ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.12, 1–11

Memory interface design for AVS HD video encoder with Level C+ coding order

Xiaofeng Huang[1a)], **Kaijin Wei**[2], **Guoqing Xiang**[2], **Huizhu Jia**[2], **and Don Xie**[2]

[1] *NVIDIA Co.,*

Qiuyue Road, Pudong New Area, Shanghai, China

[2] *Peking University,*

No. 5 Yiheyuan Road, Haidian District, Beijing, P. R. China

a) *hehuang@nvida.com*

Abstract: In video encoder chip, memory interface design is a must to transfer various data between the encoder pipeline and the off-chip memory. Reducing the required off-chip memory bandwidth and improving the memory access efficiency are the two key targets for optimized memory interface design. To achieve these two targets, three novel technologies are proposed in Level C+ coding order based AVS HD video encoder. Firstly, an improved Level C+ coding order with necessary NOP insertions are proposed to achieve 61% bandwidth reduction and make MB pipeline scheduling regular. Secondly, MB-level synchronous memory interface design is proposed by trading off external bandwidth, MB pipeline structure, and internal buffer size. Finally, address mapping and arbitration are proposed to improve the access efficiency by 12%. The optimized memory interface design is successfully implemented on our 1080P@45fps AVS encoder with Xilinx Virtex-6 FPGA at an operating frequency of 200 MHz.

Keywords: memory interface, bandwidth, coding order, scheduling, address mapping

Classification: Integrated circuits

References

[1] J. V. Team: Draft ITU-T Rec. and Final Draft Int. Standard of Joint Video Spec. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC (2003).

[2] Information technology - Advanced coding of audio and video - Part 2: Video. AVS Standard Draft (2005).

[3] S. H. Lee, *et al.*: "Lossless frame memory recompression for video codec preserving random accessibility of coding unit," IEEE Trans. Consum. Electron. **55** (2009) 2105 (DOI: 10.1109/TCE.2009.5373775).

[4] A. D. Gupte, *et al.*: "Memory bandwidth and power reduction using lossy reference frame compression in video encoding," IEEE Trans. Circuits Syst. Video Technol. **21** (2011) 225 (DOI: 10.1109/TCSVT.2011.2105599).

[5] H. Kim and I.-C. Park: "High-performance and low-power memory interface architecture for video processing applications," IEEE Trans. Circuits Syst. Video Technol. **11** (2001) 1160 (DOI: 10.1109/76.964782).

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

[6] C.-Y. Chen, *et al.*: "Level C+ data reuse scheme for motion estimation with corresponding coding orders," IEEE Trans. Circuits Syst. Video Technol. **16** (2006) 553 (DOI: 10.1109/TCSVT.2006.871388).
[7] J. C. Tuan, *et al.*: "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Trans. Circuits Syst. Video Technol. **12** (2002) 61 (DOI: 10.1109/76.981846).
[8] K. Wei, *et al.*: "An optimized hardware video encoder for AVS with Level C+ data reuse scheme for motion estimation," 2012 IEEE Inter. Conf. on Multimedia and Expo (ICME) (2012) 1055 (DOI: 10.1109/ICME.2012.11).
[9] H. B. Yin, *et al.*: "A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders," IEEE Trans. Circuits Syst. Video Technol. **20** (2010) 1242 (DOI: 10.1109/TCSVT.2010.2058476).
[10] X. F. Huang, *et al.*: "A highly efficient external memory interface architecture for AVS HD video encoder," 2013 IEEE Inter. Conf. on Multimedia and Expo Workshop (ICMEW) (2013) 1 (DOI: 10.1109/ICMEW.2013.6618419).

1 Introduction

To efficiently compress the video to meet new application requirements, many complex video coding standards have been developed, such as H.264/AVC [1] and Audio Video Coding Standard (AVS) [2]. AVS, established by China Audio Video Coding Standard Working Group, has been accepted by ITU-TFGIPTV as an option for IPTV applications.

Reducing the required off-chip memory bandwidth and improving the memory access efficiency are the two key targets for optimized design. Many previous works, where can be classified into three methods, have been proposed to achieve these two targets. Lossless and lossy compression methods [3, 4] are proposed to minimize reference sample access. Advanced address mapping, arbitration, and controller scheduling [5] methods are proposed to improve memory access efficiency. The third method reduces the data access from architecture perspective, by trading off the external memory bandwidth using increased internal buffer size or data reusing for motion estimation between neighboring MBs [6, 7]. These methods can optimize the memory interface design to some extent, but they are not sufficient for AVS encoder memory interface design. This paper proposed three novel methods to reduce memory bandwidth and improve access efficiency.

The remainder of this paper is organized as follows. Section 2 presents our improved Level C+ HF_mV_n scheme with NOP insertion, which is proposed to reduce bandwidth and regularize the MB pipeline scheduling. In section 3, the interface design is optimized by jointly considering the memory bandwidth, internal buffer size and the MB pipeline structure. In section 4, advanced address mapping and arbitration methods are proposed to improve access efficiency. Section 5 shows the implementation results and section 6 concludes this paper.

2 Improved level C+ coding order

Level C data reuse scheme [5] is based on the search window of which two horizontal adjacent MBs can overlap most of the area except one MB column. For

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

the next MB, only one MB column needs to be loaded from off-chip memory, as shown in Fig. 1(a). Within a Level C+ HF_mV_n scheme as in Fig. 1(b), the reference window can be reused in two directions. This brings a benefit that after loading one MB column, all the n MBs located in the same slash line can perform motion estimation. On average, the bandwidth of each MB is reduced to $(2 \times SRV + n)/n$ from $(2 \times SRV + 1)$ of Level C, and the consumed on-chip memory is increased to $(2 \times SRV + n) \times (2 \times SRH + m)$ from $(2 \times SRV + 1) \times (2 \times SRH + 1)$.

In spite that level C+ is first proposed to reduce the bandwidth for IME, it has another advantage in alleviating the data dependency for MB pipelining. In AVS, rate distortion (RD) optimized is applied, where rate needs to calculate the consumed bits for motion vector (MV). MV predictor (MVP) for current block depends on the MV of left and top MBs after mode decision (MD) as Fig. 2(a). Accurate MVP can be obtained for level C+ coding order, thus achieve better coding quality than level C coding order.

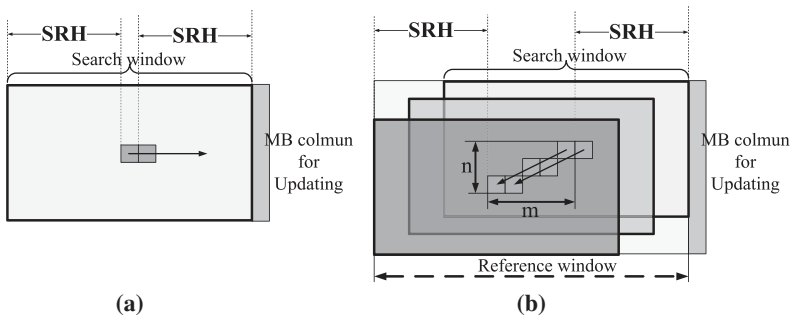

Fig. 1. (a) Level C data reuse. (b) Level C+ data reuse.

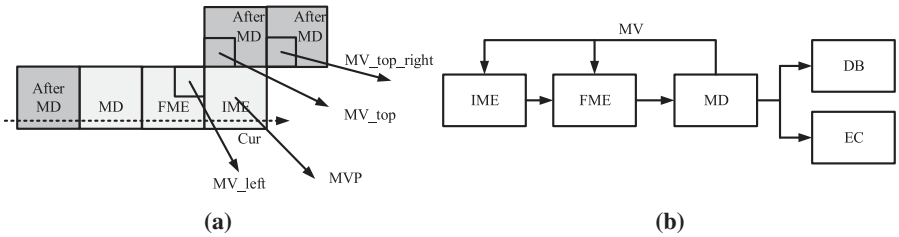

Fig. 2. (a) MB level MVP dependency. (b) Video encoder pipeline.

A popular video encoder pipeline is shown in Fig. 2(b), where integer motion estimation (IME) searches the best integer pixels, fractional motion estimation (FME) searches the best fractional pixels, mode decision (MD) decides the best mode from all intra/inter mode candidates, de-blocking (DB) eliminates the blocking effects and entropy coding (EC) codes the syntax to bit-stream. In raster coding order based encoder, when current MB is in IME stage, left MB is being processed in FME stage and left-two MB is at the MD stage. The top and top-right MB's MV after MD can be easily obtained. The pipeline and scan order cause the left MB's MV is impossible to get for MVP.

In Level C+ coding scheme, the three MVs can all be obtained if only the following three conditions are satisfied, which are conclusions in our previous work

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

[8]. The Level C+ coding can not only reduce bandwidth but also improve MB pipelining, if choosing an appropriate pair of parameter (m, n) for HF_mV_n coding scheme.

- Left MB's MV is available, if

$$(n+1) > (MD - ME + 1) \tag{1}$$

- Top-right MB's MV is available, if

$$((m-n)/(n-1) \times n + 1) > (MD - ME + 1) \tag{2}$$

- Top MB's MV is available, if

$$((m-n)/(n-1) \times n + n + 1) > (MD - ME + 1) \tag{3}$$

According to formulas from (1) to (3), HF_5V_3 zigzag stitch scheme can be used for AVS video encoder, where the search window is set to $[-128, 128] \times [-96, 96]$ for IME. However, it will cause irregular scanning. The irregular appears at the bottom stitch or at the beginning and ending in one stitch as in Fig. 3. In Fig. 3(a), there can be 3-MB-row stitch, 2-MB-row stitch, and 1-MB-row stitch at the end of a picture. In Fig. 3(b), the irregularity also happens at the beginning and ending of a stitch. These irregularities bring too much trouble in MB scheduling. Besides, MVP dependency cannot be removed in these corner conditions.

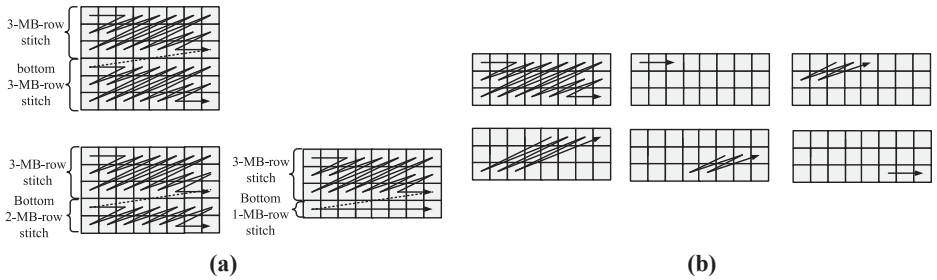

Fig. 3. (a) Irregular bottom stitch. (b) Irregularity in one stitch.

To regularize the MB scheduling and remove the MVP dependency, an improved slash scan coding order is proposed to make the whole image consist of only one slash pattern HF_5V_3. As illustrated in Fig. 4, NOP operations are inserted to keep regular coding order at the beginning and ending of stitch and the non-3-MB-row bottom stitch. Fig. 4(a) is an image with 2-MB-row stitch at the bottom, and Fig. 4(b) shows the proposed NOP insertion at the beginning of the stitch and the bottom stitch. Fig. 4(c) shows the final MB coding order, where the image consists of only one slash pattern. The NOP operations insertion can make the MB pipelining and scheduling easier and fully remove MVP dependency with no more than 6% bubbles increase in the worst case.

The proposed regular coding order can also bring benefits to the MB scheduling for search window (SW) updating. For HF5V3, one MB column should be loaded after coding three MBs in the slash line. For HF5V3, only 1/3 MB column of reference window needs to be loaded for IME in each MB period to balance bandwidth, as in Fig. 5(a). For the case of search range $[-128, 128] \times [-96, 96]$, only 5 MBs will be loaded.

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

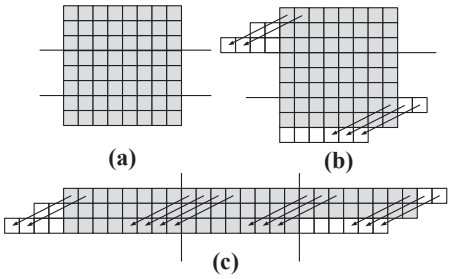

Fig. 4. Proposed regular slash coding order with NOP insertion.

When the SW is moving outside the right border of the picture as in Fig. 5(b), it is not needed to load the area outside of the picture. Instead, the reference data for next stitch are loaded. Based on the two schemes above, the SW updating can always load 5 MBs per MB. The improved scheduling makes it become easier to implement than our previous paper [8].

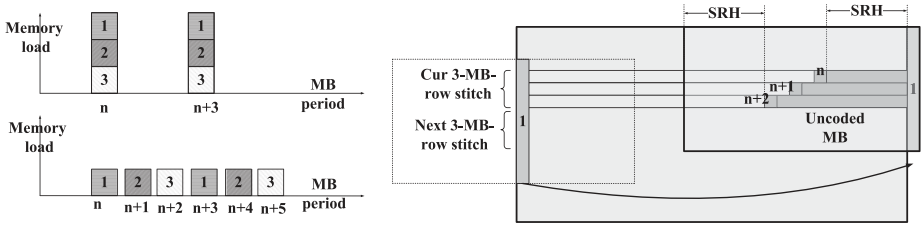

Fig. 5. (a) Bandwidth balance. (b) Pre-fetch for next stitch.

3 Memory interface design optimization

In this chapter, memory interface design is optimized by jointly considering the external memory bandwidth and the MB pipeline structure including internal buffer size, MB pipelining and scheduling.

3.1 Trade-off between external bandwidth and internal buffer

HF_5V_3 data reuse scheme is proposed to reduce the bandwidth in chapter 2, and the search range for IME is set to $[-128, 128] \times [-96, 96]$. The scheme can achieve 61% bandwidth reduction with 24064B on-chip buffer increase. Besides, the MV dependency can be fully removed using the Level C+ coding order with the NOP insertion. To trade off the external bandwidth and internal buffer, we define *RA* as the exchange efficiency using an internal buffer for bandwidth saving (4).

$$\mathrm{RA} = \mathrm{increased\_}\mathit{buffer} : \mathit{reduced\_bandwidth\_per\_MB} \tag{4}$$

Then, the RA value of HF_5V_3 data reuse is derived below.

$$\mathrm{RA}_{\mathrm{HF5V3}} = (21 \times 15 - 17 \times 13) \times 256 : (13 - 5) \times 256 = 11.75 : 1 \tag{5}$$

Like IME, FME module also needs reference samples to do motion estimation. For each sub-block, the used reference data for FME is just a small piece of area around the best integer MV. However, the best integer MVs for different sub-blocks may distribute in any place of the large search window of IME. This brings difficulties in the search window loading for FME. In our encoder, we use the

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

multi-resolution block matching method proposed in paper [9] to do IME. Experiments show that 48×48 local search window (LSW) is a good balance between internal buffer size and quality degradation from the reduction of direct or skip mode. The LUMA LSW for FME is loaded from SW in IME. For CHROMA interpolation and motion compensation (MC), the corresponding 24×24 CHROMA local search window (LSW) is loaded from external memory directly, as shown in Fig. 6.

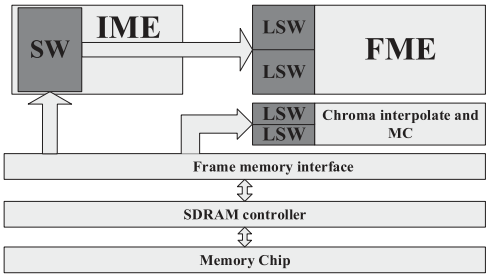

Fig. 6. Local search window for FME and CHROMA interpolation.

For intra prediction (IP), the up-row and left column pixels are used for candidate intra prediction modes. These data need to be saved either in on-chip or external memory. The required data for IP is 16×2 for LUMA and 8×2 for CHROMA in an MB. Because of HF_5V_3, 3 MB rows stitch can share a single above MB-row data buffer. We can get the RA for IP as in (6). This value is bigger than that of RA_{HF5V3}, so the data is stored in external memory.

$$RA_{IP} = (1 \times 1920 \times 2) : (16 \times 2 + 8 \times 2 \times 2)/3 = 180 : 1 \quad (6)$$

In AVS encoder, three rows and columns of pixels at the border of 8×8 block should be filtered to alleviate the block effects [2]. For current 8×8 block, it needs up and down 3×8 blocks, left and right 8×3 blocks. Extra double bandwidth will be consumed to store and reload these three lines if it is stored off-chip. Considering the burst length access alignment for DDR2, the read and write block size for each MB is 8×4 for CHROMA and 16×4 for LUMA. The RA value for DB module is shown in (7). Due to the large RA value, the data is stored into external memory.

$$RA_{DB} = (3 \times 1920 \times 2) : (16 \times 4 + 8 \times 4 \times 2) \times 2/3 = 135 : 1 \quad (7)$$

3.2 Level C+ encoder with synchronous memory interface

Based on the analysis and design above, our optimized encoder architecture jointly considering the memory interface, the internal buffer size, and the MB pipeline structure is given as in Fig. 7. The AVS encoder contains 6-stage pipeline, where the 1st stage is the PreFetch module in memory interface design, it loads CURRENT and REF samples for following stages. The 2nd stage is IME module to search the best integer motion vectors. The 3rd stage module is FME to get the best fractional motion vectors. The 4th stage decides the best mode from all intra/inter candidates and generates quantized coefficients and reconstruction samples for following stages. The 5th stage is the DB and EC (entropy coder) module, and the 6th stage is the data store modules in memory interface design. The frame-level

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

sync splice bit-stream module transfers zig-zag scan order bit-stream to standard compliant bit-stream. In Fig. 7, memory interface design transfers various data between encoding pipeline and external memory. It contains many sub-modules and spans several pipeline stages.

In Fig. 7, the MB scheduling module distributes MB information to all modules in AVS encoder and synchronize the pipeline operation. The MB scheduling module sends MB start signal to start-up the pipeline. The memory interface is also designed to be synchronized. It will not send memory request until the MB start signal is received, even if the request has been written into the request FIFO.

The synchronized memory system has two advantages. Firstly, it can promise a predictable and definite memory behavior to ease the analysis for memory access. The time interval between the start and idle is the memory response time for one MB and it should be less than the processing time for real-time encoding of one MB. Secondly, the synchronized memory interface design eases the design of arbitration algorithm since all memory requests are definite.

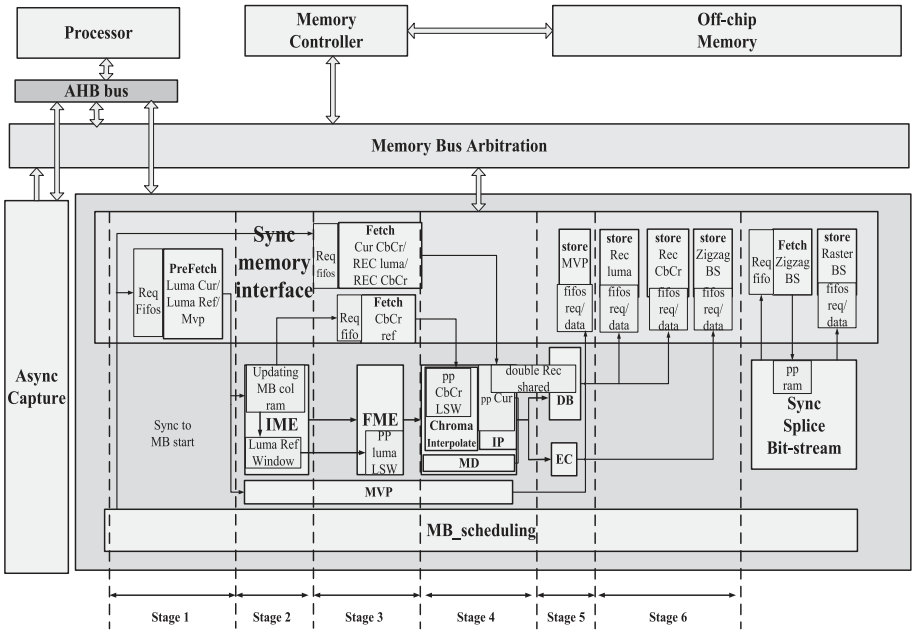

Fig. 7. Proposed Level C+ encoder architecture.

4 Address mapping and arbitration

4.1 Memory address mapping

Address mapping is to decide how the data is organized in the memory. In Fig. 7, the data of "CUR LUMA/CHROMA", "REF LUMA", "REF CHROMA", "BS", "MVP" need to be stored into external memory. The target is to improve the memory access efficiency by decreasing same bank different row access in DDR2 SDRAM [10].

4.1.1 CUR LUMA/CHROMA address mapping

In video encoder pipeline, MB CUR LUMA/CHROMA samples need to be loaded from external memory. However, these samples are written by capture module in

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

line-by-line raster scan order. This inconsistency cannot be eliminated completely. To improve the access efficiency for encoder pipeline, the CUR LUMA/CHROMA MBs are mapped into a continuous address space, even though it decreases 50% efficiency for capture writes. The mapping of an MB is shown as (8).

$$addr = (MBWpitch \times mby + mbx) \times w \times h/8 \tag{8}$$

where, *mbx* and *mby* are the horizontal and vertical index of MB, respectively. Both *w* and *h* are 16 for LUMA, *w* is set to 16 and *h* is set to 8 for CHROMA. *MBWpitch* is the MB number aligned to the power of 2 in the horizontal direction to facilitate address calculation. For 1920×1080, MB width is 120, so the *MBWpitch* is set to 128.

4.1.2 REF LUMA address mapping

LUMA reference is loaded by MB column style from the Level C+ data reuse scheme. Thus, MB-column data will be mapped continuously. The address mapping for an MB is shown as (9)

$$addr = (MBHpitch \times mbx + mby) \times 16 \times 16/8 \tag{9}$$

where, *MBHpitch* is the MB number aligned to the power of 2 in vertical direction. For 1920×1080, MB height is 68, so the *MBHpitch* is set to 128.

4.1.3 REF CHROMA address mapping

The REF CHROMA data is loaded in two-dimensional block style. So, it needs a two-dimensional address mapping, where is illustrated in Fig. 8. Firstly, the picture is split into slices in the vertical direction and each slice contains *blockH*'s MB. The *blockH* can be set to 4 or 8. Secondly, slice level address mapping is calculated based on formula (11), the horizontal neighboring MB will locate in different banks. Finally, cyclic address mapping formula (12) is proposed to achieve two-dimensional address mapping.

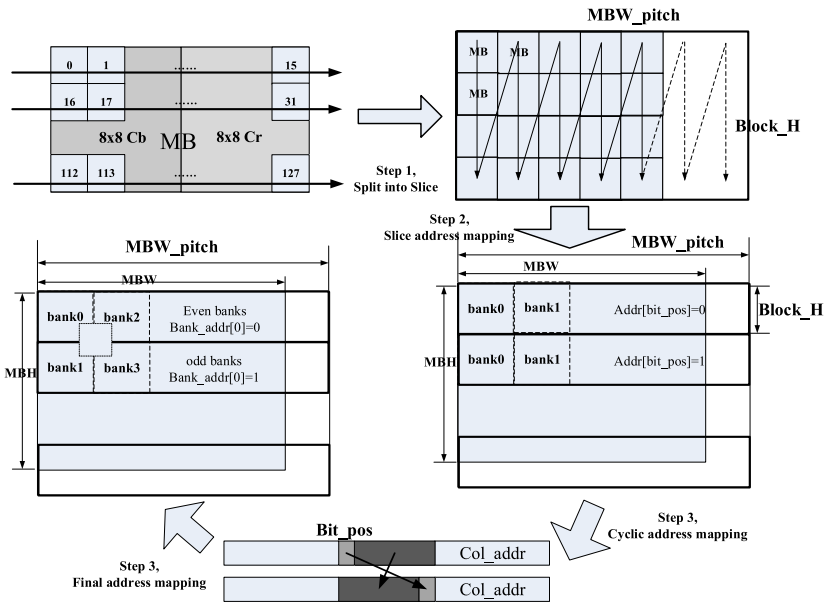

Fig. 8. Address mapping for REF CHROMA.

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

$$sliceSize = blockH \times MBWpitch \times 16 \times 8 \quad (10)$$

$$bitPos = \log_2(sliceSize)$$

$$sliceIdx = mby/blockH$$

$$innerMby = mby\%blockH$$

$$\text{addr} = sliceIdx \times sliceSize/8 + (blockH \times mbx + innerMby) \times 16 \times 8/8 \quad (11)$$

$$addr[bitPos : colBits] = \{addr[bitPos - 1 : colBits], addr[bitPos]\} \quad (12)$$

4.1.4 MV/BS address mapping

Only one MB data is accessed for MV and zigzag BS module. Thus, liner mapping is adopted. The address mapping for an MB is as (13). The *mbsize* is set to 32 for MV and set to 400 for BS.

$$addr = (MBWpitch \times mby + mbx) \times mbSize/8 \quad (13)$$

4.2 Arbitration

Two strategies are adopted in our arbitration scheme. The first is read client has a higher priority than write client. This will eliminate read/write client switch overhead. The second scheme is that address based arbitration is used to interleave the command to eliminate the same bank different row access [10]. By address mapping and arbitration optimizations, the memory access efficiency can improve around 12%.

5 Implementation results

The optimized memory interface design is successfully implemented within our 1080P@45fps AVS encoder in Xilinx Virtex-6 FPGA with a 64-bit width DDR2 SDRAM controller at an operating frequency of 200 MHz. The HD AVS encoder can support a large search range $[-128, 128] \times [-96, 96]$ with maximum two reference frames for IME.

In memory interface design, fetch client will pass the read data to the next pipeline stage directly. Store clients will use 2MB size RAM or FIFO to buffer the store data. 64B memory is used for buffering MV writing. 640B memory is used for buffering DB LUMA writing, and 384B of memory is used for buffering DB CHROMA writing. 768 bytes of memory is used for buffering zigzag BS writing, considering that compressed BS is same as the MB size in worst case. The gate count for the proposed memory interface design is 117K.

Table I summarizes the required bandwidth and consumed DDR2 cycles for one MB. Due to Level C+ scanning order, 5MBs are loaded from memory for IME module. An aligned window about 64x26B is needed for CHROMA interpolation. MV is stored at P frame and loaded at B frame, so the MV loading and storing will not happen simultaneously. Supposing the compression ratio is 12:1, BS data store for each MB is 32 bytes. But 400 bytes BS data should be entirely loaded for reordering because the actual BS size cannot know in advance. The total DDR2 cycles consumed for an MB is 465. It can achieve 1080p@45fps under 200 MHz frequency.

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

Table I. Consumed bandwidth and cycles for one MB

Interface	Data Types	Size (bytes)	Burst Cycles
Fetch → IME	Cur Y out	16 × 16	16
Fetch → INTRA	Cur C out	16 × 8	8
Fetch → IME	Ref 1 Y out	16 × 16 × 5	80
Fetch → IME	Ref 2 Y out	16 × 16 × 5	80
Fetch → interpolation	Ref 1 C out	64 × 26	104
Fetch → interpolation	Ref 2 C out	64 × 26	104
Mem ↔ MVP	MV in/out	32	2
MEM → DB	Rec y out	16 × 4	4
MEM → DB	Rec c out	16 × 4	4
DB → Mem	Rec Y in	16 × 20	20
DB → Mem	Rec C in	16 × 12	12
EC → Mem	Zigzag BS in	32	2
Mem → Splice	Zigzag BS out	400	25
Splice → Mem	Raster BS in	32	2
Mem → stream channel	Raster BS out	32	2
Total	-	7440	465

Finally, the quality comparison between Level C and Level C+ HF5V3 video encoder is shown as in Fig. 9. The test is on sequence *BQTerrace*. Our proposed Level C+ encoder can achieve about 0.15 db gain improve than traditional Level C video encoder. The gain is from accurate MVP for IME, FMD and MD modules.

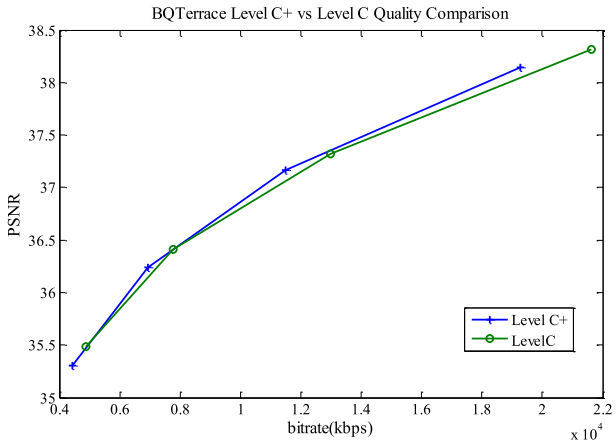

Fig. 9. The quality comparison of Level C+ and Level C encoder.

6 Conclusion

In this paper, a highly efficient memory interface is designed for AVS HD video encoder. The memory interface is optimized considering both the encoder architecture and the SDRAM access features. Three novel methods are proposed to

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017

achieve the targets of reducing the required off-chip memory bandwidth and improving the memory access efficiency. The memory interface is successfully implemented in our 1080P@45fps AVS encoder with Xilinx Virtex-6 FPGA at an operating frequency of 200 MHz. The consumed gate count is 117k.

DOI: 10.1587/elex.14.20170501
Received May 14, 2017
Accepted May 22, 2017
Publicized June 12, 2017
Copyedited June 25, 2017