IEICE TRANS. INF. & SYST., VOL.E91–D, NO.12 DECEMBER 2008

LETTER

Cache Optimization for H.264/AVC Motion Compensation*

Sangyong YOON[†a)], ***Student Member and*** **Soo-Ik CHAE**[†], ***Nonmember***

SUMMARY In this letter, we propose a cache organization that substantially reduces the memory bandwidth of motion compensation (MC) in the H.264/AVC decoders. To reduce duplicated memory accesses to P and B pictures, we employ a four-way set-associative cache in which its index bits are composed of horizontal and vertical address bits of the frame buffer and each line stores an 8×2 pixel data in the reference frames. Moreover, we alleviate the data fragmentation problem by selecting its line size that equals the minimum access size of the DDR SDRAM. The bandwidth of the optimized cache averaged over five QCIF IBBP image sequences requires only 129% of the essential bandwidth of an H.264/AVC MC.

key words: *cache, H.264, motion compensation, memory bandwidth, DDR SDRAM*

1. Introduction

H.264/AVC is currently the most popular video standard because it provides good video quality at a substantially lower bit rate than other video standards. However it requires more computation and higher memory bandwidth from an external SDRAM. Its memory bandwidth requirement becomes a performance bottleneck, especially in the high-definition video decoder systems [5].

Motion compensation accounts for about 75% of the total external memory bandwidth in the H.264/AVC decoders [1]. Therefore, reducing the memory bandwidth of MC is very important in designing H.264/AVC decoders for high-performance video and mobile applications [5], [6]. MC bandwidth reduction methods used in the previous works can be summarized into two types of data fetches: 1) motion-vector precision aware ones which load less data for integer motion vectors [3], [4]; 2) block-size aware ones which load less data for MC blocks larger than 4×4 [3]–[6].

A system-on-a-chip requires higher memory bandwidth because multiple multimedia applications share an external SDRAM. Therefore, adopting a double data rate (DDR) SDRAM becomes inevitable to achieve higher memory bandwidth especially for high-definition video decoders. However, the DDR SDRAMs have a larger minimum access size than the single data rate (SDR) SDRAMs, which incurs a data fragmentation problem that part of data loaded from the SDRAM is not utilized especially in non-sequential access patterns from MC. To further reduce this bandwidth loss, it is necessary to keep loaded data for later use even if they are not used immediately.

In designing an H.264/AVC video decoder, we decided to use MC with a cache because using a cache is generally effective in reducing memory bandwidth for duplicated data accesses. Moreover, we selected a line size of the cache that equals the minimum access size of the DDR SDRAM to alleviate this data fragmentation problem. Direct-mapped caches for the H.264/AVC MC were proposed using a split-index mapping [1] and a tile-based mapping [2]. However, these direct-mapped caches show degraded performance especially in B pictures. Therefore, we also tuned cache organization based on the memory access patterns of the MC to maximize the data reuse by increasing the data locality in the cache. To find an optimized cache for H.264/AVC motion compensation, we first examine its memory access patterns including P and B pictures and then explore the cache design space by configuring cache size, set-associativity, cache index mapping, and cache offset mapping.

This letter is organized as follows. We explain the motivation of using a cache for H.264 MC in Sect. 2. We describe typical memory access patterns of H.264 MC and a cache organization suitable to memory bandwidth reduction in Sect. 3. We show experimental results for MC bandwidth reduction in Sect. 4 and conclude this letter in Sect. 5.

2. Data Reuse in H.264 Motion Compensation

In this letter, we assume that all MC engines employ a motion-vector precision aware method [3], [4] and focus on finding an effective data reuse method to reduce memory bandwidth loss for H.264/AVC MC. Note that the motion-vector precision aware method is orthogonal to a method using a cache for data reuse or block-size aware methods [3]–[6]. Memory data accesses of the H.264/AVC MC can be divided into three types: 1) essential data accesses; 2) duplicated data accesses loading already-loaded data again; 3) redundant data accesses loading data that are not utilized immediately. Bandwidth used by duplicated and redundant accesses can be minimized by effectively reusing previously loaded data.

Although the H.264 standard employs quarter-pel inter prediction for several different sized blocks such as 4×4, 4×8, 8×4, and 16×16, hardware implementation of the MC is based on a 4×4 block to simplify its control logic

Manuscript received May 9, 2008.
Manuscript revised July 31, 2008.

[†]The authors are with the School of Electrical Engineering and Computer Sciences, Seoul National University, 151–742 Korea.

[*]This work was supported by "System IC 2010" project of Korea Ministry of Knowledge Economy and the Brain Korea 21 project.

a) E-mail: syyoon@sdgroup.snu.ac.kr

DOI: 10.1093/ietisy/e91–d.12.2902

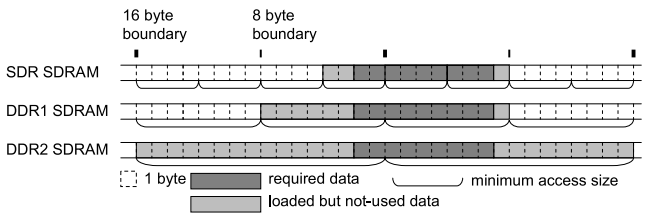

Fig. 1 Data Fragmentation due to minimum access size in SDRAM.

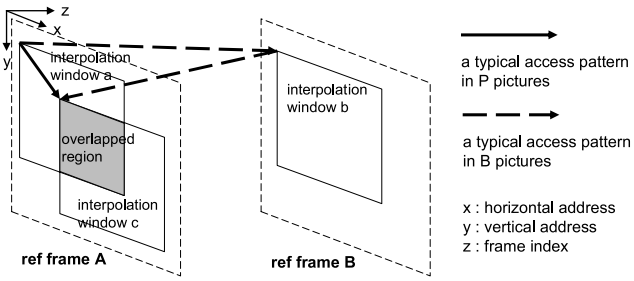

Fig. 2 Typical MC memory access patterns in P and B pictures.

and to reduce its hardware complexity [3]. However, this approach introduces severe bandwidth overhead because interpolation operations of MC for each 4×4 block use 6-tab filtering, which requires a 9×9 interpolation window. In the block-size aware method, overlapped pixels between the 9×9 interpolation windows of two adjacent 4×4 blocks are reused by storing them in a register file for an MC block larger than 4×4. For example, the block-size aware method loads only 13×13 bytes instead of 4 times 9×9 bytes if the MC block is 8×8. However, it cannot reuse the data between two 4×4 MC blocks having different motion vectors even though they have substantially overlapped region between their interpolation windows. Therefore, we employed a cache for motion compensation to substantially reduce the bandwidth loss incurred by the duplicated memory accesses because the cache can provide more general data reuse.

The DDR SDRAM achieves high-speed external transfers by using an internal prefetch architecture, which determines the minimum access size in the external bus. The minimum access sizes of SDR, DDR1 and DDR2 SDRAM are 32, 64 and 128 bits, respectively, for 32-bit data width. If the memory accesses are not sequential, the minimum access size causes a data fragmentation problem that part of data loaded from SDRAM is not used, as shown in Fig. 1. Moreover, this bandwidth loss due to the fragmentation increases for the DDR SDRAMs, which have a larger minimum access size. To reduce the overhead of the redundant accesses, we employed a cache with line size equal to the minimum access size of the DDR SDRAMs to further reuse data even if they are not used immediately.

3. Cache Organization Optimized for MC Memory Access Patterns

To minimize the miss rate of a cache, its organization must properly be tuned to the memory access patterns of MC. We explain memory access patterns of H.264/AVC MC and the suitable cache organization in this section.

To predict a 4×4 block with a quarter-pel precision motion vector, forward prediction in P pictures loads a 9×9 interpolation window while bidirectional prediction in B pictures loads two 9×9 interpolation windows from two different reference frames. MC memory access patterns for a 4×4 block in B pictures are quite different from those in P pictures, as shown in Fig. 2. Therefore, the cache organization supporting both P and B pictures for H.264/AVC MC should be able to store two-dimensional (2D) overlapped region of interpolation windows without conflict and reduce cache conflicts during interleaved access to different reference frames.

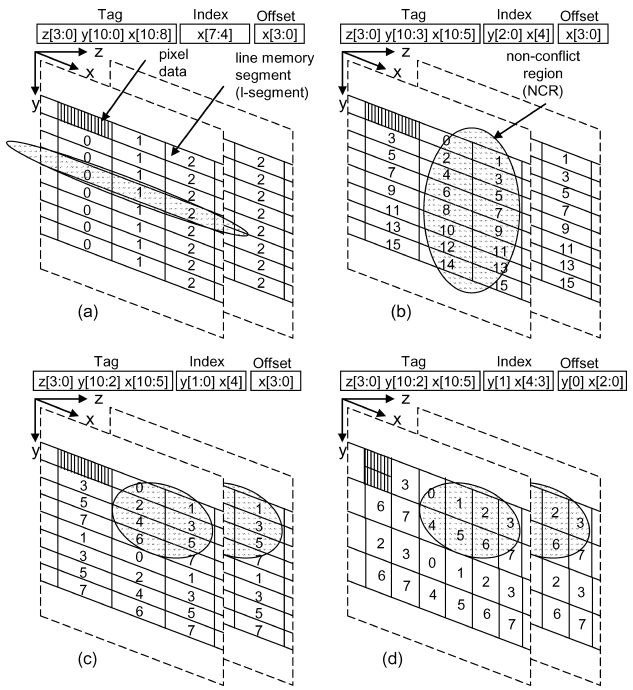

Fig. 3 Non-conflict regions for (a) direct-mapped cache with row-major mapping, (b) direct-mapped cache with block-index mapping, (c) 2-way set-associate cache with block-index mapping, (d) 2-way set-associative cache with block-index and block-offset mapping.

Here, we introduce two concepts: a line memory segment (l-segment) and a non-conflict region (NCR) to explain a cache organization that satisfies these constraints. An l-segment is defined as a collection of memory locations that can be loaded together into a cache line and obviously its size is equal to the cache line size. An NCR is defined as a collection of l-segments that can be loaded together into the cache without conflicts. The shape of an NCR in the three-dimensional (3D) memory space of the reference frame buffer depends on cache parameters like set-associativity, index mapping, and offset mapping, which is shown in Fig. 3.

In a direct-mapped cache with row-major mapping, as shown in Fig. 3 (a), an NCR is a horizontal grouping of l-segments, each of which is loaded into a different cache line.

An NCR for a direct-mapped cache with block-index mapping is shown in Fig. 3 (b) where the cache index is composed of both horizontal and vertical addresses of the frame buffer. Because the NCR is a 2D array of l-segments, cache conflict misses are minimized and data reuse is maximized when overlapped 2D interpolation windows for MC need to be loaded into the cache. In a two-way set-associative cache with block-index mapping, an NCR is a 3D array of l-segments and two l-segments that are not located in the same frame are mapped into the same set with the same cache index as shown in Fig. 3 (c), which can substantially reduce cache conflict misses when two different frames are alternately accessed for motion compensation in B pictures. Because MC memory access patterns have 2D locality, we can improve this set-associative cache further by using l-segments that are a 2D arrays of pixels as shown in Fig. 3 (d), where cache offset bits are composed of both horizontal and vertical addresses of the frame buffer, which we call block-offset mapping. While an l-segment is a 16×1 array of pixels in Figs. 3 (a), (b), (c), it is an 8×2 array of pixels in Fig. 3 (d).

4. Experimental Results

By using an evaluation flow shown in Fig. 4, we evaluated the performance of caches for main-profile bit streams of five image sequences of QCIF (Forman, Carphone, Claire, Coastguard, and Salesman) coded at 30 frames/s by a JM 8.6 encoder [7] in which its rate control was disabled. First, we collected traces of motion vectors and reference indices by decoding the bit streams with a JM 8.6 decoder [7]. Then, we generated cache access traces with a model for an H.264 MC engine using a motion-vector precision aware method [3], [4]. In the next step, for each cache configuration we generated memory access traces from the cache access traces by using its cache model and repeated this step by changing cache size, cache index mapping, set-associativity, and cache offset mapping. In this experiment, the cache line size was fixed to 16 bytes, which is equal to the minimum access size of the DDR2 SDRAMs. Finally, we evaluated the memory access traces with a bandwidth analyzer for various types of SDRAMs such as SDR, DDR1, and DDR2 memories, each of which has a different minimum access size. In this letter, we focus only on the memory bandwidth for luma MC.

As shown in Fig. 5 (a), we compared the average memory bandwidth of direct-mapped caches for two cache index mappings: a row-major one and a block-index one. The latter mapping reduces the memory bandwidth substantially in P pictures by reducing the cache conflicts in loading a 2D region for motion compensation because a 2D region in the 3D memory address space of the reference frame buffer is an NCR for a direct-mapped cache with block-index mapping. In B pictures, however, the bandwidth reduction is relatively lower due to the cache conflicts from interleaved accesses to different frames. This type of cache conflicts can substantially be reduced especially in B pictures by using set-associative caches because a 3D cube is an NCR for a set-associative cache. As shown in Fig. 5 (b), we compared the bandwidth of the caches with block-index mapping by changing the set-associativity. Finally, we compared the bandwidth of four-way set-associative caches with block-index mapping for several offset mappings, as shown in Fig. 5 (c). Cache offset mapping changes the shape of each l-segment, which is a rectangle for block-offset mapping. About 10% improvement can be obtained in both P and B pictures by block-offset mapping. Based on the overall experimental results, we propose to employ a 1-Kbytes four-way set-associative cache adopting block-index mapping with 8×2 cache line for H.264 luma MC, which is optimal in reducing the memory bandwidth. Although the cache performance for 4×4 cache lines is slightly better than that for 8×2 cache lines, as shown in Fig. 5 (c), we selected 8×2

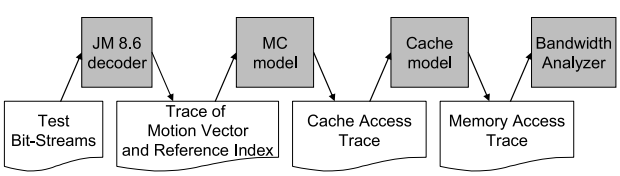

Fig. 4 Cache performance evaluation flow.

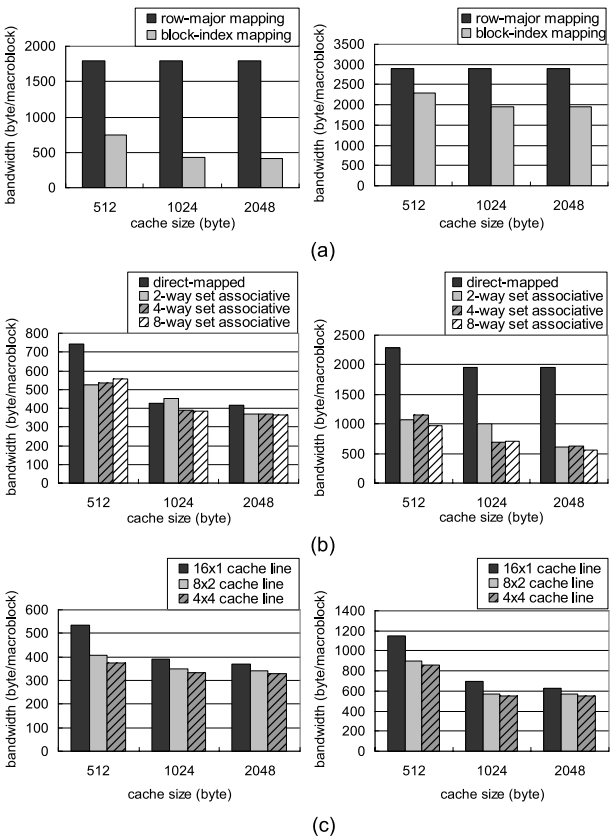

Fig. 5 Bandwidth evaluation for the luma MCs with a cache by configuring three cache parameters: (a) cache index mapping, (b) set-associativity, and (c) cache offset mapping. Left and right graphs show the average bandwidth of P and B pictures, respectively, for the five bit-streams.

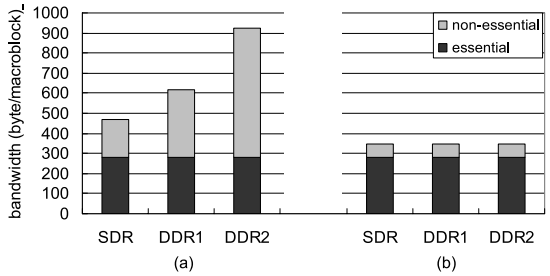

Fig. 6 Average memory bandwidth of the MC (a) using the block-size aware method and (b) with the proposed cache for several types of the SDRAMs in a sequence of IPP (N=15, without B picture).

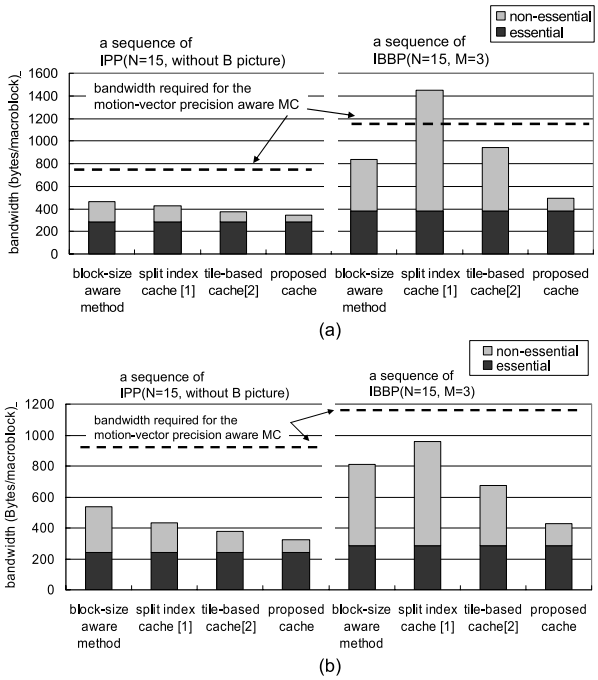

Fig. 7 MC memory bandwidth comparisons of several bandwidth reduction schemes averaged for (a) 5 QCIF image sequences and (b) 3 SD1 video clips.

cache lines simply because the line buffer size in the LCD controller based on a raster scanning pattern depends on the height of l-segments which determines memory mapping in the SDRAMs.

Block-size aware MC engines suffer from a significant amount of bandwidth loss from data fragmentation in the DDR SDRAMs, as shown in Fig. 6 (a). By introducing a cache with a line size that equals to the minimum access size of the DDR2 SDRAMs, we reduce their bandwidth losses substantially, which are equal to those for SDR SDRAM, as shown in Fig. 6 (b). The memory bandwidths of three different cache organizations and a block-size aware MC engine are compared in Fig. 7 for two differently encoded versions of the bit streams: one for IPP (N=15, without B picture) and the other for IBBP (N=15, M=3). Figure 7 (a) shows averaged results of 5 QCIF (176×144) image sequences mentioned earlier in this section, and Fig. 7 (b) illustrates those of three SD1 (720×480) 30-frames video clips sampled from The Rock, War of the Worlds, and Underworld Evolution. By using the proposed cache, we could substantially reduce the non-essential memory bandwidth from SDRAM. Compared to the motion-vector precision aware H.264 MC, the non-essential bandwidth is reduced to 18% and 16% for the QCIF and SD1 IBBP sequences, respectively. The split index cache [1] is a direct-mapped cache with block-index mapping and the tile-based cache [2] is a direct-mapped cache adopting block-index and block-offset mapping, where their cache size and line size were also set to 1 Kbytes and 16 bytes, respectively, for fair comparison. Note that the split index cache requires substantially higher memory bandwidth for the IBBP sequence because of increased conflicts due to cache thrashing.

5. Conclusions

To reduce the bandwidth loss in the DDR SDRAM, we propose a four-way set-associative cache for H.264 MC in which its index bits are composed of horizontal and vertical address bits of the frame buffer and each line stores an 8×2 pixel data in the reference frames. The experimental results show that the proposed cache is effective for reducing the required memory accesses for P and B pictures while reducing data fragmentation loss. The non-essential memory bandwidth of the proposed cache from SDRAM is substantially reduced, compared to that of the motion-vector precision aware H.264 MC.

References

[1] J.-H. Kim, G.-H. Hyun, and H.-J. Lee, "Cache organization for H.264/AVC motion compensation," Proc. IEEE Int. Conf. Embedded and Real-Time Computing Systems and Applications, pp.534–541, Aug. 2007.

[2] Y. Li, Y. Qu, and Y. He, "Memory cache based motion compensation architecture for HDTV H.264/AVC decoder," Proc. IEEE Int. Symposium on Circuit and Systems, pp.2906–2909, May 2007.

[3] C.-Y. Tsai, et al., "Bandwidth optimized motion compensation hardware design for H.264/AVC HDTV decoder," 48th Midwest Symposium on Circuit and Systems, vol.2, pp.1199–1202, Aug. 2005.

[4] R.G. Wang, J.T. Li, and C. Huang, "Motion compensation memory access optimization strategies for H.264/AVC decoder," Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, vol.5, pp.97–100, March 2005.

[5] C.C. Lin, et al., "A 160K gates/4.5 KB SRAM H.264 video decoder for HDTV applications," IEEE J. Solid-State Circuits, vol.42, no.1, pp.170–182, Jan. 2007.

[6] T.M. Liu, et al., "A 125 uW, fully scalable MPEG-2 and H.264/AVC video decoder for mobile applications," IEEE J. Solid-State Circuits, vol.42, no.1, pp.161–169, Jan. 2007.

[7] H.264/AVC JM Reference Software, version JM 8.6, http://iphome.hhi.de/suehring/tml/download/old_jm/