Bandwidth Optimized Motion Compensation Hardware Design for H.264/AVC HDTV Decoder

Chuan-Yung Tsai, Tung-Chien Chen, To-Wei Chen, and Liang-Gee Chen
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering,
National Taiwan University, Taipei, Taiwan; Email: cytsai@video.ee.ntu.edu.tw

***Abstract*— Design of H.264/AVC motion compensation (MC) is very challenging through the high memory bandwidth and low hardware utilization caused by the new functionalities of variable block size and 6-tap interpolation filter. In this paper, the Vertically Integrated Double Z (VIDZ) schedule, and Interpolation Window Reuse (IWR) and Interpolation Window Classification (IWC) bandwidth reduction schemes are proposed to keep the MC highly utilized and save 60-80% memory bandwidth. The hardware of proposed MC is implemented at 120MHz with 47K logic gates and can support 2048×1024 30fps H.264/AVC HDTV decoder with less than 200MB/s memory bandwidth.**

I. Introduction

H.264/AVC [1] is the new generation video coding standard developed by the Joint Video Team (JVT), which consists of experts from ITU-T VCEG and ISO/IEC MPEG. H.264/AVC can save about 25-45% bit-rate compared to MPEG-4 Advanced Simple Profile (ASP). The ultra high coding efficiency comes from many new features, including sub pixel inter prediction with variable block size (VBS) and multiple reference frames, intra prediction, and context-based adaptive entropy coding—CAVLC and CABAC. However, its overall computational complexity also increases greatly such that an H.264/AVC decoder requires two times the computational power of an MPEG-4 decoder. According to the runtime analysis of H.264/AVC decoder software, the MC can use up to 55% of total decoding time. This explains the necessity of hardware acceleration for MC in an H.264/AVC decoder.

MC hardware design for an H.264/AVC decoder is much more difficult than those for previous standards. Memory bandwidth and hardware utilization are the two responsible problems caused by H.264/AVC new coding features—6-tap interpolation filter for luma sub pixels and variable block size (VBS). During luma quarter pixel interpolation, a large amount of frame memory accesses is required due to the combined effect of very small block size (e.g. 4×4) and 6-tap interpolation filter, and the memory bandwidth requirement can be 80% more than MPEG-4 ASP's. The variety of block sizes also makes the MC hardware engine need to be carefully designed and scheduled to keep the hardware utilization.

In this paper, we propose a bandwidth optimized MC hardware design for H.264/AVC HDTV decoder system. Using the proposed interpolation flow and bandwidth reduction schemes, we successfully design an MC with high hardware utilization and low bandwidth requirement (60-80% bandwidth saving). The proposed MC can support 2048×1024 30fps H.264/AVC HDTV decoder with 47K logic gates and less than 200MB/s memory bandwidth, when operating at 120MHz. The rest of this paper is organized as follows. In Section II, we describe H.264/AVC new features and the resultant problems in designing MC. Section III and IV present the proposed methods and architecture. Simulation and implementation results are shown is Section V and VI. Finally, Section VII gives the conclusion.

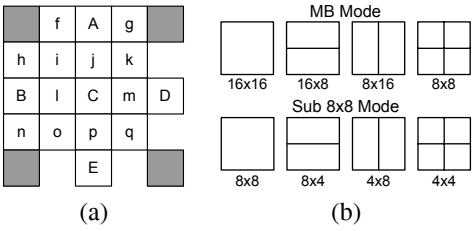

Fig. 1. (a) Integer pixels (gray blocks), half pixels (blocks with upper-case letters), and quarter pixels (blocks with lower-case letters) of luminance. (b) H.264/AVC block modes of VBS.

II. Problem Statement

In H.264/AVC, the difficulties of MC hardware design arise from the 6-tap sub pixel interpolation filter and VBS. The 6-tap finite impulse response (FIR) filter uses [1 -5 20 20 -5 1] as factors for interpolating luma half pixels, and luma quarter pixels are average of two integer/half pixels. For the chroma part, bilinear interpolation filter is used to generate 1/8 pixels. Positions of each kind of luma pixels are shown in Fig. 1(a) for later references. VBS allows four macro-block (MB) modes as illustrated in Fig. 1(b)—16×16, 16×8, 8×16, and 8×8; for an 8×8 block, it can be further partitioned as 8×4, 4×8, or 4×4 mode. Under these features, interpolation of an $X \times Y$ luma quarter pixel block requires $(X+5)\times(Y+5)$ integer pixels.

During the designing of H.264/AVC MC hardware, memory bandwidth and hardware utilization are the two main problems cause by H.264/AVC new coding features. A large amount of frame memory accesses is required due to the very small block size (e.g. 4×4) and 6-tap interpolation filter. The MC hardware also needs great effort in designing and scheduling to keep its utilization for supporting 7 different block sizes. To solve these two problems, two basic hardware architectures can be firstly considered—4×4-based and 16×16-based MC hardware. A 4×4-based MC is very ideal for hardware utilization since all VBS can be decomposed into 4×4 ones. However this causes serious overhead in bandwidth (or memory access number), because for every 4×4 block, a 9×9 interpolation window (the

0-7803-9197-7/05/$20.00 © 2005 IEEE.

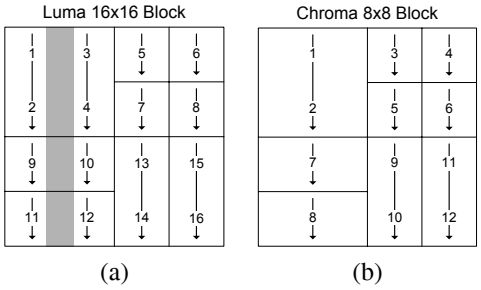

Fig. 2. Example of VIDZ interpolation flow of: (a) luma block (buffer shaded region for horizontal reuse); (b) chroma block.

integer pixel block used during interpolation) is required. For our objective specification—2048×1024 30fps HDTV decoder operating at 120MHz with 32-bit system bus, at least 432 cycles among 488 total usable cycles will be required to finish only luma MB memory access and interpolation. The timing budget will be impossible for processing chroma pixels. On the contrary, a 16×16-based MC requires the minimal bandwidth because it can cover all interpolation data for any VBS, without reloading pixels between decomposed blocks. But its average hardware utilization is obviously low, especially when the MB consists of smaller blocks.

To solve the memory bandwidth and hardware utilization problems efficiently, we need to propose an MC hardware architecture with the hardware utilization (and area cost) of 4×4-based MC, and the minimal bandwidth of 16×16-based MC. Reference [2] presented an MC hardware for 1920×1088 30fps HDTV decoder operating at 100MHz. However, considering the integration of a complete decoder system, its timing budget may be too tight for inter-module data transfer. Therefore, sufficient timing margin should be included in this MC hardware design.

III. Proposed Methods

In this paper, one unified processing flow and two bandwidth reduction schemes are proposed for the hardware architecture of bandwidth optimized MC. We first adopt the 4×4-based MC as the basic architecture. With the 4×4 block decomposition strategy [3], all types of VBS are decomposed into 4×4 ones, and then can be processed by this MC unit with full hardware utilization. The bandwidth requirement of 4×4-based MC can be minimized by the two bandwidth reduction schemes.

The proposed unified processing flow of decomposed 4×4 blocks is called *Vertically Integrated Double Z* (VIDZ) flow; an example is shown in Fig. 2. It can provide a regular schedule for the MC unit and help to save all vertical redundant memory accesses (about 25-35% of total bandwidth). Inside one block, the vertical scan order is adopted for easier integration of common interpolation operations between two vertically decomposed 4×4 blocks. Between different blocks (original ones, not decomposed), the H.264/AVC native double Z scan is used as the processing order for all MB modes and sub 8×8 modes. Worth to note, 4×2 block decomposition is used for chroma blocks to make luma and chroma MC units

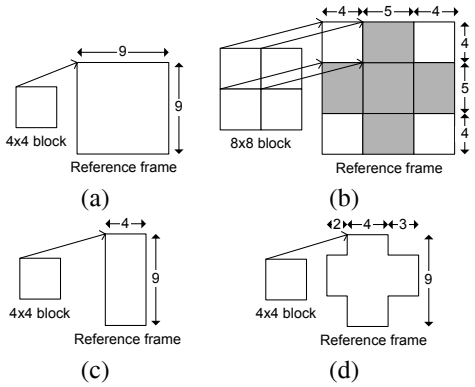

Fig. 3. (a) General case interpolation window. (b) Four interpolation windows for an 8×8 block (shaded region means reusable). (c) Interpolation window when MV pointing to horizontal integer pixels. (d) Interpolation window when MV pointing to quarter pixel i.

have good hardware reusability. Utilization of the chroma MC is halved only when processing 4×8 or 4×4 blocks.

In the following paragraphs, we will present the two bandwidth reduction schemes. With both schemes, the bandwidth requirement of 4×4-based MC is further reduced to minimum.

A. Interpolation Window Reuse Scheme

The first scheme is *Interpolation Window Reuse* (IWR). As described in Section II, MC of an H.264/AVC decoder can have very high memory bandwidth requirement. A straightforward memory access scheme is to process every decomposed luma 4×4 blocks as the general case—always load a 9×9 interpolation window for a decomposed luma 4×4 block, as shown in Fig. 3(a). But its performance is just the same as the failed case mentioned in Section II.

In order to minimize the bandwidth requirement, we must design an MC hardware capable of reusing all common interpolation data between decomposed luma 4×4 (and chroma 4×2) blocks. Although an all-4×4-block MB can have all 16 interpolation windows spaced out in the reference frames, yet this kind of MB is not frequent. In other words, there tends to exist overlapped regions between interpolation windows of decomposed blocks for most MBs. As the example of an 8×8 block shown in Fig. 3(b), the shaded regions are required by more than one decomposed block and thus can be reused. Based on the VIDZ flow, all vertically overlapped interpolation windows can be easily reused. Therefore, the IWR scheme only adds a 21×64-bit on-chip memory to save horizontally overlapped interpolation windows of decomposed blocks. The on-chip memory is referred to as horizontal reuse memory in later sections. More details of the IWR scheme will be given in Section IV.

B. Interpolation Window Classification Scheme

The second scheme is *Interpolation Window Classification* (IWC). After examining the interpolation formulas of luma sub pixels at different positions, we can find the interpolation window is not always $(X+5)\times(Y+5)$ for a luma $X\times Y$

TABLE I

SUMMARY OF INTERPOLATION WINDOW CLASSIFICATION

Luma Sub Pixel Type	Interpolation Window Size
Integer Pixel	$X \times Y$
Pixel *f*, *A*, *g*	$Y \times (X+5)$
Pixel *h*, *B*, *n*	$X \times (Y+5)$
Pixel *i*, *k*, *o*, *q*	$(X+5) \times (Y+5) - 25$
Others	$(X+5) \times (Y+5)$

Chroma Sub Pixel Type	Interpolation Window Size
Integer Pixel	$X \times Y$
Horizontal-Integer Pixel	$X \times (Y+1)$
Vertical-Integer Pixel	$Y \times (X+1)$
Others	$(X+1) \times (Y+1)$

Window size X is for width, and Y is for height.
Refer to Fig. 1(a) for luma sub pixel position labels.

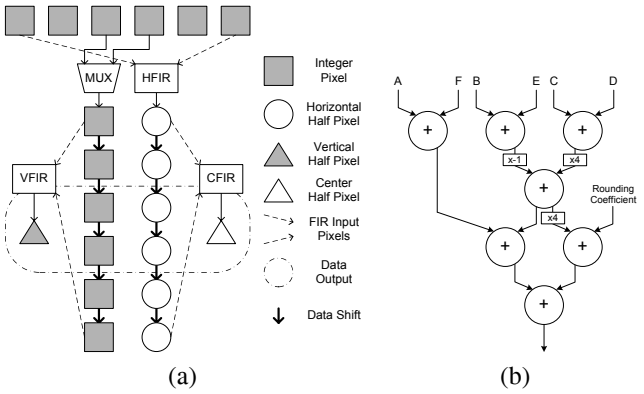

Fig. 4. (a) Luma interpolation unit. (b) Adder tree based 6-tap FIR.

block. For example, a luma 4×4 block with MV pointing to horizontal-integer pixels—pixel *h*, *B*, or *n* in Fig. 1(a)—should have an interpolation window sized 4×9 as shown in Fig. 3(c). This is because its sub pixel interpolation procedure involves no horizontal filtering, and thus no horizontal 5-pixel extension of interpolation window is needed. Figure 3(d) gives another example of interpolation at quarter pixel *i*. Value of pixel *i* is the average of half pixel *A* and *B*, which are filtered from horizontal and vertical strips of pixels respectively, and the cumulated strips form a cross-shaped interpolation window.

In brief, the IWC scheme aims to precisely control the MC hardware to load a smaller and exact interpolation window, rather than a general one. For luminance, four classes of reduced interpolation windows with different bandwidth-saving levels are derived from the H.264/AVC interpolation formulas. The complete classification is summarized in Table I; there are three classes of reduced chroma interpolation windows. With few added control logics, the IWC scheme can further provide about 10-20% bandwidth reduction.

IV. ARCHITECTURE

Before looking into the complete architecture of MC, we first present the most fundamental circuit of MC—the luma interpolation unit. The 2D 6-tap luma half pixel filter can

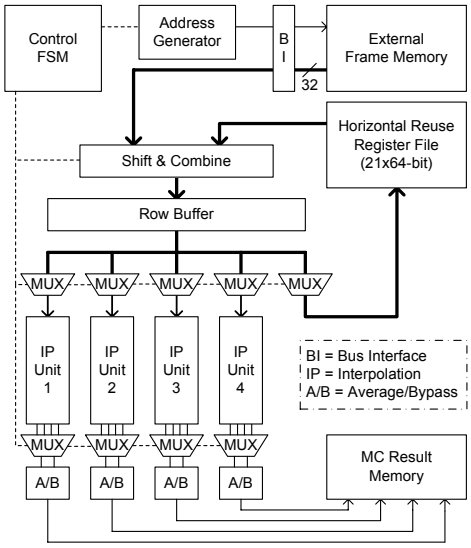

Fig. 5. Block diagram of proposed MC hardware.

be decomposed into three 1D 6-tap filters (HFIR, VFIR, and CFIR), as shown in Fig. 4(a). Based on the VIDZ flow, the interpolation unit has good data reusability, because the two vertical shift register arrays as VFIR/CFIR input buffers keep all intermediate values across vertically decomposed blocks. Worth to note, there is always only one vertical-half pixel (either *B* or *D*) involved in a block's interpolation; therefore the unit needs only one shift register array for integer pixels, with the input selected by the multiplexer (MUX). The adder tree based 6-tap FIR filter as shown in Fig. 4(b) is improved from our previous work [3] and mapped into pipeline stages for running at 120MHz. The utilization of interpolation unit also benefits from the VIDZ flow and IWR scheme greatly such that it can even achieve 100%.

Figure 5 is the block diagram of proposed MC architecture. The architecture employs a central control finite state machine (FSM) to generate control signals for all components. Firstly, the control FSM translates MB information input for address generation of the external frame memory. Then, the control signals for data path are passed through a delay-reconfigurable control signal pipeline to synchronize with the reference frame data. All data are loaded consecutively without bubble to help increasing the memory bus utilization and reducing total cycles for MC. Worth to note, the IWR scheme functions according to *block/sub-block mode*, while the IWC scheme according to *motion vector value*.

The data path starts from a shift-and-combine circuit, which selects and packs the pixels loaded from external frame memory and horizontal reuse memory into the 12-byte row buffer. The required size of row buffer is determined as illustrated in Fig. 6; 9 pixels for luma interpolation (of one row of four sub pixels) and at most 3 preloaded pixels (possibly data for later interpolation) must be stored. Four interpolation units form a processing element (PE) for supporting both luma and

TABLE II
SIMULATION RESULTS OF BANDWIDTH REDUCTION SCHEMES

Access Per MB / Reduction	IWR-Vertical	IWC	IWR-Vertical + IWC	IWR	IWR + IWC
Akiyo	346 / 35%	127 / 76%	121 / 77%	207 / 61%	113 / 79%
Foreman	391 / 26%	311 / 41%	252 / 53%	252 / 53%	184 / 65%
Mobile Calendar	401 / 25%	335 / 37%	288 / 46%	262 / 51%	198 / 63%
Stefan	402 / 25%	343 / 36%	284 / 47%	263 / 51%	201 / 62%
Table Tennis	363 / 32%	204 / 62%	176 / 67%	222 / 58%	143 / 73%

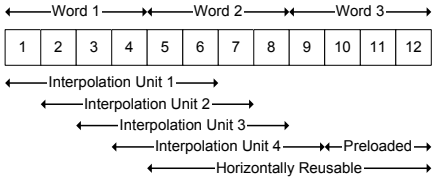

Fig. 6. Maximum usage of row buffer (nine pixels for interpolation distributed in 4-4-1 fashion across three RAM words and three pixels preloaded).

chroma decomposed blocks. During the VIDZ flow, every time the row buffer is filled with necessary amount of pixels, all interpolation units pull corresponding data in and push results out for luma quarter pixels (via average) or other type of sub pixels (via bypass). Meanwhile, at most eight pixels (64 bits, as shown in Fig. 6) in the row buffer need to be written into the horizontal reuse memory. For luma 16×16 or 8×16 blocks, there should be 21 (16+5) words in the the horizontal reuse memory. Therefore, the horizontal reuse memory sizes 21×64 bits in order to fully support the IWR scheme. This memory specification is more suitable to be implemented as register file than as SRAM, for the chip area consideration. Finally, all luma and chroma MC results are buffered in the MC result memory, and then combine with residues to complete the P-frame reconstruction loop.

V. SIMULATION RESULTS

To evaluate the resultant performance of proposed bandwidth reduction schemes for MC, we modify the H.264/AVC reference software JM 8.2 [4] decoder for simulating the MC memory access. Simulation results are listed in Table II. Five CIF sized video sequences are simulated with quantization parameter set to 20, 30, and 40 for averaging the results. We compare five combinations of IWR-Vertical, IWR, and IWC schemes; the IWR-Vertical scheme equals the IWR scheme without adding horizontal reuse memory (i.e. can be covered by the VIDZ flow). The complete bandwidth reduction scheme (IWR+IWC) apparently can save the most memory bandwidth. From the last column of Table II, the bandwidth optimization result is about 60-80%, and the contribution of each proposed method is as mentioned in Section III. In the 2048×1024 30fps H.264/AVC HDTV decoder specification, the resultant memory bandwidth of proposed MC hardware is less than 200MB/s; the simulated timing budget is also very sufficient (about 100 cycles), which implies the proposed MC is able to support even higher H.264/AVC HDTV decoder specification.

TABLE III
MC INTERPOLATION UNIT HARDWARE COMPARISON

	Component	Gate Count	Optimization	Clock Rate
Ref. [2]	FIR×13 Bilinear×2	20686	30%	100MHz
Proposed	FIR×12 Bilinear×4	21506 & Reg File	60-80%	125MHz

VI. IMPLEMENTATION RESULTS

The proposed MC hardware is implemented in Verilog HDL and synthesized using Synopsys Design Compiler with TSMC 0.18μm cell library, with its clock rate set to 125MHz. The total MC gate count is 46,646 gates. This number includes the synthesized register file for horizontal reuse memory (15,197 gates), which may be reduced using a register file macro. Table III gives a comparison of the MC key hardware—interpolation unit. Although our work is larger in area, yet our performance is better optimized. Our work also has been integrated into an H.264/AVC HDTV decoder system [5] and verified to function correctly; the system memory bandwidth is saved by 40-50%.

VII. CONCLUSION

In this paper, we propose a memory bandwidth optimized MC hardware design for H.264/AVC HDTV decoder supporting 2048×1024 30fps high definition videos, with 47K logic gates and less than 200MB/s memory bandwidth when operating at 120MHz. Using the proposed VIDZ interpolation flow and IWR/IWC bandwidth reduction schemes, we have successfully designed an H.264/AVC MC with high hardware utilization and low bandwidth requirement. In total, about 60-80% memory bandwidth reduction is achieved.

REFERENCES

[1] Joint Video Team, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[2] S. Z. Wang, T. A. Lin, T. M. Liu, and C. Y. Lee, "A new motion compensation design for H.264/AVC decoder," in *Proc. of Int. Symposium on Circuits and Systems (ISCAS'05)*, 2005, pp. 4558–61.

[3] T. C. Chen, Y. W. Huang, and L. G. Chen, "Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC," in *Proc. of Int. Conf. on Accoustics, Speech, and Signal Processing (ICASSP'04)*, 2004, pp. V – 9–12 vol.5.

[4] *Joint Video Team H.264/AVC Reference Software, version JM 8.2*. http://iphome.hhi.de/suehring/tml/download/.

[5] T. W. Chen, Y. W. Huang, T. C. Chen, Y. H. Chen, C. Y. Tsai, and L. G. Chen, "Architecture design of H.264/AVC decoder with hybrid task pipelining for high definition videos," in *Proc. of Int. Symposium on Circuits and Systems (ISCAS'05)*, 2005, pp. 2931–34.