PAPER Special Section on Circuits and Design Techniques for Advanced Large Scale Integration

A High Performance and Low Bandwidth Multi-Standard Motion Compensation Design for HD Video Decoder

Xianmin CHEN†, Peilin LIU†a), Dajiang ZHOU††, Jiayi ZHU†, Xingguang PAN†, *Nonmembers*, *and* Satoshi GOTO††, *Fellow*

SUMMARY Motion compensation is widely used in many video coding standards. Due to its bandwidth requirement and complexity, motion compensation is one of the most challenging parts in the design of high definition video decoder. In this paper, we propose a high performance and low bandwidth motion compensation design, which supports H.264/AVC, MPEG-1/2 and Chinese AVS standards. We introduce a 2-Dimensional cache that can greatly reduce the external bandwidth requirement. Similarities among the 3 standards are also explored to reduce hardware cost. We also propose a block-pipelining strategy to conceal the long latency of external memory access. Experimental results show that our motion compensation design can reduce the bandwidth by 74% in average and it can real-time decode 1920x1088@30 fps video stream at 80 MHz.

key words: *high performance, low bandwidth, motion compensation, multi-standard, 2-D cache*

1. Introduction

Motion compensation (MC) is widely used in many video compression standards. In recent standards, such as H.264/AVC [1] and Chinese AVS [2], more complex MC algorithms are introduced to achieve high compression rate. In MC hardware design, 3 major difficulties exist. The first one is the huge bandwidth requirement caused by complex interpolation algorithms. The second problem is the long access latency of external memory, because accessing external memory (SDRAM) needs activating and pre-charging. The third problem is the computation complexity, which exists mainly in interpolation and weighted prediction. To design high performance MC hardware, we must overcome the 3 difficulties.

Many researches have been done about MC architecture design. In [3], C. Lee proposed a H.264 MC architecture, where 2-dimensional circular register files were used to reduce bandwidth. K. Luo designed a high throughput bandwidth optimized AVS MC architecture, which used Vertical Z processing order for reference data re-use to save memory bandwidth [4]. J. Zheng proposed a novel VLSI architecture of MC for multiple standards [5]. Special efforts have also been taken by many researchers to reduce bandwidth, conceal latency and design high performance interpolator in [6]–[9].

Manuscript received July 13, 2009.
Manuscript revised September 27, 2009.

†The authors are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China.
††The authors are with IPS, Waseda University, Kitakyushu-shi, 808-0135 Japan.
a) E-mail: liupeilin@sjtu.edu.cn

DOI: 10.1587/transele.E93.C.253

Problems with previous MC designs are: (1) potentials of external memory bandwidth reduction are not fully explored; (2) cache schemes are not optimal for hardware implementation; (3) prefetch methods used before for latency concealment suffer from problems when prediction in prefecth fails; (4) similarities among standards are not fully used when multiple standards support is needed.

In this work, we propose a MC design supporting H.264, MPEG-1/2 and AVS standards. Our design has 3 main features: low bandwidth, high performance and high hardware sharing rate among standards. Low bandwidth of our design comes from a 2-Dimensional (2-D) cache. Parameter design of 2-D cache is discussed in this paper. We also propose a block-pipelining strategy to conceal the long latency of external memory access. This strategy, combined with our high throughput Interpolator and Weighted Predictor, can greatly enhance the performance of our MC design. Experimental results show that our motion compensation design can reduce the bandwidth by 74% in average and it can real-time decode 1920x1088@30 fps video stream at 80 MHz. We also explore similarities among standards to make our MC design support multiple standards with relatively small area cost.

The rest of the paper is organized as follows. Section 2 presents the 2-D cache employed in our MC design to reduce bandwidth. Section 3 gives the proposed MC hardware architecture. In Sect. 4, experimental results are presented. Section 5 concludes this paper.

2. 2-D Cache for Motion Compensation

2.1 Block Size Unification for 3 Standards

Block size unification is the preparation work for multi-standard support in MC hardware. MPEG-1/2, H.264, AVS all use variable block sizes. The minimum block size in H.264, MPEG-1/2, and AVS are 4×4, 16×8 and 8×8 respectively. We unify block sizes into basic block types for hardware implementation. Since the smallest block size in 3 standards is 4×4, we adopt it as one basic block type. To reduce redundant data transferring caused by using 4×4 only, we also adopt another larger basic block type 4×8. For example, if two vertical adjacent 4×4 blocks belong to the same partition in H.264 stream and fractional part of MVY is not 0, transferring reference data of two 4×4 blocks row by row needs $9 \times 2 = 18$ cycles, while transferring one

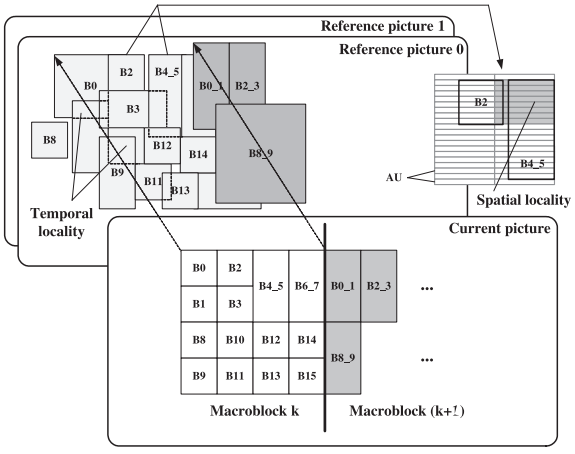


Fig. 1 Locality in motion compensation.

4×8 block needs only 13 cycles. Both basic block types above are in luma, and they are 2×4 and 2×2 in chroma respectively.

2.2 Locality in Motion Compensation

Figure 1 is an example of reference picture blocks (reference block) needed in MC by MB k and MB (k+1) of current picture. In Fig. 1, some regions of the reference picture are shared by several basic blocks. Thus, there is a high probability that the reference block needed by the current basic block will also be used by subsequent basic blocks. This locality in MC is Temporal Locality, as shown in Fig. 1.

Another type of locality is caused by storage of data in external memory. Several pixels are stored together in external memory called Access Unit (AU). In this paper, AU size is 8 bytes, which contains a luma 8x1 block, or a combination of 4x1 Cb block and 4x1 Cr block. An AU is read from external memory even when part of it is required, and there is a high probability that the non-required part of this AU will be required soon by subsequent basic blocks. This locality in MC is called Spatial Locality, as shown in Fig. 1. Both localities in MC are in 2-D space, different from 1-dimensional situation in computer [10].

2.3 2-D Cache and Parameters Design

We employ a 2-D cache to utilize localities in MC. Mapping from external memory to cache takes place in 2-D space. The mapping mechanism is similar to that of computer cache except that address used for mapping has two dimensions, as shown in Figs. 2(a), (b). Hereafter, cache mentioned in this paper refers to 2-D cache.

To construct a high performance cache, we need study cache parameter design here. We fix cache line to one AU (8 bytes) in the following discussion. Two parameters that we need to determine for cache are:

- Associativity (A) — Direct (sperate forward & backward) or 2-way set associative (2 W FIFO);

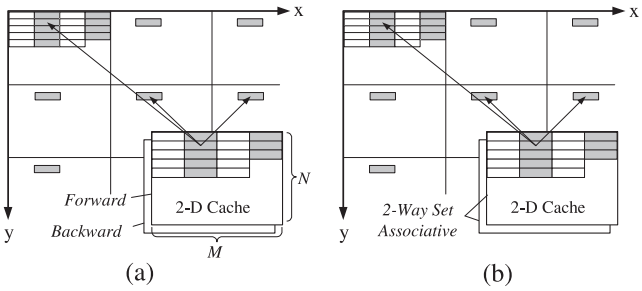


Fig. 2 (a) Direct (separate forward and backward) 2-D cache (b) 2-way set associative (2 W FIFO) 2-D cache.

Table 1 H.264 test streams MV statistics.

Stream	MV Range $\sqrt{MVX\_int^2 + MVY\_int^2}$			
	[0,32)	[32, 64)	[64, 128)	[128, ∞)
Flower	97.2%	2.5%	0.3%	0%
BigShips	99.8%	0.2%	0%	0%
Driving	86.4%	10.3%	3.0%	0.3%
WhaleShow	90.4%	3.8%	2.9%	2.9%

- Size (S) — $S = M \times N \times 2$ ($M \times N$ for short), M: Width, N: Height.

Since H.264 MC needs larger reference block than the other two standards do when MV is the same, bandwidth requirement is larger than the other two standards almost all the time. Thus, conclusion drawn for H.264 can also apply to the other two standards. Table 1 shows MV statistics of H.264 test streams, where MVX_int and MVY_int are integer part of luma MVX and MVY respectively. Resolution of the 4 streams are 720×576, 1280×720, 1920×1024 and 1920×1024 respectively and all of them have "IPBPB..." pattern and at most 4 reference pictures.

We define bandwidth reduction ratio (R_c) by Eq. (1), where B_c is bandwidth of MC with cache using 4x4 and 4x8 blocks, B is bandwidth of MC without cache using only 4x4 blocks.

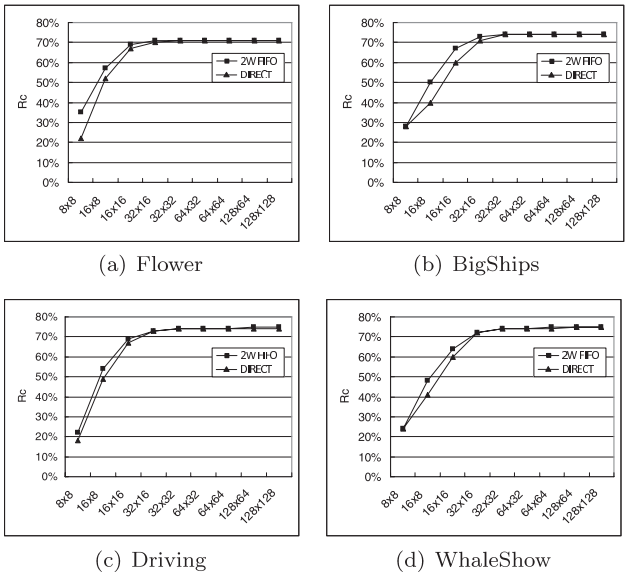
$$R_c = (B - B_c)/B \tag{1}$$

With modified JM9.3 [12] decoder model, we test R_c for different cache parameters. Figure 3 shows R_c for streams in Table 1 with different cache parameters. X-axis in Fig. 3 is cache size, which ranges from 8×8 to 128×128.

From Fig. 3, we can draw the following conclusions:

- R_c increases with the cache size, but the increasing rate becomes slow with the growth of cache size.
- 2 W FIFO cache shows advantage against direct cache in R_c. However, the advantage becomes very small when $S \geq 32 \times 32$.

Since R_c increases very slow when $S \geq 32 \times 32$, we choose $S = 32 \times 32$ (4×32 in AU) for hardware implementation. In addition, noticing that R_c of direct cache is nearly the same as that of 2 W cache when $S \geq 32 \times 32$, and hardware cost of direct cache is much smaller than 2 W FIFO cache, we select direct mapping for cache in hardware implementation. Taking chroma and dual directions



(a) Flower (b) BigShips

(c) Driving (d) WhaleShow

Fig. 3 R_c of different 2-D cache parameters.

into account, storage space in cache is divided into four parts: luma-forward, luma-backward, chroma-forward and chroma-backward. Each luma part contains one 32×32 space, each chroma part contains two 16×16 spaces (One for Cb and one for Cr).

3. Proposed Motion Compensation Architecture

The block diagram of the proposed MC architecture is shown in Fig. 4. MC architecture consists of a 2-D Cache, a Pel Shifter, an Interpolator and a Weighted Predictor. Dataflow in our MC hardware is as follows. First, 2-D Cache reads reference picture pixels from external memory according MV and PIC_ID (Picture Slot Index). Then, pixels from 2-D Cache are processed by Shifter, Interpolator and Weighted Predictor sequentially to generate predicted pixel data. Pipeline of our MC architecture is in block level (4×4 and 4×8 basic blocks), which is shown in Fig. 5. Pel Shifter, Interpolator and Weighted Predictor contain one pipeline stage each, while 2-D Cache contains several stages, which will be discussed later. Different pipeline stages can work on different basic blocks at the same time, making hardware run efficiently.

3.1 2-D Cache Module Design

3.1.1 Latency Concealment in 2-D Cache

To solve the latency problem, we propose a block-pipelining strategy. Basic idea of this strategy is to conceal external memory latency with a well-designed pipeline. To explain the how to design this pipeline, we first divide the processing of each basic block in cache into four steps.

- CALC: Calculate the position and size of reference block;

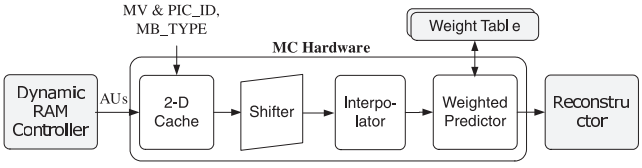


Fig. 4 Block diagram for motion compensation.

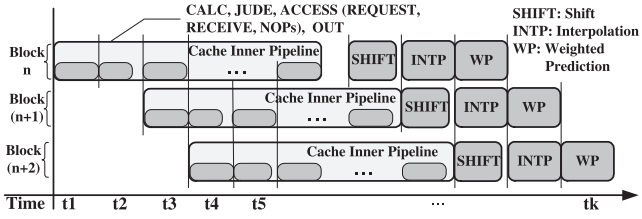


Fig. 5 Block level pipeline in motion compensation.

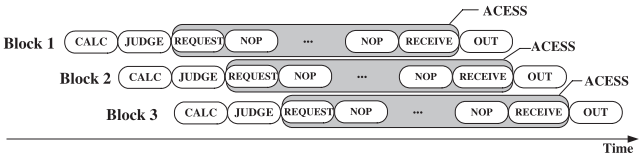


Fig. 6 Block-level pipeline in 2-D cache.

- JUDGE: Judge hit or miss of AUs inside the reference block;
- ACCESS: Access external memory for missed AUs, and store them in local storage space;
- OUT: Output reference block for interpolation.

Since external memory access suffers from long latency, the ACCESS step will take more cycles than the others. We divide it into several sub-steps: REQUEST, RECEIVE and NOPs. REQUEST launches request to external memory; RECEIVE receives the data read from external memory; NOPs is to wait response of external memory. After the division, a block-level pipeline can be constructed as shown in Fig. 6. Pipeline stages are CALC, JUDGE, REQUEST, NOPs, RECEIVE and OUT. Using this pipeline, subsequent basic blocks can continuously be processed during the memory latency. As a result, the waiting time can be used and multiple reading requests can be sent to external memory. To coordinate with our cache, an advanced schedule strategy is used in Memory Controller to enhance the external memory efficiency, which is part of our previous work [11]. With this Memory Controller, multiple access requests from cache can increase the memory access efficiency. Ideally, if there are proper amount of NOPs in the pipeline and no conflict happens, the pipeline will keep running and latency of external memory can be perfectly concealed.

Conflict happens when an AU of current basic block is missed and loading this AU to cache will flush AU needed by unfinished previous basic block or blocks. To solve the conflict, basic block in JUDGE stage of the pipeline must stall until all previous basic blocks involved in the conflict finish OUT stage of the pipeline.

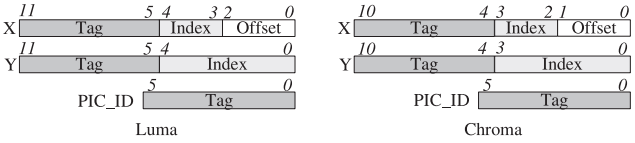


Fig. 7 Division of AU address.

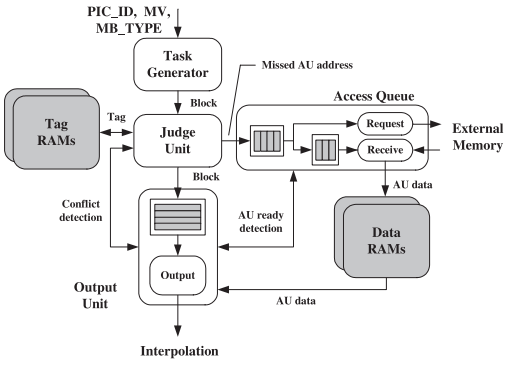


Fig. 8 2-D cache architecture.

Compared with the prefetch strategy used in [6], our block-pipelining strategy has the following advantages:

- Unlike previously used prefetch technique for latency concealment, no prediction is introduced in block-pipelining strategy. For prefetch, when prediction fails, many cycles must be spent to read the right data; the wrongly-fetched data will also increase the bandwidth of MC.
- Prefetch fails if two neighboring blocks have different MV or reference picture indices, which often happens at boundaries of partitions. This will make MC hardware wait for correct data, reducing the MC efficiency. However, our block-pipelining strategy does not suffer from this problem.

3.2 Address Mapping

In our proposed cache scheme, mapping process is as follows: (i) External memory address is represented with: X (Horizontal Position), Y (Vertical Position) and PIC_ID; (ii) AU address is divided into tag, index and offset as shown in Fig. 7; (iii) Each AU of external memory is directly mapped into a cache line whose address in cache is combination of index in X and Y. Tag used to judge hit or miss contains part of X, Y and PIC_ID, as shown in Fig. 7. Another valid bit is used in tag to indicate whether cache line contains valid data.

3.2.1 2-D Cache Architecture

The block diagram of the proposed hardware architecture is shown in Fig. 8.

(1) Task Generator

Task Generator implements the CALC stage of the pipeline, location and size of reference picture block is calculated in this unit. Size of reference block is determined by standard mode, MV fractional part and basic block type. Take 4×8 basic block (luma) as a example, for H.264, its reference block is from 4×8 to 9×13; for AVS, its reference block is from 4×8 to 8×12. We define a basic block as a task inside cache.

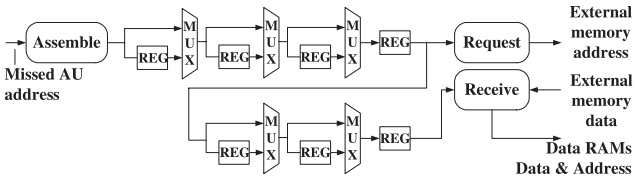


Fig. 9 AU-Block queue.

(2) Judge Unit

Judge Unit implements the JUDGE stage of the pipeline. It receives task from Task Generator and check availability of the AUs needed in the task. To increase the speed of judgment, two vertical adjacent AUs are judged together. The tags from tag RAMs are compared with the target tags to judge whether the two AUs hit or not.

(3) Access Queue

Access Queue implements the REQUEST, RECEIVE and part of NOP stages of pipeline. It first assembles vertically successive AUs received from Judge Unit to an AU-Block, and then launches requests to external memory for the AU-Block. Data received from external memory are written to Data RAMs. An AU-Block Queue is maintained in Access Queue to store the AU-Blocks of tasks in NOP stages of the pipeline. Figure 9 shows the structure of the Access Queue.

(4) Output Unit

Output Unit implements the OUT stage and part of NOP stages of the pipeline. A task-level queue is maintained in Output Unit to store tasks in NOP stages. Output Unit will check whether all the AUs needed by a task are ready in data RAMs. If all the AUs are ready, Output Unit reads AUs of current task and outputs them for interpolation. Otherwise, it has to wait for the AUs. To detect whether AUs are ready, we add a Task ID to each of AU-Block in Access Queue. Task ID contains a task number, a luma/chroma flag and a forward/backward flag. When a task is processed by Output Unit, its ID is compared with IDs of all the AU-Blocks in Access Queue. If one AU-Block contains ID of current task, some requests of AUs for current task must be pending in Access Queue. Otherwise, all the AUs are ready for current task.

(5) Tag RAMs & Data RAMs

Tag RAMs contain two-port RAMs to store tag information. Two RAMs are used to support simultaneous judgment of two vertically adjacent AUs. Tag of AU with even Y is stored in the first Tag RAM; tag of AU with odd Y is

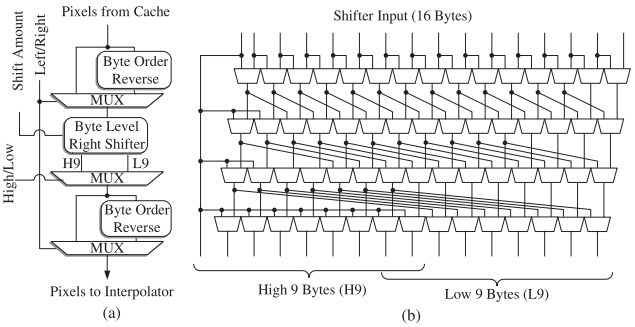


Fig. 10 (a) Pel Shifter (b) Byte-level right shifter in Pel Shifter.

stored in the second Tag RAM. Data RAMs contains two-port RAMs to store AU data. Two RAMs are used to support simultaneous reading of data of two horizontally adjacent AUs. Data of AU with even X is stored in the first Data RAM; data of AU with odd X is store in the second Data RAM.

3.3 Pel Shifter Module Design

Pel Shifter is used to shift pixels in AU properly for interpolation. In addition, when the reference block exceeds picture boundaries, it also replicates boundary pixels to pad exceeded region. We employ a Pel Shifter architecture shown in Fig. 10(a). Since reference block is transferred row by row and the largest reference block width is 9, each row spans at most two horizontal adjacent AUs. Thus, the input data width to Pel Shifter is 128 bits (2 AUs). Utilizing the symmetric property of left and right shift, we can use a byte level right shifter, together with byte order reversal before and after shift to perform both left and right shift. Figure 10(b) shows hardware of right shifter. Interpolator needs at most 9 bytes each row, which may be either high 9 bytes or low 9 bytes of the shifting results.

3.4 Interpolator Module Design

MPEG-1/2, AVS and H.264 all employ fractional sample interpolation to enhance inter prediction accuracy. In luma, MPEG-1/2 only uses half samples, while H.264 and AVS use half and quarter samples. For half samples, MPEG-1/2 uses a bilinear filter; H.264 uses a 6-tap filter (1, −5, 20, 20, −5, 1); AVS uses a 4-tap filter (−1, 5, 5, −1). For quarter samples, H.264 uses a bilinear filter; AVS uses 4-tap filter (1, 7, 7, 1). In chroma, MPEG-1/2 uses a bilinear filter to generate half samples the same as in its luma. H.264 and AVS use the same algorithm in Eq. (2), where A, B, C and D are top-left, top-right, bottom-left and bottom-right integer samples. dx and dy are fractional parts of MVX and MVY respectively, and both are 3 bits.

$$S = ((8-dx)(8-dy)\times A + dx(8-dy)\times B + (8-dx)dy\times C + dxdy\times D + 32) \gg 6 \quad (2)$$

We can find that MPEG-1/2 luma and chroma interpolation

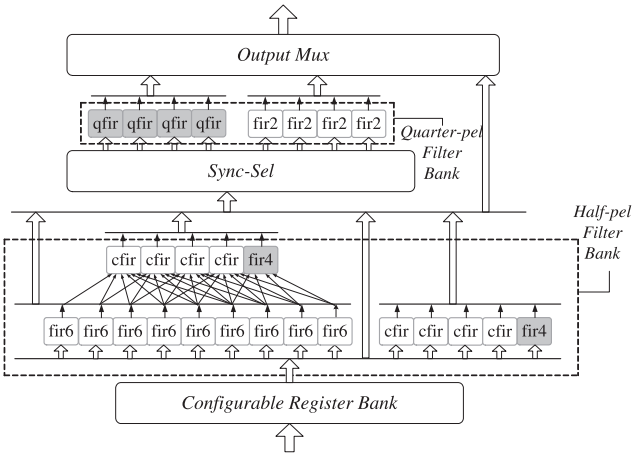


Fig. 11 Proposed H.264 and AVS dual standard interpolator.

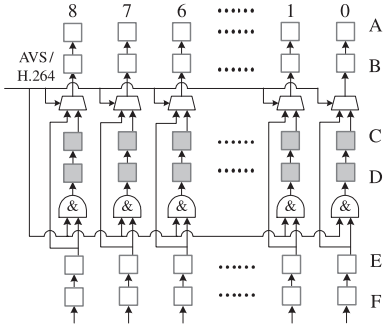


Fig. 12 Configurable Register Bank (CRB).

can be mapped into Eq. (2) by setting the lower 2 bits of dx and dy to 0. Since Eq. (2) is simple to implement and many previous designs are available [7], we will focus on luma interpolator design supporting H.264 and AVS in this paper.

Figure 11 shows the architecture of our proposed H.264 and AVS dual standard luma interpolator. At first, the reference samples needed for interpolation are transferred into Configurable Register Bank (CRB) row by row. CRB delays these samples for 4 or 6 cycles for generating vertical half samples in AVS (4-tap FIR) or H.264 (6-tap FIR), respectively. Then, Half-pel Filter Bank is responsible for generating all the half samples. Sync-Sel Module is used to select and synchronize the half samples needed by quarter-pel filtering. Quarter-pel Filter Bank generates the quarter samples if needed. Finally, the target fractional samples are selected by Output Mux. As the proposed architecture is fully pipelined, the cycle cost equals to the time consumed for reference samples transferring. Take H.264 4x8 basic block as an example, the number of cycles is only 13 in worst case and 8 in best case.

CRB is a $6row\times 9column$ 2-D array of sample registers, as shown in Fig. 12. Each clock cycle, a row of reference samples are shifted into CRB via row F. In H.264 mode, samples are shifted continuously through the 6 rows. In AVS mode, samples shifted out of row E are directly stored into row B, while row C and row D are filled with 0.

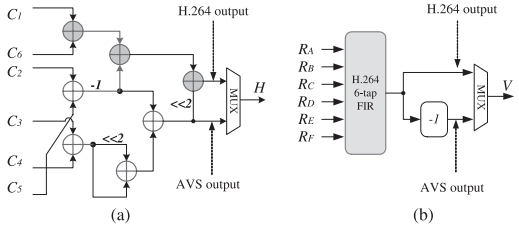


Fig. 13 (a) Architecture of cfir. (b) Architecture of fir6.

Although the filters employed by H.264 and AVS are different, there is correlation between them, which can be used in hardware design. There are five types of filters that we use: cfir, fir6, fir4. qfir and fir2. Cfir and fir6 are reconfigurable and shared by H.264 and AVS to save hardware. Cfir is for horizontal filtering. As illustrated in Fig. 13(a), cfir is composed of an improved adder tree and a MUX for mode selection. In H.264 mode, all inputs of cfir (C_1 to C_6) are active and $H = C_1 - 5C_2 + 20C_3 + 20C_4 - 5C_5 + C_6$. Inputs for these filters are from the 9 samples in row B of CRB. In AVS mode, only 4 inputs are active (except for C_1 and C_6) and $H = 5C_3 + 5C_4 - C_2 - C_5$. This structure uses 7 adders, less than the number of adders used in dedicated realizations for H.264 (6 adders) [13] and AVS (4 adders) [14]. Fir6 is for vertical filtering. As illustrated in Fig. 13(b), fir6 consists of a normal H.264 6-tap filter, a complementer and a MUX for mode selection. Inputs of each fir6 filter are from a corresponding column of CRB. In H.264 mode, $V = R_A - 5R_B + 20R_C + 20R_D - 5R_E + R_F$. In AVS mode, inputs C and D are fixed to zero and the complementer is activated, so $V = 5R_B + 5R_E - R_A - R_F$. This structure uses 6 adders, less than the number of adders used in dedicated realizations. Fir4, qfir and fir2 are dedicated filters. fir4 is a 4-tap filter (−1, 5, 5, −1) for AVS half samples; Qfir is a 4-tap filter (1, 7, 7, 1) for AVS quarter samples; fir2 is bilinear filter for H.264 quarter samples.

3.5 Weighted Predictor Module Design

Weighted predictor is used for weighted prediction (WP). In MPEG-1/2, pixel combination in prime mode and bi-direction mode is equal to WP with weight fixed to 1. To utilize the similarities among 3 standards, we first use a unified formula Eq. (3) to represent their WP algorithms.

$$P = Clip1(((AS(P_0 * w_0) + AS(P_1 * w_1) + 2^{n-1}) \gg n) + o) \quad (3)$$

In Eq. (3), AS means Scale for AVS standard. For H.264 and MPEG-1/2, AS bypasses its input; for AVS, the function of AS is shown in Eq. (4). A_o in Eq. (4) represents luma_offset or chroma_offset in AVS.

$$AS(X) = Clip1((X + 16) \gg 5 + A_o); \quad (4)$$

We propose a WP data path shown in Fig. 14, where multipliers, adders, clips, registers and buffers are shared among 3 standards. For each basic block, w_0, w_1 and o are obtained from a lookup table called Weight Table. In bi-direction prediction mode, forward data come first, which are weighted using w_0 and stored in Bi-Direction Prediction Buffer (BDPB). Then, backward prediction data come, which are weighted using w_1 and added with previously stored forward weighted results. Finally, after post-processing (rounding, offsetting and clipping), we get the final WP results. In single direction prediction mode, weighted data go directly to post processing without entering BDPB. Since our data path is fully pipelined, the cycle cost equals to the time consumed for data transferring.

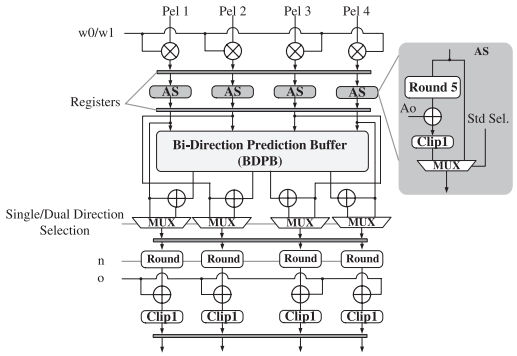


Fig. 14 Implementation of multi-standard weighted predictor.

4. Experimental Result and Analysis

We design our proposed MC architecture using Verilog-HDL. In experimental environment construction, we use 32-bit DDR RAM and SDRAM Controller described in [11] as external memory. H.264 test streams are listed in Table 1; MPEG-1/2 and AVS test streams are encoded with similar parameters and same YUV files as in Table 1. The whole decoder implementation which can decode 1920x1088@30 fps stream at a higher frequency (100 MHz) has also been published as our previous work [15].

To evaluate bandwidth reduction ratio (R_c), we test R_c of hardware using streams in Table 1. The average R_c of our 2-D cache is 74%, better than split-index cache and circular cache in [6]. Moreover, since the cache mapping scheme we used is less complex than [6], it will lead to lower hardware implementation complexity.

Experimental results show that our MC architecture can real-time decode 1920x1088@30 fps video streams at 80 MHz. This is much better than previous works [3]–[5], as shown in Table 2. Performance advantage of our MC design comes first from high throughput of our interpolator and weighted predictor. It also comes from successful concealment of latency using our block-pipelining strategy. In our experimental environment, average latency of external memory is 12 cycles. Our 2-D cache can offer data to interpolator with almost no stall.

Bandwidth reduction ratio of our MC design is 74% in average, which is better than 60%, 30% and 38% in previous design [3]–[5]. This is due to the 2-D cache and the advantageous parameters for cache that we select after study of

Table 2 Comparison of experimental results.

		[3]	[4]	[5]	This Work
Support Standards		H.264	AVS	H.264,MPEG-1/2,AVS	H.264,MPEG-1/2,AVS
Performance		1920x1080 30 fps @115 MHz	1920x1088 30 fps @108 MHz	1920x1088 30 fps (or 60field/s)@148.5 MHz	1920x1088 30 fps@80 MHz 1920x1088 60 fps@160 MHz
Bandwidth Reduction		60%	30%	38%	74%
Technology		0.18 μm CMOS	0.18 μm CMOS	0.18 μm CMOS	TSMC 0.13 μm
Gate Count	Cache	N/A	N/A	9k	11.0k
	Shifter	N/A	N/A	N/A	4.9k
	Interpolator	N/A	N/A	31k	25.5k
	Weighted Predictor	Not support	Not support	Not support	10.3k
	Total	38.8k	44.3k (Without register file)	56k (With MVP)	41.4k (Without WP) 51.7k (With WP)
On-Chip Memory		N/A	N/A	4 kbytes	4 kbytes

parameter design.

We synthesize our MC design by Design Compiler v2007 using TSMC 0.13 technology with 5ns clock constraint. Without Weighted Predictor, gate count of our design is 41.4k; with Weighted Predictor, it is 51.7k. Compared with previous H.264 MC design [3], our MC architecture can support 3 standards with only 6.7% increase in gate count. Because we fully utilizes similarities among standards, our MC design also shows advantage in gate count compared with previous multi-standard MC design [5].

5. Conclusion

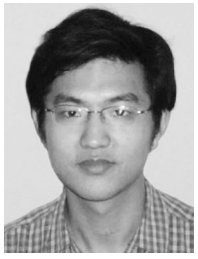
In this paper, we propose a high performance and low bandwidth MC design, which supports H.264/AVC, MPEG-1/2 and AVS standards. By study, we find the suitable 2-D cache parameters for hardware implementation. Using these parameters, we design a high performance 2-D cache. We also propose a block-pipelining strategy for latency concealment. These efforts, together with our high-throughput Interpellator and Weighted Predictor design, make our MC design capable of real-time decoding of 1920x1088@30 fps video streams at 80 MHz. In addition, aggressive hardware sharing in our MC design among 3 standards greatly saves the hardware cost compared with separate implementations.

References

[1] JVT, "Draft ITU-T recommendation and fianl draft International standard of joint video specification," ITU-T Rec.H.264 and ISO/IEC 1449610 AVC, May 2003.
[2] AVS Workgroup, "China Audio and Video Standard (AVS), Information technology-advanced coding of audio and video part2: Video," 2006.
[3] C. Lee and Y. Yu, "Design of a motion compensation unit for H.264 decoder using 2-dimensional circular register files," International SoC Design Conf., pp.II-109–II-112, 2008.
[4] K. Luo, D. Li, and M. Zhang, "High throughput bandwidth optimized VLSI design for motion compensation in AVS HDTV decoder," Journal of Zhejiang University-Science A, vol.9, no.6, pp.822–832, June 2008.
[5] J. Zheng, W. Gao, D. Wu, and D. Xie, "A novel VLSI architecture of motion compensation for multiple standards," IEEE Trans. Consum. Electron., vol.54, no.2, pp.687–694, May 2008.
[6] J.H. Kim, G.H. Hyun, and H.J. Lee, "Cache organization for H.264/AVC motion compensation," IEEE International Conf. on Embedded and Real-Time Computing Systems and Applications, pp.534–541, Aug. 2007.
[7] S.-Z. Wang, T.-A. Lin, T.-M. Liu, and C.-Y. Lee, "A new motion compensation design for H.264/AVC decoder," IEEE International Symposium on Circuits and Systems, vol.5, pp.4558–4561, May 2005.
[8] B. Zatt, A. Azevedo, L. Agostini, A. Sunsin, and S. Bampi, "Memory hierarchy targeting Bi-predictive motion compensation for H.264/AVC decoder," IEEE Computer Society Annual Symp. on VLSI, pp.445–446, March 2007.
[9] R. Wang, M. Li, J. Li, and Y. Zhang, "High throughput and low memory access sub-pixel interpolation architecture for H.264/AVC HDTV decoder," IEEE Trans. Consum. Electron., vol.51, no.3, pp.1006–1013, Aug. 2005.
[10] J.L. Hennessy and D.A. Patterson, "Computer architecture: A quantative approach, 3rd ed. morgan kaufmann," pp.424–425, 2002.
[11] J. Zhu, P. Liu, and D. Zhou, "An SDRAM controller optimized for high definition video coding application," IEEE International Symp. on Circuits and Systems, pp.3518–3521, 2008.
[12] Joint Video Team, H.264/AVC Reference Software JM9.3, http://iphome.hhi.de/suehring/tml/download/old_jm/
[13] K. Denolf, C. Blanch, G. Lafruit, and J. Bormans, "Initial memory complexity analysis of the AVC codec," IEEE Workshop on Signal Processing Systems, pp.222–227, 2002.
[14] C. Yang, S. Goto, and T. Ikenaga, "High performance VLSI architecture of fractional motion estimation in H.264 for HDTV," IEEE International Symp. on Circuits and Systems, pp.2605–2608, 2006.
[15] D. Zhou and Z. You, et al., "A 1080p@60 fps multi-standard video decoder chip designed for power and cost efficiency in a system perspective," 2009 Symp. on VLSI Circuits, pp.262–263, 2009.

Xianmin Chen received his B.E. degree from Shanghai Jiao Tong University, China, in 2007. Now, he is a master student in Department of Electronic Engineering, Shanghai Jiao Tong University. He has been a member of MediaSoC Lab of Shanghai Jiao Tong University since 2006. His main research interests are Multimedia VLSI design and DSP processor design.

Peilin Liu was a PhD Graduate (1992–1998) from the University of Tokyo majoring in Electronic Engineering and worked there as a Researcher in 1999. After that, she was hired as a Senior Researcher for Central Research Institute of Fujitsu, Tokyo (1999–2003). Her research mainly focuses on Multimedia (Audio/Video) Processing, IC Design and High-performance Processor Development. She is now a professor of Department of Electronic Engineering in Shanghai Jiao Tong University, Chief of MediaSoC Lab and responsible for a series of important projects, such as HDTV SoC Platform Development, High-Performance Portable Audio/Video System, etc.



Dajiang Zhou received B.E. and M.E. degrees from the Department of Electronic Engineering of Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. Since Apr. 2008, he has been working toward a Ph.D. degree at the Graduate School of Information, Production and Systems of Waseda University, Japan. Since April 2009, he has been a research fellow of Japan Society of the Promotion of Science (JSPS). Currently, his main interest is in algorithms and VLSI architectures for multimedia and communication technologies.

Jiayi Zhu received B.E. and M.E. degrees from Shanghai Jiao Tong University, at the Department of Electronic Engineering in 2006, and the School of Microelectronics in 2009, respectively. Currently he is working toward a Ph.D. degree at the Department of Electronic Engineering of Shanghai Jiao Tong University. His interest is in VLSI implementations for video coding technologies.

Xingguang Pan received his B.E. degree from Shanghai Jiao Tong University, China, in 2008. Now, he is a master student in Department of Electronic Engineering, Shanghai Jiao Tong University. His main research interest is Multimedia VLSI design.

Satoshi Goto received B.E. and M.E. degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970, respectively. He received the Doctor of Engineering degree from the same university in 1981. He joined NEC Laboratories in 1970 where he worked on LSI design, multimedia systems and software as GM and VP. Since 2003, he has been a professor at the Graduate School of Information, Production and Systems of Waseda University. Currently, his main interest is in VLSI design methodologies for multimedia and mobile applications. He has published 7 books, 38 journal papers and 67 international conference papers. He served as GC of ICCAD and ASPDAC and was a board member of IEEE CAS society. He is IEEE Fellow and Member of Academy Engineering Society of Japan.