AN EFFICIENT REFERENCE FRAME STORAGE SCHEME FOR H.264 HDTV DECODER

Peng Zhang, Wen Gao, Di Wu,

Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
{zhangpeng, wgao, dwu}@jdl.ac.cn

Don Xie

Grandview Semiconductor
Beijing, China
don.xie@grandviewsemi.com

ABSTRACT

This paper proposes an efficient reference frame storage scheme for HDTV VLSI decoder to reduce external memory bandwidth requirement. The proposed scheme consists of the pixel duplication mechanism and the L-C (luma-chroma) correlated mapping method. Pixel duplication completely eliminates the possibility of an access crossing word boundary and therefore substantially increases the memory bandwidth efficiency. L-C correlated mapping exploits address relationships between the luma and chroma reference pixels and largely reduces bank conflict overhead of memory accesses. The two mechanisms combined together efficiently improve the bandwidth usage: up to 47% bandwidth in worst case is saved compared with the previous schemes, and 25% in average case.

1. INTRODUCTION

ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding (commonly referred as H.264/AVC) [1] is the newest entry in the series of international video coding standards, developed jointly by ITU-T VCEG and ISO/IEC MPEG. For high coding performance, H.264 recommends a series of efficient coding tools targeting at a variety of applications from broadcasting to mobile communication, etc. Comparing with previous video standards, such as MPEG-2 and MPEG-4, H.264 achieves high performance gain by introducing smaller block size and flexible prediction modes, etc. But it is accompanied by high computation complexity. H.264 decoder, especially for HDTV applications, contains very large amount of data processing, which requires large amount of data transfer between storage units and processing units. The throughput of processing units can be improved by advanced ASIC process and design methodology exploiting parallelism. But because of the limitation of physical design, implementations and process factors, DRAM performance is not improved as fast as CMOS logic circuit [2]. The gap between them causes memory bandwidth a bottleneck in overall system performance. Efficient memory controller design is a key technique in video decoder designs [3-8].

Memory controller design has been broadly studied in computer architecture [2] for a long time, and was discussed in video decoder design from around the mid-1990's. Bus scheduling schemes were evaluated on precise simulation models by Nam Ling [3], etc. With the prevalence of modern advanced SDRAM, more attention was paid on utilizing the 3-D feature of prevalent (commercial) SDRAMs. Methods have been worked out to avoid bank conflict, the major overhead of SDRAM. Macroblock-based mapping [4][5][7] can substantially reduce bank conflict overhead within one macroblock(MB); memory access scheduling [5][6] reorders memory accesses to insert non-conflicted accesses between conflicted ones, and memory mode prediction [8] reduces conflict overhead by adaptively using auto-precharge. But H.264 requires much more memory bandwidth than MPEG2, because of long tap interpolation filter, dispersive reference blocks and additional mass data that need to store off-chip.

Worst case guarantee is the basic requirement in real time system design. In this paper, we find that previous memory mapping schemes can not meet the worst case requirement of memory bandwidth for HD-1080i and above. Based on worst case consideration, we propose a pixel duplication mechanism to effectively avoid word boundary crossing during reference frame accesses. And we also propose a correlated luma-chroma mapping scheme to effectively minimize bank conflicts between luma and chroma reference pixel access. A substantial reduction of memory bandwidth is observed by using the two approaches together. This paper is organized as follow. Memory access is analyzed in section II. We propose the novel memory mapping scheme, and address translating formulas in section III. Section IV lists experimental results and implementation issues and followed by a brief summary in section V.

2. MEMEROY ACCESS ANALYSIS

2.1. Memory Access Patterns

This work is supported by the National High Technology Research and Development Program of China (863 No. 2003AA1Z1290), and Grandview Semiconductor Inc.

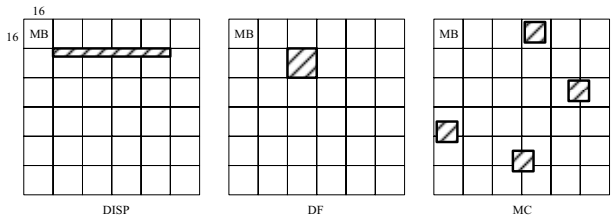

Figure 1 Memory access patterns of DISP, DF and MC

There are five memory clients in typical H.264 video decoder, and their access patterns vary a lot. Input Buffer stores input coded stream into memory, and VLD reads them from memory for syntax parsing. Because of the high compression ratio (10x ~ 100x) of H.264, their memory bandwidth consumption is relatively small. De-blocking Filter (DF) stores decoded MBs into memory for display and reference, and DF accesses are MB aligned two dimensional block accesses. And DF overhead can be minimized by mapping pixels of each whole MB into the same page. Display (DISP) feeds decoded pictures to the display device, and it accesses memory in one dimensional raster manner. And its overhead can be reduced by mapping more horizontal MBs into the same page, and horizontal neighboring pages into different banks. Though motion compensation (MC) accesses are also two dimensional block accesses, they have variable sizes and are not aligned with the MB boundaries, because motion vectors are independent with the boundaries.

2.2 H.264 MC Reference Block

H.264 adopts a 6-tap FIR filter for half-pixel interpolation, and a 2-tap filter for quarter-pixel interpolation. Chroma fractional pixels are interpolated by bilinear filtering.

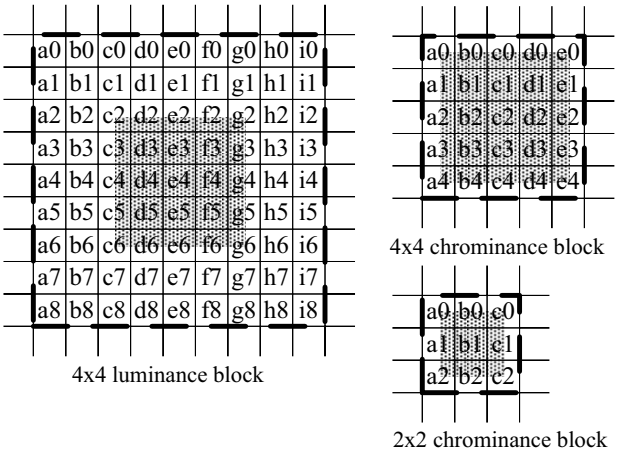

Figure 2 MC reference block

Fig. 2 shows the reference integer pixels required for fractional pixel interpolation in the shadowed blocks in the worst case. To obtain the fractional pixel between c2 and d2, a2~f2 are required; for the pixel between f2 and g2, d2~i2 are required; and it is similar to the vertical direction. Finally, 9x9 reference block a0~i8 are required for one 4x4 fractional luma block. Similarly, an 8x8 luma block requires 13x13 integer reference pixels. For chroma, the reference pixel block is one pixel larger than the referrer block size in both directions.

2.3 Memory Bandwidth Requirements

SDRAM architecture is introduced in [4][7][8]. Its bandwidth consumption consists of CAS (Column Access Strobe) cycles and overhead]. CAS cycles are mainly determined by decoding procedure of H.264, and overhead is mainly generated by bank conflicts. Bank conflicts occur when successive accesses address different rows of the same bank. To read one MC reference block that crosses four banks, the overhead is at most 12 cycles for the typical SDRAMs to open all the conflicted four banks.

Table 1 Cycles consumed by MC per MB

Pattern	CAS	OH	MC	Pattern	CAS	OH	MC
One 16x16 Block	162	48	210	Eight 8x4 block	384	384	768
Two 16x8 Block	196	96	292	Eight 4x8 block	576	384	960
Two 8x16 Block	240	96	336	Sixteen 4x4 block	768	768	1536
Four 8x8 Block	288	192	480				

Table 1 assumes SDRAM has 128-bit word and four banks, and pixel depth is 8-bit. It lists the cycles consumed by MC per MB. For 8x8 blocks, MC CAS cycles are 13x2(cross-word)x2(bi-direction)x4+10x2x4(chroma)=288, and MC overhead is 12x2x4+12x2x4=192. Summing up DF and DISP, the total cycle count is 560. Required SDRAM frequency for HD1080i (30fps) is 1920x1088x30x560/256= 131(MHz). For 4x4 blocks, required frequency is 377MHz. As a DDR memory system, data rate is as high as 754MHz. It is very expensive for the external memory chip, and also very challenging in memory interface design. Large amount of cycles are consumed by word crossing and chroma overhead. Our scheme eliminates the two cases, and reduces the bandwidth cost by half, so a typical 200MHz DDR memory is competent for the task.

3. PROPOSED STORAGE SCHEME

3.1 Pixel Duplication

MC is the largest bandwidth consumer because its requests are irregular and dispersive. A typical reference line with 9 pixels (72 bits) may require two 128-bit words, when the 9 bytes cross the word boundary in memory as shown in Fig. 1 (for MC). Two cycles are consumed to read one line, which reduces bandwidth efficiency by half. In addition, it is also possible that the two words involved located in different banks (or pages), more cycles will be needed to activate the additional bank or page.

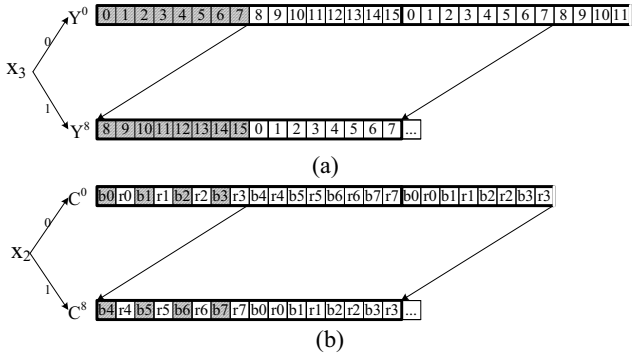

Figure 3 Redundant word organization

In our mapping scheme, reference pixels are stored with duplication such that one pixel will appear in two different words. Memory controller can address the proper word that contains the whole reference line so as to avoid word boundary crossing. As shown in Fig. 3(a), each MB line has two different copies associated with it, corresponding to left-shifting the original word for the MB line by 8 pixels and padding the LSB with the pixels in the raster scan order. The right-most 8 pixels of current word and left-most 8 pixels of the next word are concatenated to form a new word stored in Y^8 MB. Thus, a line of 9 consecutive pixels (bytes) will be fully contained in Y0 or Y^8 word. Which word to pick is determined by the third bit of the byte address within a 16-byte word (x3). The last three bits x2x1x0 are used to indicate the starting pixel position within the corresponding word. It can be observed that a reference line less than 10 bytes consumes only one single cycle, in spite of theirs start positions. 13-pixel lines for 8x8 reference block can be split into two word accesses. In case of chroma, Cr and Cb pixels are interleaved as shown in Fig3. b. C^0 and C^8 (left-shifting by 8 pixel positions) can fully contain all cases of start position of 5x5 blocks within one 128-bit (or 16 pixel) word.

3.2 Luma-Chroma Correlated Mapping

In most video coding standards, Luma and chroma blocks share the same motion vector for motion compensation. This relationship can be utilized to eliminate bank conflict between luma and chroma accesses. The additional overhead is saved for the chroma assuming bank activation can be hidden into the luma CAS cycles. But there is an offset between motion vector and reference pixels start points. Let's assume the chroma format is 4:2:0, the integer part of motion vector is (x, y) for luma, and (x/2, y/2) for chroma; and predicted block size is MxN. Reference block for luma is the rectangle ranging from (x-2, y-2) to (x+M+2, y+N+2). For chroma blocks, the rectangle ranges from (x/2, y/2) to (x/2+M/2, y/2+N/2). If we enlarge the chroma coordinates by a factor of two, though the two rectangles are not aligned at start positions, the enlarged chroma one can be totally contained in the luma one. So only if we map the correlated luma pixel and its chroma corresponding into different banks, and access luma block and chroma block contiguously, bank conflict between luma block and chroma block can be avoid.

3.3 Combined Mapping Scheme

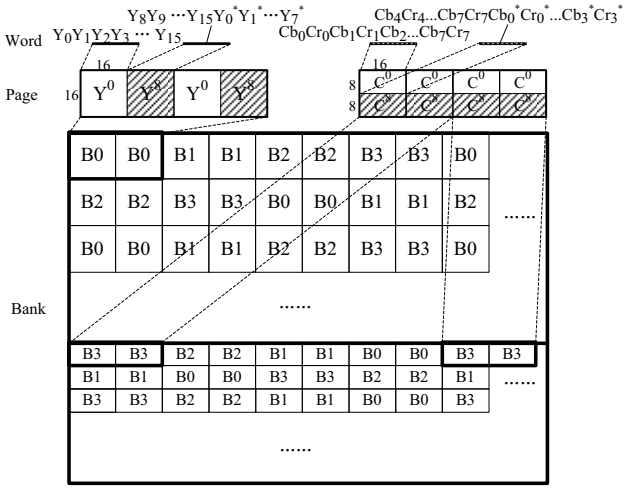

Figure 4 Address mapping scheme

Detailed mapping scheme is shown in Fig. 4. Luma pixels and chroma pixels are stored in two segments of memory. In the figure, one white block stands for one MB, and luma MB is two times larger in vertical direction than the interleaved chroma one. Y^0 and Y^8 of the same MB are mapped into the same page; neighboring MBs which is mapped into the same bank constitute one page; neighboring pages are mapped into different banks. A typical page size is 1024 bytes, one page can just contain two neighboring luma MBs or four neighboring chroma MBs, which are mapped into the same bank. For example in Fig. 4, luma MB0 and MB1 are mapped into bank 0 row 0, which is indicated by thick rim around the first two MBs marked by B0. MB2 and MB3 are mapped into bank 1 row 0. And MB 4 also can be mapped into bank0 to ensure different bank for neighboring pages, but we allocate MB4 into bank3 to increase burst length in DISP accesses. For chroma, MB0 and MB1 only need half of one page, so MB8 and MB9 are stored in the same page as MB0 and MB1.

As we adopt duplicated mapping, word crossings are avoided, and meanwhile bank crossings in horizontal direction are also avoided. At most two banks are accessed for one luma reference block, so we allocate the remaining two banks to corresponding chroma block. If luma MB is mapped into bank N, the corresponding chroma MB is mapped into bank 3-N, or in another form, bit reversion of N.

To describe the mapping scheme clearly, equations are summarized here. In the formulas, {,} is a bit-concatenation

operator, $\oplus$ is a bit-XOR operator, ! is bit-inverse operator, $1'b0$ is one-bit string of zero, and [,] is a bit-selection operator among a bus. $x[10:0], y[10:0]$ is the pixel location of the reference pixels for luma. And $x[9:0], y[9:0]$ is for chroma Cb.

For luma:

$BankAddr = \{x[6] \oplus y[4], x[5]\}$

$RowOffset = \{y[10:4], x[10:7]\}$

$ColumnAddr = \{x[4,3], y[3:0]\}$

$ByteAddr = \{1'b0, x[2:0]\}$

And for chroma:

$BankAddr = \{!(x[5] \oplus y[3]), !x[4]\}$

$RowOffset = \{y[9:3], x[9:7]\}$

$ColumnAddr = \{x[6], x[2], y[2:0]\}$

$ByteAddr = \{1'b0, x[1:0], 1'b0\}$

4. EXPERIMENTS AND IMPLEMENTATION

The worst case of memory bandwidth consumption occurs in B MB, and each block has bi-directional fractional motion vectors, and reference block has four bank conflicts.

Table 2 Worst cases analysis

Schemes / block size	MC CAS	MC OH	Freq (MHz)	Schemes / block size	MC CAS	MC OH	Freq (MHz)
(1) 4x4	768	768	377	(1) 8x8	288	192	131
(2) 4x4	384	768	304	(2) 8x8	144	192	114
(3) 4x4	384	384	211	(3) 8x8	144	96	88
(4) 4x4	384	384	200	(4) 8x8	288	192	136

Four schemes are compared: (1) for the previous schemes[3][4], (2) for the scheme with only pixel duplication, and (3) is reference scheme which duplicates the pixel by four times and each copy offset four pixels, and (4) is proposed scheme. Required SDRAM frequencies for real time decoding are list in Tab. 2. For the duplication, DF needs to store more data into SDRAM, but the additional CAS cycles are well offset by the savings from the bank-crossing of the MC reads. Proposed scheme can reduce up to 47% bandwidth requirement for 4x4 blocks, while reference scheme (3) performs well for 8x8 blocks. Fig. 5 shows the simulation results on a variety of test streams. 25% bandwidth requirement is saved by proposed scheme. Integer motion vector and especially zero motion cases will not benefit from our proposed scheme, but meanwhile these cases relax the system real time requirement as well.

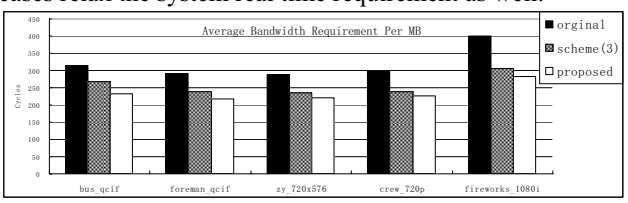

Figure 5 Average bandwidth consumption

From the formulas in above section, we can see that the mapping scheme is straight forward and the address translation complexity is extremely low.

In our scheme, to improve the bandwidth utilization, two times of memory capacity are needed. We investigate a typical case for five pictures reference MC in H.264, and six decoded pictures are stored in memory. The memory size required for the proposed scheme is about (1920x1088x6)x2 +(1920x544x6)x2=38MB. We only consume an addition memory of 19MB, but decrease the required memory interface frequency from 754MHz to 400MHz. Off-chip memory density is relatively cheap compared with bandwidth. And design cost on high frequency memory interface is also reduced much. Data compression can also be adopted in the decode picture storage to further reduce the density demand.

5. CONCLUSION

In this paper, we proposed an efficient memory mapping scheme for H.264 VLSI decoder, which can save up to 47% of the total external memory bandwidth consumption compared with the previous schemes. Pixel duplication can effectively reduce total access time by avoiding accesses crossing word boundary. L-C correlated mapping can save the access overhead for the chroma pixels in one MB. Significant memory bandwidth efficiency is achieved with a relatively small additional off-chip memory density demand.

6. REFERENCES

[1] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", July 2004.

[2] J. L. Hennessy and D. A. Partterson, *Computer Architecture: A Quantitative Approach, 3rd ed.* San Francisco, CA: Morgan Kaufmann, 2002.

[3] J. H. Li, N. Ling, "Architecture and Bus-Arbitration Schemes for MPEG-2 Video Decoder", *IEEE trans. on CSVT*, vol. 9, no. 5, Aug. 1999.

[4] T. Takizawa, J. Tajime, H. Harasaki, "High performance and cost effective memory architecture for an HDTV decoder LSI", *Proc. of ICASSP 1999*, vol. 4, pp. 1981-1984, Mar. 1999.

[5] T. Takizawa, M. Hirasawa, "An Efficient Memory Arbitration Algorithm for A Single Chip MPEG-2 AV Decoder", *IEEE trans. on Consumer Electronics*, vol. 47, no. 3, Aug. 2001

[6] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, J. D. Owens, "Memory Access Scheduling", *Proc. of ISCA 2000*, pp. 128-138, Jun. 2000.

[7] H. Kim, I. C. Park, "High-Performance and Low-Power Memory-Interface Architecture for Video Processing Applications", *IEEE trans. on CSVT*, vol. 11, no. 11, Nov. 2001.

[8] S. Park, Y. Yi, I. C. Park, "High Performance Memory Mode Control for HDTV Decoders", *IEEE trans. on Consumer Electronics*, vol. 49, no. 4, Nov. 2003.