF	26.35	21.34	3.23
hall	CIF	26.18	21.82	3.39
highway	CIF	25.08	20.90	3.23
mobile	CIF	23.92	19.93	2.93
mother-daughter	CIF	26.62	22.18	3.41

Table4. The average execution time between different methods using 62 cores

Format	wavefront (ms)	PLPM(ms)	proposed (ms)
CIF	25.63	21.23	3.24
SD	42.93	35.47	7.39
HD	64.37	53.26	13.90

6. CONCLUSION

The deblocking filter is considered as the most computationally expensive part of the H.264/AVC decoder. A new method is proposed to speedup deblcoking filter and achieves good performance on Tile64 platform.

7. ACKNOWLEDGMENTS

This work was supported by the National Nature Science Foundation of China (60802028, 60873165), National High Technology and Research Development Program of China (863 Program, 2009AA01A403), National Basic Research Program of China (973Program, 2007CB311100), Co-building Program of Beijing Municipal Education Commission, Beijing New Star Project on Science & Technology (2007B071).

8. REFERENCES

[1] Joint Video Team of ITU-T and ISO/IEC JTC1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050(2003).

[2] List, P., Joch, A., Lainema, J., Bjntegaard, G., & Karczewicz, M: Adaptive deblocking filter. IEEE Transactions on Circuits and Systems for Video Technology, 13(7), 614–619 (2003).

[3] Chen, T.C., Fang, H.C., Lian, C.J., Tsai, C.H., Huang, Y.W.,Chen, T.W., et al.: Algorithm analysis and architecture design for HDTV applications-a look at the H.264/AVC video compressor system. IEEE Transactions on Circuits and Devices Magazine, 22(3), 22–31(2003).

[4] Wenying Wang, Dongming Zhang, Yongdong Zhang and Jintao Li: “parallel spatial matching for object retrieval implemented on GPU,” Proc. of the Intl. Conf. on Multimedia & Expo (ICME)(2010)

[5] J. Friedrich, B. McCredie, N. James, et al.: “Design of the Power6TM Microprocessor”, ISSCC Dig. Tech. Papers, pp. 96-97(2007)

[6] J. Dorsey, S. Searles, M. Ciraula, et al.: “An Integrated Quad-CoreTM Opteron Processor”, ISSCC Dig. Tech. Papers, pp. 102-103(2007).

[7] U. Nawathe, M. Hassan, L. Warriner, et al.: “An 8-Core 64-Thread 65b Power-Efficient SPARC SoC”, ISSCC Dig. Tech. Papers, pp. 108-109 (2007).

[8] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, M. A. Mesa, and A. Ramirez.: Parallel Scalability of Video Decoders. Delft University of Technology(2008).

[9] Aho, A.V., Sethi, R., & Ullman, J.D. Compilers: principles, techniques, and tools. Boston: Addison-Wesley Longman(2007).

[10] K.-H. Chen: “48 Cycles-per-macro block deblocking filter accelerator for high-resolution H.264/AVC decoding”, in IET Circuits, Devices & Systems. 2010.

[11] S.-W. Wang, S.-S. Yang, H.-M. Chen, C.-L. Yang, and J.-L. Wu,“A multicore architecture based parallel framework for H.264/AVC deblocking filters,” J. Signal Process. Syst., vol. 57, no. 2, pp. 195–211, Nov. 2009.

[12] GSAIM, Chung-Ang University, Seoul, "Variable block-based deblocking filter for H.264/AVC on low-end and low-bit rates terminals ", Signal Processing: Image Communication, vol. 25, pp. 255-267, 2010.

[13] M. Taylor, J. Kim, J. Miller, et al.:“A 16-Issue Multiple-Program-Counter Microprocessor with Point-to-Point Scalar Operand Network”,ISSCC Dig. Tech. Papers, pp. 170-171(2003)

[14] .S. Vangal, et al.: “An 80-tile 1.28TFLOPS Network-on-Chip in 65nm CMOS”, ISSCC Dig. Tech. Papers, pp. 98(2007)

[15] A. Agarwal, L. Bao, J. Brown, et al.: “Tile Processor: Embedded Multicore for Networking and Digital Multimedia”, Hot Chips(2007)

[16] S. Bell, B. Edwards, J. Amann, et al.:TILE64-Processor: A 64-Core SoC with Mesh. Interconnect Solid-State Circuits Conference(2008)

[17] H.Chen, R.Hu and Y.Gao, "An effective method of deblocking filter for H.264/AVC", Communications and Information Technologies, The Conf Oct 17-0ct 19, pp.l092-1095, Oct. 19 (2007).