Architecture of Global Motion Compensation for MPEG-4 Advanced Simple Profile

Yi-Hau Chen, Ching-Yeh 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 : {yhchen, cychen, lgchen}@video.ee.ntu.edu.tw

Abstract— **Global motion compensation (GMC) is an important coding tool in MPEG-4 Advanced Simple Profile (ASP). In this paper, we propose an efficient GMC hardware architecture for MPEG-4 ASP@L5. Based on analysis of affine model, the proposed memory arrangement and cascaded scheduling reduce the impact of irregular memory access and improve processing ability. It can process 30 fps at only 25 MHz. The implementation result shows that total gate count is 19.3K and internal memory size is 1.28Kb. It is suitable to be integrated into MPEG-4 ASP encoders and decoders.**

I. Introduction

Among video coding standards, MPEG-4 [1] aims at providing efficient storage and transmission in multimedia applications. MPEG-4 Advanced Simple Profile (ASP) is a powerful video coding. Compared to MPEG-4 Simple Profile (SP), MPEG-4 ASP saves 50% bit-rate. The coding gain is derived from several advanced motion compensation tools adopted by MPEG-4 ASP, such as quarter-pixel motion estimation and compenstion (QME/QMC), and global motion estimation and compensation (GME/GMC). In MPEG-4 ASP, QME/QMC targets at compensating the individual object motion, and GME/GMC focuses on compensating the camera motion.

Since GMC became an important coding tool, some global motion models have been proposed. Global motion models are used to describe the camera motion. Panning, zooming, and rotation are three major components of camera motion. MPEG-4 ASP supports five global motion models, and each global motion model can be defined as the motion trajectories of some reference points on the consideration of encoding the transformations. The type of global motion model depends on the number of reference points received. The number of reference points is up to four in MPEG-4 ASP. In this case, a perspective model can be defined as :

$$x' = (m_0x + m_1y + m_2)/(m_6x + m_7y + 1) \qquad (1)$$

$$y' = (m_3x + m_4y + m_5)/(m_6x + m_7y + 1) \qquad (2)$$

where (x,y) is the coordinate of a pixel in the current frame, (x',y') is the coordinate of the corresponding pixel in the reference frame, ($m_0m_1m_2m_3m_4m_5m_6m_7$) are global motion parameters, m_0 and m_4 are scaling factors, m_1 and m_3 are rotation factors, m_2, m_5 are translation factors, and m_6, m_7 are tilt factors. The major operations of GMC are transferring these parameters and computing the transformation.

Compared to traditional local motion compensation (LMC), GMC has an obvious difference in its irregular memory access because GMC supports the deformation of scaling, rotation, and tilt. Irregular memory access means that the needed data are not always in raster scan order, so direct implementation is not efficient enough. It results in low hardware processing efficiency. In this paper, we propose an efficient hardware architecture and scheduling to achieve high processing ability, and reduce 60.94% external memory bandwidth by use of local memory.

The structure of this paper is as follows. In Section II, we introduce the global motion model and analyze the computation of affine model. Next, a hardware architecture and scheduling of GMC are proposed in Section III. The simulation and implementation results are shown in Section IV. Finally, a conclusion is given in Section V.

II. Global Motion Compensation in MPEG-4 VM

In this section, GMC in MPEG-4 Verification Model (VM)[2] would be introduced, and the computation of affine model is analyzed for further hardware implementation.

A. Global Motion Models for GMC

As mentioned in Section I, MPEG-4 ASP defines five kinds of motion models. In these models, stationary and translational models are simpler. They can only represent still video or translation of camera like LMC. Isotropic and affine models are more complex, and they both can support scaling and rotation. Perspective model further deals with camera tilt. However, it is too sensitive to provide a stable performance. Among these models, affine model is usually used. According to our previous work [3], the coding performance of affine model is quite similar to that of perspective model, and affine model is much simpler than perspective model. Affine model can be defined as follow,

$$x' = m_0x + m_1y + m_2, \qquad (3)$$

$$y' = m_3x + m_4y + m_5. \qquad (4)$$

In this paper, our GMC hardware implementation targets on processing four global motion models including stationary, translation, isotropic, and affine models.

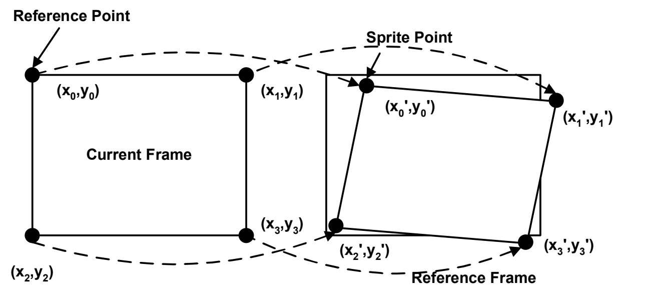

Fig. 1. In MPEG-4 ASP, the sprite points are transmitted to represent GMC parameters.

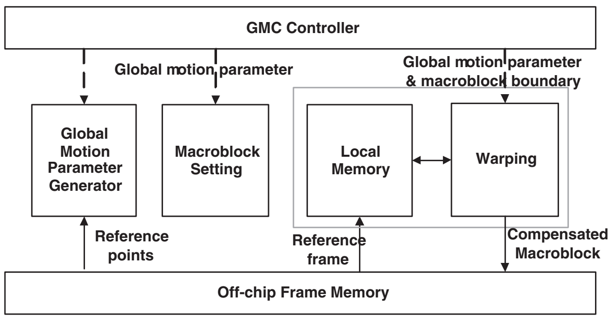

Fig. 2. Overview of the proposed GMC hardware architecture.

B. Analysis of Affine Model

In MPEG-4 ASP, for GMC, the sprite points corresponding to reference points are transmitted instead of the coefficients of global motion parameters. These reference points are some predefined positions of a frame in half-pixel accuracy. In this paper, four corners of a frame are chosen as reference points, as shown in Fig. 1. Since affine model needs three reference points to reconstruct the global motion parameters, it needs six equations to calculate the affine parameters.

In [2], the virtual reference points and virtual sprite points are used to simplify the transform functions. The warping process can be expressed as follows :

$$i'(x,y) = (m_0H'x + m_1W'y)/(W'H') + m_2, \qquad (5)$$

$$j'(x,y) = (m_3H'x + m_4W'y)/(W'H') + m_5, \qquad (6)$$

$$m_0 = (W'(x_1' - x_0'))/W, \qquad (7)$$

$$m_1 = (W'(x_2' - x_0'))/W, \qquad (8)$$

$$m_2 = x_0', \qquad (9)$$

$$m_3 = (H'(y_1' - y_0'))/H, \qquad (10)$$

$$m_4 = (H'(y_2' - y_0'))/H, \qquad (11)$$

$$m_5 = y_0', \qquad (12)$$

where the operator "/" indicates division with hexadecimal accuracy, W and H are the width and height of video sequences , and $W' = 2^\alpha$, $H' = 2^\beta$, $W' \geq W$, $H' \geq H$, $\alpha > 0$, $\beta > 0$, both α and β are integers and are chosen as 10 in MPEG-4 VM for MPEG-4 ASP@L5. (x_0', y_0'), (x_1', y_1'), and (x_2', y_2') are the received locations of sprite points as shown in Fig. 1. Since W' and H' are power of two, the divisors in (5) and (6) can be replaced by binary shift. To further reduce the complexity of warping, the scanline algorithm in [4] [5] is adopted. Then, the multiplication can be replaced with addition by the following equations.

$$i'(0,0) = m_2 \qquad (13)$$

$$i'(x+1,y) = i(x,y) + m_0/W' \qquad (14)$$

$$i'(x,y+1) = i(x,y) + m_1/H' \qquad (15)$$

Since GMC in MPEG-4 ASP is macroblock-based process, some blocks are skipped in GMC if they are not coded by GMC mode. Even the above (14) and (15) are applied, (5) and (6) still need to execute twice in each macroblock for luminance and chrominance blocks. Therefore, the operation of affine transformation in one 16×16 luma block is reduced from 1024 multiplications and 1024 additions to only 4 multiplications and 514 additions.

III. Proposed GMC Hardware Architecture

In this section, based on the GMC algorithm in MPEG-4 VM and analysis in Section II-B, a hardware architecture for GMC in MPEG-4 ASP@L5 is proposed. As shown in Fig. 2, there are four major components, *GMC Controller*, *Global Motion Parameter Generator*, *Macroblock Setting*, and *Warping* with *Local Memory*. An off-chip frame memory is required in this architecture to store the reference frame and the output data, a compensated macroblock. GMC controller controls other modules and assigns work of each module. The detailed architectures of these modules are described in the following subsections.

A. Global Motion Parameter Generator

According to the equations in Section II-B, a global motion parameter generator is designed. Since one frame only has one set of global motion parameters, *Global Motion Parameter Generator* is executed once in each frame. Therefore, one processing element to calculate global motion parameters is enough, and this processing element can handle four motion models except perspective model. However, based on the analysis of Section II-B, a division to generate m_0, m_1, m_3, and m_4, is needed. Since W and H are constants, multipliers can be used instead of dividers. Furthermore, these multipliers are shared with *Macroblock Setting* and *Warping* to reduce overall overhead.

B. Macroblock Setting

Figure 3(a) shows a prototyping pixel distribution of LMC, and Fig. 3(b) shows the pixel distribution of GMC which leads to irregular memory access. If the accessed region can be obtained before loading into *Local Memory*, we only need to load these reference data once regardless of the deformation of block. In *Macroblock Setting*, the processing element generates the left-top corner pixel's location and then calculates the deformation block's boundaries. According to definitions of

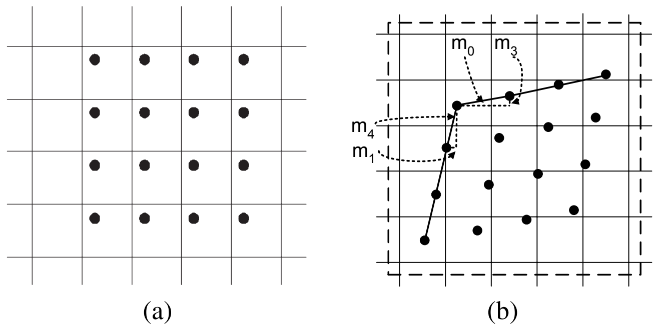

Fig. 3. (a)The corresponding region in reference frame for LMC. (b)The corresponding region in reference frame for GMC. The black points here mean the locations of the pixels for compensation. The region surrounded by dashed line is the loaded into Local Memory in proposed architecture.

global motion parameters, the processing element can easily determine which pixel is on the specific boundary. For example, if $m_3 < 0$, we can choose the right-top corner point to decide upper boundary without considering left-top corner point as shown in Fig. 3(b); otherwise, the left-top corner pixel is chosen.

C. Warping

The detailed hardware architecture of *Warping* is shown in Fig. 4. *Warping Controller* arranges the scheduling of reading reference frame from external memory and accessing neighboring pixels from local memory. *External Address Generator* generates the addresses of reference pixels of the block, which corresponds to the current macroblock, in raster scan order, and *Memory Location Decision* arranges the reference data in local memory as described in Section III-D later. *Warping Address Generator* generates the corresponding positions of current pixels in the reference frame. Because of the scaling, rotation, and translation factors, the corresponding pixel is usually not located in integer grid. Therefore, the luminance and chrominance of the reference pixels are interpolated by bi-linear *Interpolation Filter* .

Figure 5 shows the block diagram of the scanline-based address elements in *Warping Address Generator*. Since the location of the left-top corner point has already been calculated in *Macroblock Setting*, the address elements in *Warping* are multiplication-free, and the resource of multiplier can be directly used by *Interpolation Filter*.

D. Local Memory

In GMC algorithm, the irregular memory access increases the difficulty of accessing reference data from external frame memory. On the other hand, the interpolation of GMC needs four neighboring pixels and results in high external memory bandwidth. To solve the above difficulties, an on-chip *Local Memory*, which stores reference frame data for data-reuse, is always required [3] [6]. Since the interpolation of GMC needs four neighboring pixels, the interleaved arrangement of reference data in *Local Memory* is adopted [3]. In this way, four neighboring pixels can be accessed for interpolation in

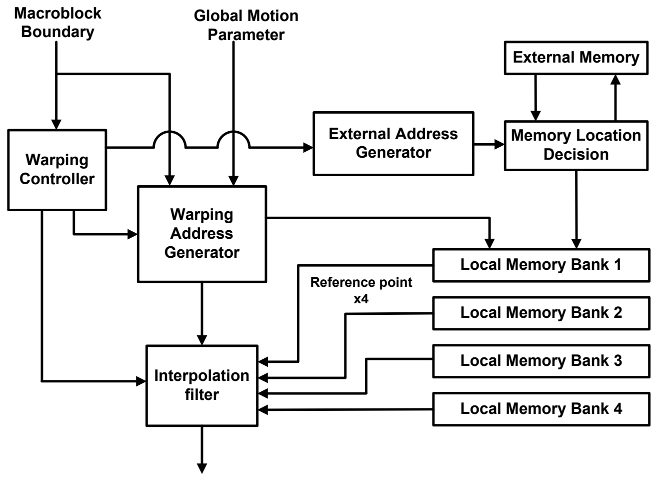

Fig. 4. Hardware architecture of Warping and Local Memory.

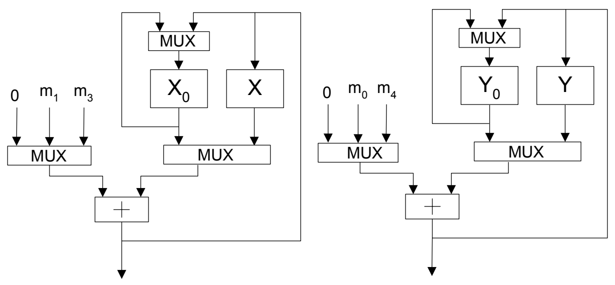

Fig. 5. Block diagram of the scanline-based x-coordinate address PE. m_0, m_1, m_3, and m_4 are affine parameters. X and Y are registers to save the locations of current corresponding pixels, and X_0 and Y_0 save the locations of first corresponding pixel in current processing row.

one cycle. The whole flow of accessing memory data is as follows. The boundaries of reference data have been calculated in *Macroblock Setting*, so we can load reference data into local memory row by row. Furthermore, all four local memory banks are separated into two parts. The first four row data are saved in first part of four local memory and then the later four row data are saved in second part of four memory. Hence, at most eight row of data are saved in local memory at same time. As soon as the loaded reference data in local memory is enough for *Interpolation Filter* to access, the *Warping Address Generator* begins to produce the location of the pixel corresponding to current pixel in reference frame. The scan order of current macroblock's pixels is raster scan.

However, global motion compensation needs not only to generate predicted luminance block but also to produce predicted two chrominance blocks. For example, after reconstructing luminance block, the warping address generator is stalled until enough reference data of Cb-block are loaded into local memory. But it results in longer cycles for GMC in each macroblock. We propose an improved cascaded scheduling as shown in Fig. 6. Once the reference data of luminance block have all been loaded into local memory, the *warping controller* finds out the reference data that have been out of corresponding region of current row according to scaling and

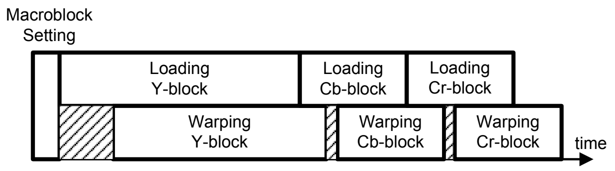

Fig. 6. The cascaded scheduling of loading and warping in processing one macroblock. The shadow region indicates that the warping address generator is stalled.

rotation factors. Then these reference data are discarded, and the reference data of Cb or Cr block are loaded into local memory. It can reduce processing cycles of GMC for one macroblock and make it more suitable to be integrated into MPEG-4 ASP encoders/decoders.

Because loading reference data into *Local Memory* and accessing neighboring pixels are processed in the same time, these actions of reading and writing data in same memory bank may happen. Then two port RAMs are required to implement *Local Memory*. On the other hand, the hardware efficiency depends on the range of supported global motion parameters. If the size of *Local Memory* is too small, the reference data cannot be refreshed before the requirement of warping under some deformation coefficients. Then the hardware utilization will be low and some extreme cases cannot be supported. The relation between each local memory's size S and deformation coefficients can be formulated as $S = (15\times(m_0+m_1)+2)\times 2$. Depend on the above analysis, we decide to use four 40×8 two port RAMs so that our design can support scaling factor up to 1.1 and rotation factors from -0.1 to 0.1. Hence, it can tolerate that video sequences enlarge their frame width or height 15.86 times per second, and such condition covers almost all video sequences.

IV. Simulation Results

A. Proposed Architecture

Figure 7 shows the simulation results of memory bandwidth and processing ability of proposed hardware architecture under 25 MHz. By use of local memory and interleaved memory arrangement, it can access four neighboring pixels in one cycle, the external memory bandwidth can be reduced 60.94%. With cascaded scheduling, the processing ability of proposed architecture can improve about 12%, and it can process about 31 fps at working frequency of 25 MHz.

B. Hardware Implementation

In Fig. 8, the hardware implementation result of proposed GMC architecture is shown. The target specification is MPEG-4 ASP@L5. That is, the frame size of reconstructed frame is 720×576 with 30 fps. The target working frequency is only 25 MHz. The hardware is implemented with Verilog-HDL and synthesized with SYNOPSYS Design Vision. ARTISAN 0.18um cell library is adopted to implement hardware. The total gate cout is about 19.3 K, and the internal memory size is 1.28 Kb. It is quite reasonable to be integrated into MPEG-4 ASP encoders and decoders.

	Memory Bandwidth (MB/30frame)	Processing Ability (frames/sec)
GMC without local memory	74.65	10.05
Proposed GMC architecture without cascaded scheduling (average/worst)	22.32/29.16	27.75
Proposed GMC architecture with cascaded scheduling (average/worst)	22.32/29.16	31.12

Fig. 7. Memory bandwidth of original and proposed GMC architecture and the comparison of processing ability at 25MHz in MPEG-4 ASP@L5. Assume that one byte can be read from external memory at a time.

Module	Gate Count	Internal Memory
GMC Controller	4066	0 b
Global Motion Parameter Generator	2821	0 b
Macroblock Setting	4787	0 b
Warping	7602	1280 b
Total gate count	19276	1280 b

Fig. 8. The result of GMC hardware implementation in MPEG-4 ASP@L5 (frame size 720 × 576, 30 fps.). Four 40 × 8 dual ports RAMs are used in Internal Memory since our ARTISAN 0.18um cell library has no two port RAM.

V. Conclusion

In MPEG-4 ASP, global motion compensation is an important coding tool. But there are few GMC hardware architectures. In this paper, we simplify the computation of GMC and propose an efficient hardware architecture. By use of interleaved memory structure and cascaded scheduling, the external memory bandwidth saved 60.94%, and this architecture can achieve MPEG-4 ASP@L5 at 25 MHz. For MPEG-4 ASP@L5, the gate count is 19.3 K, and the internal memory size is 1.28 Kb. This architecture is suitable to be integrated into MPEG-4 ASP encoders and decoders.

References

[1] MPEG Video Group, *AMENDMENT4: Streaming Video Profile*, ISO/IEC JTC1/SC29/WG11 N3904, 2001.

[2] MPEG Video Group, *The MPEG-4 Video Verification Model version 18.0*, ISO/IEC JTC1/SC29/WG11 N3908, 2001.

[3] C.-Y. Chen, S.-Y. Chien, W.-M. Chao, Y.-W. Huang, and L.-G. Chen, "Hardware architecture for global motion estimation for mpeg-4 advanced simple profile," in *International Symposium on Circuits and Systems*, May 2004, vol. 2, pp. 301–304.

[4] K. Kannappan, "An interleaved scanline algorithm for 2-d affine transformations of images," in *Proceedings of the 35th Midwest Symposium in Circuits and System*, Aug. 1992, vol. 1, pp. 179–182.

[5] M. Sayed and W. Badawy, "A novel memory architecture for real-time mesh-baed video motion compensation," in *International Workshop on System-on-Chip for Real-Time Applications*, July 2004, vol. 1, pp. 153–157.

[6] S.-Y. Chien, C.-Y. Chen, W.-M. Chao, Y.-W. Huang, and L.-G. Chen, "Analysis and hardware architecture for global motion estimation in mpeg-4 advanced simple profile," in *International Symposium on Circuits and Systems*, May 2003, vol. 2, pp. 720–723.