J Sign Process Syst (2008) 53:285–300
DOI 10.1007/s11265-008-0169-7

Analysis and Hardware Architecture Design of Global Motion Estimation

Yi-Hau Chen · Shao-Yi Chien · Ching-Yeh Chen · Yu-Wen Huang · Liang-Gee Chen

Received: 28 March 2006 / Revised: 3 March 2008 / Accepted: 3 March 2008 / Published online: 12 April 2008

Abstract Global motion estimation and compensation (GME/GMC) is an important video processing technique and has been applied to many applications including video segmentation, sprite/mosaic generation, and video coding. In MPEG-4 Advanced Simple Profile (ASP), GME/GMC is adopted to compensate camera motions. Since GME is important, many GME algorithms have been proposed. These algorithms have two common characteristics, huge computation complexity and ultra large memory bandwidth. Hence for realtime applications, a hardware accelerator of GME is required. However, there are many hardware design challenges of GME like irregular memory access and huge memory bandwidth, and only few hardware architectures have been proposed. In this paper, we first analyzed three typical algorithms of GME, and a fast GME algorithm is proposed. By using temporal prediction and skipping the redundant computation, 91% memory bandwidth and 80% iterations are saved, while the performance is kept, compared to Gradient Descent in MPEG-4 Verification Model. Based on our proposed algorithm, a hardware architecture of GME is also presented. A new scheduling, Reference-Based Scheduling, is developed to solve the irregular memory access problem. An interleaved memory arrangement is applied to satisfy the memory access requirement of interpolation. The total gate count of hardware implementation is 131 K with Artisan 0.18 *um* cell library, and the internal memory size is about 7.9 Kb. Its processing ability is MPEG-4 ASP@L3, which is 352×288 with 30 fps, at 30 MHz.

Keywords VLSI architecture · MPEG-4 · Global motion estimation · Global motion

Y.-H. Chen · S.-Y. Chien · C.-Y. Chen
Y.-W. Huang · L.-G. Chen
DSP/IC Design Lab., Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Y.-H. Chen
e-mail: ttchen@video.ee.ntu.edu.tw

C.-Y. Chen
e-mail: cychen@video.ee.ntu.edu.tw

Y.-W. Huang
e-mail: yuwen@video.ee.ntu.edu.tw

L.-G. Chen
e-mail: lgchen@video.ee.ntu.edu.tw

S.-Y. Chien (✉)
Room 540, Department of Electrical Engineering II, National Taiwan University, 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
e-mail: sychien@cc.ee.ntu.edu.tw

1 Introduction

Global motion and local motion are two types of motions in a video sequence. The former is camera motion, and the latter is the motion of individual objects. Global motion estimation (GME) is to find the global motion between two frames and use only a small set of parameters [1], global motion parameters, to describe the motion instead of many local motion vectors, as shown in Fig. 1. Global motion model is a set of global motion parameters with physical meanings and is used to describe the camera motions including scaling, rota-

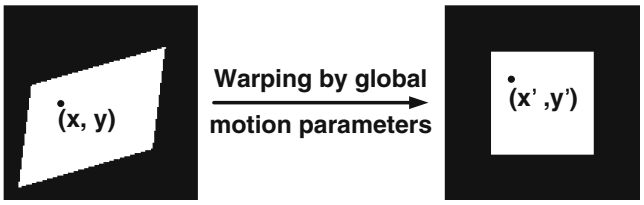

Figure 1 The deformation between two frames with affine model, where $m_0 = 0.8$, $m_1 = 0.1$, $m_2 = 10$, $m_3 = 0.3$, $m_4 = 1.05$, and $m_5 = -30$.

tion, and translation. Many global motion models have been proposed, such as perspective, affine, isotropic, and translation models. For example, the affine model is

$$x' = m_0 x + m_1 y + m_2, \tag{1}$$

$$y' = m_3 x + m_4 y + m_5, \tag{2}$$

where (x, y) is the position of current pixel in current frame, (x', y') is the corresponding position of current pixel in the reference frame, and $(m_0, m_1, m_2, m_3, m_4, m_5)$ are global motion parameters, where m_0 and m_4 are scaling factors, m_1 and m_3 are shear factors, and m_2 and m_5 are translation factors in x and y directions, respectively. Figure 1 shows the deformation between two frames with affine model, where $m_0 = 0.8$, $m_1 = 0.1$, $m_2 = 10$, $m_3 = 0.3$, $m_4 = 1.05$, and $m_5 = -30$.

Global motion estimation and compensation (GME/GMC) plays an important role in many applications. For example, in image stabilizers, GME/GMC is adopted to estimate and compensate the camera motions [2–4]. The consistency between global motion and local motion of a macroblock is used to be the criterion in video segmentation [5–7]. MPEG-7 [8, 9] also has many descriptors about camera motions and global motions. In the applications of video coding, GME/GMC is the core technology of sprite/mosaic coding [10–14], and GME/GMC is also adopted in MPEG-4 Advanced Simple Profile (ASP) [15]. MPEG-4 ASP is undoubtedly a powerful video coding standard. It can save 50% bitrate compared to MPEG-4 Simple Profile, which is because several advanced motion compensation tools are included in MPEG-4 ASP, and one of them is GME/GMC [16–19].

Since GME/GMC becomes an important video processing tool, many GME algorithms [18, 20] have been proposed. They can be classified into three types [21]: *Feature-Point Based Algorithms* [22], *Differential-Technique Algorithms* [10, 23, 24], and *Frame-Matching Algorithms* [25, 26]. In *Feature-Point Based Algorithms*, some feature points are selected to represent the whole frame. By motion vectors of these feature points, global motion parameters are derived. In *Differential-Technique Algorithms*, Taylor series are applied to expand a criterion function, which is used to calculate global motion parameters. On the other hand, in *Frame-Matching Algorithms*, the whole frame is matched with all possible global motion parameters to select the optimal one. The detailed analysis and comparison of these three algorithms will be shown in Section 2.

A hardware accelerator of GME is necessary for real-time applications, such as, MPEG-4 ASP encoders [27]. However, the hardware design of GME is a challenged research topic because of the characteristics of GME. First, there are two common characteristics in GME algorithms. One is huge computation complexity, and the other is ultra large memory bandwidth. For example, Gradient Descent [23, 28] takes 35 GIPS in computation complexity and 340 MB/s in memory bandwidth at CIF Format, 30 fps. These two characteristics increase the complexity of hardware implementation for GME. The second problem is that because scaling and rotation are supported in global motions, irregular memory access of GME is inevitable. It seriously affects the hardware utilization without careful design. Thirdly, four neighboring reference pixels have to be accessed for one current pixel. Then, the memory arrangement needs to be designed well in order to satisfy the memory access requirement. Up to now, there are fewer algorithms suitable for hardware implementation, and fewer hardware architectures [27, 29] are proposed.

In this paper, we first analyze the performances of different GME algorithms and propose a fast GME algorithm which is suitable for hardware implementation in Section 2. Next, the coding performances of GME/GMC mode with different global motion models in MPEG-4 ASP are compared in Section 3. In Section 4, a GME hardware architecture is proposed, and it can solve the above-mentioned problems. Finally, the results of hardware implementation are shown in Section 5, and a conclusion is given in Section 6.

2 GME Algorithms

In this section, we introduce three types of GME algorithms, and their performances are also discussed. Next, based on these analyses, we propose a fast GME algorithm, Avoid Redundant Computation Gradient Descent (ARC-Gradient Descent).

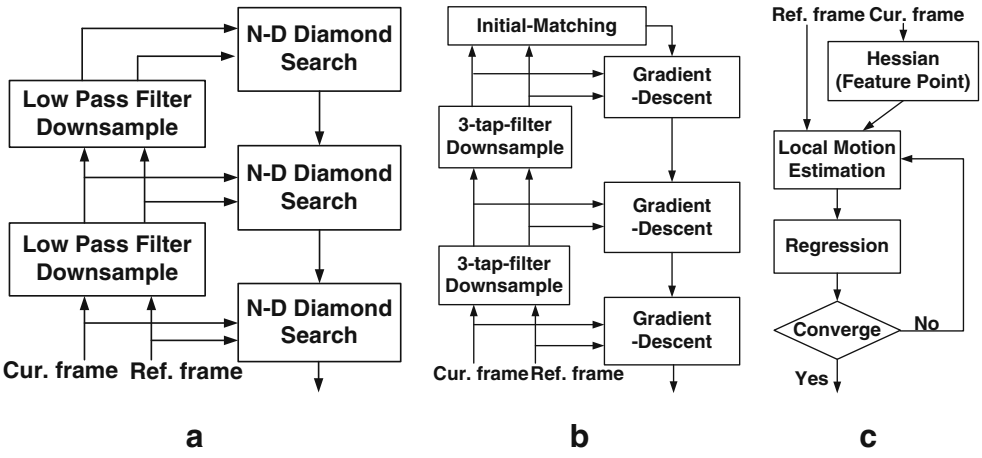

Figure 2 The flow chart of **a** N-D diamond search; **b** Gradient Descent in MPEG-4 VM; **c** Feature-point based algorithm.

2.1 Frame-Matching Algorithms

Frame-Matching Algorithms are similar to traditional block matching motion estimation algorithms. *Frame-Matching Algorithms* calculate the distortion of each possible global motion parameter and select the one which has the smallest distortion as the optimal global motion parameters. However, in GME, the searching space becomes very huge. It is extended from 2-Dimensions in traditional motion estimation to N-Dimensions (N-D) in GME, where N is the number of global motion parameters. Furthermore, in traditional motion estimation, the resolution of searching space is nearly limited, such as integer pixel, half pixel, or quarter pixel, but in GME, the resolution of searching space is almost unlimited because of a high precision requirement. Hence there are much more candidates in GME compared to traditional motion estimation. The computation complexity of this algorithm is increased largely.

N-D diamond search is a typical *Frame-Matching Algorithm* [26]. The flowchart is shown in Fig. 2a. In order to make a fair comparison between different GME algorithms, a hierarchical method which is the same as Gradient Descent in MPEG-4 Verification Model (VM) [23] is adopted. The hierarchial method is implemented with a three pyramid and a low pass filter. There are three levels, original frame size, 1/4 frame size, and 1/16 frame size in this algorithm. The low pass filter is a 3-tap filter, [1/4 1/2 1/4]. After filtering the frame, the frame is subsampled in horizonal and vertical directions. In this algorithm, the affine model, which is shown in Eqs. 1 and 2, is applied as the global motion model. At each level, two step sizes of N-D diamond search are adopted. Because there are six variables in affine model, 3^6 candidates have to be calculated in each iteration. If the sum of absolute differences (SAD) of the whole frame is convergent or the number of iterations is larger than the threshold, the processing goes to the smaller step size. After finishing the computation of two step sizes, the procedure enters to the next level. When the computation of three levels has been finished, the optimal global motion parameters is selected.

2.2 Differential-Technique Algorithms

Gradient Descent is a well-known GME algorithm of *Differential-Technique Algorithms*. The flowchart of GME in MPEG-4 VM [23] is shown in Fig. 2b. There are three major functions, *3-Tap-Filter*, *Initial-Matching*, and *Gradient-Descent* in this algorithm. Gradient Descent is a hierarchical and iterative algorithm. A three-level pyramid and a *3-Tap-Filter* is used for this propose. At the smallest frame size level, the predictive translation vector is generated in the *Initial-Matching*. After that, global motion parameters are estimated by Levenberg-Marquardt iterative minimization algorithm [30] in *Gradient-Descent*. The iterative process does not finish until it converges or achieve the maximum number of iterations at each level.

In *Gradient-Descent*, global motion parameters are estimated by minimizing the mean square error, E,

$$E = \frac{1}{TotPels} \sum_{i \in TotPels} |e(i)|^2, \tag{3}$$

$$e(i) = I(x_i, y_i) - I'\left(x'_i, y'_i\right), \tag{4}$$

where $I(x_i, y_i)$ is the luminance of current pixel (x_i, y_i) in the current frame, $I'(x'_i, y'_i)$ is the luminance of the corresponding position for current pixel (x_i, y_i) in the reference frame, and $TotPels$ is a set of effective pixels,

whose errors are smaller than the error threshold. The error threshold is used to exclude the effects of the movements of foreground objects. By an error histogram, in which the distribution of $|e(i)|$ is computed, the error threshold is set as the value which excludes top 20% of the distribution of $|e(i)|$.

The iterative procedure of *Gradient-Descent* is shown as follows.

$$M_{t+1} = M_t + A^{-1}B, \tag{5}$$

$$M = (m_0 \ m_1 \ .... \ m_{N-2} \ m_{N-1})^T, \tag{6}$$

where M_t are global motion parameters at t-th iteration, A is an $N \times N$ matrix, B is an $N \times 1$ matrix, and N is the number of global motion parameters. The coefficients of the matrix A and B are given by

$$A_{kj} = \sum_{i \in TotPels} \frac{\partial e(i)}{\partial m_k} \frac{\partial e(i)}{\partial m_j}, \tag{7}$$

$$B_k = \sum_{i \in TotPels} -e(i) \frac{\partial e(i)}{\partial m_k}, \tag{8}$$

$$\frac{\partial e(i)}{\partial m_j} = f_j\left(\frac{\partial e(i)}{\partial x}, \ \frac{\partial e(i)}{\partial y}, \ x_i, \ y_i\right). \tag{9}$$

As shown in Fig. 2b, the iteration of *Gradient-Descent* starts after *Initial-Matching* at the top level of the pyramid and repeats at the subsequent levels. At each level, the process of iterations is repeated, until the improvement of each parameter is smaller than a threshold or the number of iterations is larger than the maximum number of iterations, which is set as 32 in MPEG-4 VM.

2.3 Feature-Point Based Algorithms

In *Feature-Point Based Algorithms* [22], some feature points are selected to represent the whole frame. By the motion vectors of these feature points, global motion parameters are derived. Figure 2c shows the flowchart of one kind of these algorithms. There are three major functions in this algorithm, *Hessian*, *Local Motion Estimation*, and *Regression*. In *Hessian*, feature points, Hessian points, are selected by Hessian Value, which is defined as

$$\left[\left(\frac{d^2I(x,y)}{dx^2}\right) \cdot \left(\frac{d^2I(x,y)}{dy^2}\right) - \left(\frac{d^2I(x,y)}{dxdy}\right)^2\right].$$

First, the Hessian Value of each pixel is calculated, and those pixels who have larger Hessian Values are selected as feature points. A pixel which has a large Hessian Value is more different to its neighboring pixels. Because of the slighter aperture problem, it is easy to find the true motion of such a pixel. On the other way, in order to improve the correctness of global motion parameters, these feature points have to be dispersed evenly in the whole frame. Then, the frame is partitioned into four parts, and the Hessian points are averagely chosen in every part.

In *Local Motion Estimation*, the motion vector of each feature point is calculated. Because the feature points are Hessian points, the block matching algorithm in an L-neighborhood (typically L=4) around the Hessian point is applied. First, the corresponding position of Hessian points in the reference frame is predicted by the global motion parameters of the last iteration or the previous frame. Next, a new optimal matching position is searched around the predicted position. Motion vector prediction can reduce the computation complexity and improve the correctness of motion vectors. Finally, global motion parameters are calculated by least mean-square algorithm in *Regression*. This process is repeated iteratively, until the SAD is convergent or the number of iterations is larger than the maximum number of iterations.

2.4 Analysis and Comparison of GME Algorithms

In this section, we discuss the comparisons of three GME algorithms with affine model. Many sequences, such as Foreman, Stefan, Mobile, Table Tennis, and so on, are tested, but due to the limited space in this paper, only the data of Foreman and Stefan are listed. We summarize the performances of three algorithms in Table 1, including the run time, instruction profile, and PSNR, where the PSNR is calculated by comparing the original and reconstructed frames from the uncompressed previous frame and only GME/GMC is used to compensate the motion. The software and simulation platform of instruction profiling are iprof and P4-1.8 GHz with GB memory, where the operating system is Redhat Linux 6.2. The simulation environment of runtime profile is P4-1.8 GHz with 256 memory and Language C is used.

Table 1 The performances of *frame matching*, *gradient descent*, and *feature-point based* algorithms.

Algorithm	Runtime (s/300f)	Instruction (GIPS)	Stefan (dB)	Foreman (dB)
Frame matching	16,837.7	1,800.0	23.99	28.83
Gradient descent	280.1	35.0	24.10	28.73
Feature-point based	84.8	8.0	23.14	27.95

The computation complexity and runtime of *Frame-Matching Algorithms* are 1,800 GIPS and 16,837 s/300 frames at CIF Format with 30 fps, and these are much larger than those of others. The computation complexity and runtime of *Feature-Point Based Algorithms* are the smallest, but it loses 1 dB compared to others. *Gradient Descent* is the algorithm with a better tradeoff between the computation complexity and performance. Moreover, considering the complexity of instructions, the operation of *Frame-Matching Algorithm* is much more regular and simpler than others, but its computation complexity is too large to be implemented. The operation of *Feature-Point Based Algorithms* is the most irregular and complicated, so it is not suitable for hardware design. Consequently, *Gradient Descent* is much suitable for hardware implementation.

2.5 Proposed Avoid Redundant Computation Gradient Descent

Based on the above-mentioned analysis, Gradient Descent is much suitable for hardware implementation. However, in *Gradient-Descent*, the number of iterations is 32 in the worst case at each level. Even in the average case, more than ten iterations are required in each level, especially for Foreman and Stefan, as shown in Table 2. A lot of iterations induce the huge computation complexity and ultra large memory bandwidth. We proposed a fast GME algorithm [31], Avoid Redundant Computation Gradient Descent (ARC-Gradient Descent), to save the number of iterations and further reduce the computation complexity and memory bandwidth.

In general, camera motion is a continuous motion. The global motion parameters of successive two frames are similar, and then the global motion parameters of previous frame can be used as the prediction [32] in order to reduce the number of iterations. However, because of using the previous global motion parameters as the predictor, it is possible that the error propagation occurs, especially when the scene changes. In order to avoid this problem, the prediction scheme is interrupted once every 30 frames, while *Initial-Matching* is executed to find the initial translation parameters as the predictor for *Gradient-Descent*.

When GME/GMC mode is adopted in MPEG-4 ASP, global motion parameters are not the final data which are coded into the bitstream. Sprite points are coded instead of global motion parameters in order to match and preserve the precision of global motion parameters. The coding flow of sprite points is in the following. First, several reference points are selected in the current frame. Second, based on global motion parameters, sprite points, which are the corresponding positions of the reference points, are calculated and coded by DPCM. However, because the precision of sprite points is limited and only half pixel, the constraint on the precision of global motion parameters can be redefined to skip the redundant computation of GME. We proposed a new criterion to determine the convergency of global motion parameters. The threshold is

$$\frac{0.5}{Image\_Width}.$$

By this constraint, the error of sprite points caused by the error of global motion parameters is smaller than half-pixel. After applying the new criterion and temporal prediction, the number of iterations is much less than five in the average case. Considering the worst case, the max number of iterations is set as five. Besides, the error threshold which is used to exclude the foreground pixels is analyzed, and it is less than six in the average case. The number of error histogram bins is also reduced from 256 to 9 bins in order to reduce the hardware cost of error histogram.

The performance of ARC-Gradient Descent is summarized in Tables 2 and 3, where the simulation environment is the same as the previous subsection. The number of iterations in ARC-Gradient Descent is reduced to only 20% of that in the original algorithm. The memory bandwidth is reduced from 356.06 to 46.27 MBytes/s in the worst case and from 217.17 to 19.10 MBytes/s in the average case. Totally, the memory bandwidth is saved 87.0% in the worst case, and the runtime is also speeded up five times compared to the original algorithm. The performance of ARC-Gradient

Table 2 The comparison of the original and the propose algorithms.

Algorithm	Original			Proposed		
Sequence	Stefan	Foreman	Average	Stefan	Foreman	Average
PSNR	24.09	28.61	29.17	24.08	28.74	29.25
The number of iterations						
Original size	13.75	15.42	11.64	2.65	2.53	1.70
1/4 size	11.75	14.33	9.62	2.09	2.01	1.10
1/16 size	12.57	16.82	8.94	3.55	3.04	1.95

Table 3 The bandwidth comparison of the original and propose algorithms.

Function	Original	Proposed	
	Bandwidth	Bandwidth	Reduction (%)
3-tap-filter	3.11	3.11	0.0
Initial-matching	97.52	3.24	96.7
Gradient-descent			
The average case	116.54	12.75	89.1
The worst case	255.43	39.92	84.4
Average case (Mbytes/s)	217.17	19.10	91.2
Worst case (Mbytes/s)	356.06	46.27	87.0

Descent is very close to the original *Gradient Descent* algorithm, and in some sequences, the proposed algorithm has a better performance, as shown in Table 2.

3 The Performance and Analysis of GME/GMC Mode in MPEG-4 ASP

After the discussion of GME algorithms, we analyze the performances and computation complexity of GME/GMC mode with different global motion models in MPEG-4 ASP. Four global motion models, translation, isotropic, affine, and perspective models are supported in MPEG-4 ASP. Translation model is like as traditional motion estimation and only supports translation motion. In isotropic model, there are two more parameters, scaling and shear factors, compared to translation model. Affine model supports different scaling and shear factors in x and y directions. Perspective model is the most complicated motion model among these four global motion models. The specification is CIF Format with 30 fps, and the searching range is [−32, 32) with quarter-pixel motion estimation and compensation (QME/QMC). Moreover, there is no B-frame. Seven sequences, including Stefan, Foreman, Table Tennis, Mobile, and so on, are used to test the performances of GME/GMC with various global motion models. However, because of the limited space, only the performances of Foreman and Table Tennis are shown in the following discussion.

3.1 The Rate-Distortion Curves of GME/GMC Mode

The coding performance of GME/GMC depends on the different features of test sequences. In the average case, the coding gain of GME/GMC mode is 0.4 dB at high bitrate and 0.2 dB at low bitrate. Figure 3 shows the rate-distortion curves of no GME/GMC mode and GME/GMC mode with various global motion models in Foreman and Table Tennis. Figure 3a shows the rate-distortion curves in Foreman. The coding gain is 0.7 dB at very low bit rate (250 Kbps), and the coding gain is 0.4 dB at high bit rate compared to that without GME/GMC mode. The performances of four global motion models are similar and close in Fig. 3a. But in Fig. 3b which shows the comparison of Table Tennis, the performance of translation model is almost the same as that of no GMC mode. This is because scaling is the major global motion in Table Tennis, and the translation model cannot deal with this kind of camera motion. Hence the performance of translation model is lower 0.3 dB than those of others.

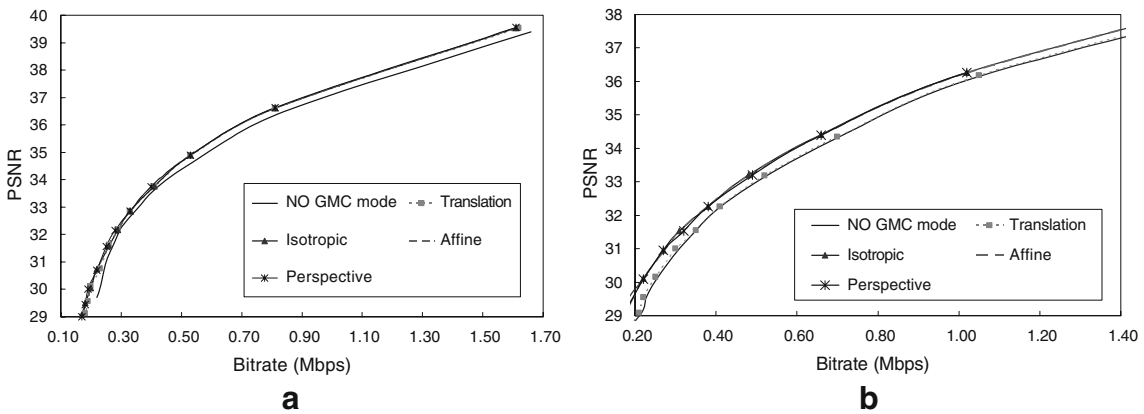

Figure 3 The RD curves of no GME/GMC mode, and GME/GMC mode with translation, isotropic, affine, and perspective models in **a** Foreman; **b** Table Tennis.

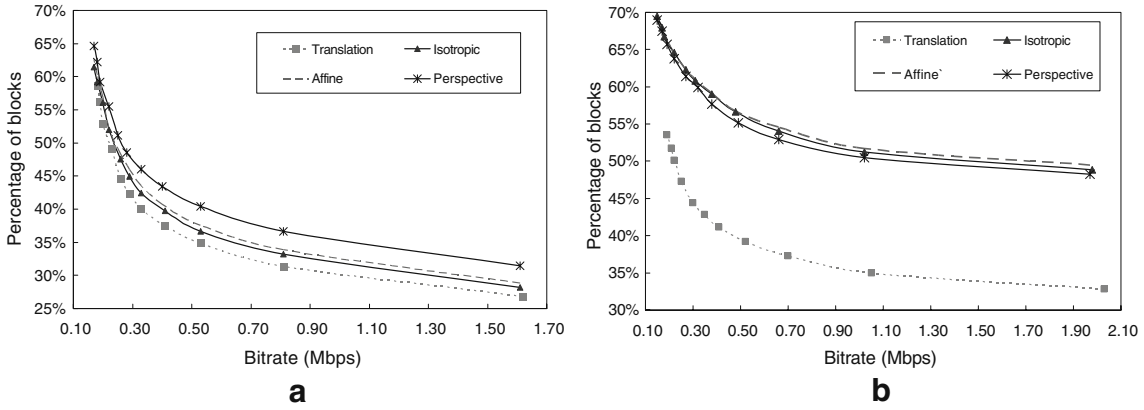

Figure 4 The percentage of macroblocks which are selected GMC mode with different global motion models in **a** Foreman ; **b** Table Tennis.

3.2 The Percentage of GME/GMC Macroblocks

Figure 4a and b show how many percentage of macroblocks are coded by GME/GMC mode in Foreman and Table Tennis, respectively. These macroblocks are called GMC macroblocks. Compared the performances of different global motion models, the coding gain is increased, as the percentage of GMC macroblocks is increased. In Fig. 4a, the percentages of GMC macroblocks for four global motion models are similar, so their performances are close. But in Fig. 4b, there are fewer GMC macroblocks for translation model, and the percentages of GMC macroblocks in the others global motion models are close. This phenomenon is also exhibited in the coding performance of Table Tennis. In average cases, there are 40% macroblocks which are coded by GME/GMC mode in one frame. If the sequence is still or small motion like Weather or Coastguard, the percentage of GMC macroblocks is higher than 70%. Moreover, at ultra low bitrate or a large quantization number, the percentage is increased significantly.

Although more than 40% macroblocks are coded by GME/GMC mode, the coding gain of GME/GMC mode in CIF Format is only 0.4 dB in average cases. This is because there are two factors to degrade the performance of GME/GMC mode. First, when GME/GMC mode is supported, for each current macroblock, one extra bit is required to represent the selected coding mode. Second, global motion parameters also have to be coded. These are the penalties of GME/GMC mode. Besides, in the local motion estimation and compensation (LME/LMC) mode, the motion vector is coded by DPCM. The motion vector difference between the motion vector and motion vector predictor is coded instead of the motion vector. If the motion vector predictor matches with the motion vector, only one bit is required to be coded. That is, only when the motion vector difference is not zero and the distortion of GME/GMC mode is less than that of LME/LMC mode, the coding gain of GME/GMC mode exists. Otherwise, even if there are many GMC macroblocks, the coding gain of GME/GMC mode is still not apparent. For example, although more than 55% macroblocks are GMC macroblocks in Table Tennis, the coding gain is only 0.3 dB at low bit rate. If we can reduce the overhead of GME/GMC mode, the coding gain of GME/GMC mode becomes apparent.

3.3 The Computation Complexity of GME/GMC Mode

Table 4 is the percentages of the run time for GME/GMC with different global motion models and LME/LMC with quarter pixel in an MPEG-4 ASP video coding system. The test sequence is Stefan, CIF Format, 30 fps. The run time of GME/GMC with translation, isotropic and affine models is 21%, 34%, and 57% of that of LME/LMC. From Table 4, the run time of GME/GMC mode is increased as the complexity

Table 4 The runtime percentage of GME/GMC and LME/LMC.

Global motion model	GME/GMC percentage (%)	LME/LMC percentage (%)	GME/LME ratio (%)
Translation model	16.8	78.1	21.5
Isotropic model	24.1	71.1	33.9
Affine model	34.8	61.1	56.9

Table 5 The instruction profiling of gradient descent with affine model.

Instruction	Percentage (%)
Arithmetic	33.14
Data instruction	54.33
Logic	1.26
Rotate & shifter	0.13
Jump, test & comparator	10.39
Stack	0.75
Floating-point operation	44.77

of global motion model increases. The computation complexity of GME/GMC mode is very huge, although GME/GMC is a frame-level operation. Therefore, a hardware accelerator of GME is required for realtime applications, such as camcoders with MPEG-4 ASP encoder.

As shown in Table 1, Gradient Descent with affine model takes 35 GIPS in Stefan at CIF Format, 30 fps. Table 5 further shows the detailed results of instruction profiling in Gradient Descent, where the simulation environment is the same as Table 1. Among huge operations, 44% of operations are floating-point operations, because a high precision is required for global motion parameters and is achieved by use of floating-point numbers. These floating-point operations enlarge the cost of hardware implementation. Besides, 55% of operations are about data instructions, including loading and storing. It means that ultra large memory access is required in this algorithm, as shown in Table 3. The requirement of memory bandwidth is too huge to be acceptable for hardware implementation. Therefore, how to effectively reduce memory bandwidth of Gradient Descent is an important issue for hardware design. Moreover, compared to four global motion models, isotropic model provides a good tradeoff between computation complexity and coding performance.

4 Proposed Hardware Architecture

4.1 Hardware Design Challenges

There are several hard problems of GME and Gradient Descent for hardware implementation. We described them in the following.

4.1.1 Huge Computation Complexity and Ultra Large Memory Bandwidth

The first problem is huge computation complexity and ultra large memory bandwidth. The huge computation complexity results in large hardware cost, and the ultra large memory bandwidth increases the loading of system bus and leads to the difficulty when the GME hardware accelerator is integrated into a complete MPEG-4 ASP video encoder system. However, this problem is solved by the proposed ARC-Gradient Descent algorithm. The computation complexity is only 20% of that in the original algorithm, and 91% memory bandwidth is saved in the average case.

4.1.2 Irregular Memory Access

The second problem is the irregular memory access. The problem is explained in Fig. 5. In Fig. 5, the circles are the reference pixels in the reference frame. The triangle is the corresponding position of current pixel. A square is a candidate of the corresponding positions of next current pixel. Figure 5 shows the phenomenons of irregular memory access with different global motion parameters. Isotropic model is taken as an example, which is

$$x' = m_0 x + m_1 y + m_2, \tag{10}$$

$$y' = -m_1 x + m_0 y + m_3. \tag{11}$$

Then, the difference, $(\triangle x, \triangle y)$, between the corresponding positions of two successive current pixels, (x, y) and $(x+1, y)$, is $(m_0, -m_1)$. In traditional motion model where $(m_0, -m_1)$ is always equal to $(1, 0)$, the memory access is regular and predictable. However, in isotropic model, scaling and shear are supported, and they are floating-point numbers for the requirement of their precision. Then, $(m_0, -m_1)$ becomes variable and is a set of floating-point numbers in GME. When the scaling factor is smaller than one, $m_0 < 1$, it is possible that the corresponding position of next current pixel stays at the same region of current pixel, as shown in Fig. 5a. Oppositely, if $m_0 > 1$, it is also possible that the corresponding position of next current pixel is not adjacent to the corresponding region of current pixel, as shown in Fig. 5b.

Figure 5c and d show the irregular memory access due to the shear factor. When the shear factor is positive, $m_1 > 0$, the corresponding positions of current pixels in the same row are moved up pixel by pixel in y direction. It is possible that the corresponding position of next current pixel is back to the last row in y direction, as shown in Fig. 5c. Conversely, if shear factor is negative, $m_1 < 0$, the corresponding positions of current pixels in the same row are moved down pixel by pixel in y direction, and may go to the next row, as shown in Fig. 5d.

Figure 5e shows the irregular memory access of GME, if $0 < m_0 < 2$ and $-1 < m_1 < 1$. It means that

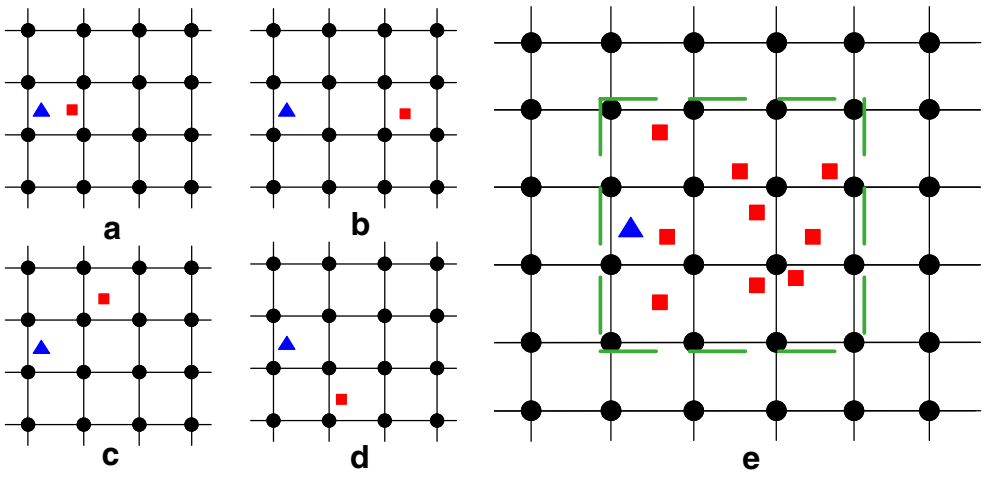

Figure 5 Irregular memory access of GME with isotropic model at different global motion parameters; (**a**) $m_0 < 1$; (**b**) $2 > m_0 > 1$; (**c**) $m_1 > 0$; (**d**) $m_1 < 0$; (**e**) $0 < m_0 < 2$ and $-1 < m_1 < 1$.

we cannot predict the corresponding position of next current pixel effectively. Irregular memory access also reduces the hardware utilization, when the scheduling of traditional motion estimation is adopted. This phenomenon becomes more and more serious, when the camera motion is larger and larger in scaling or shear dimension. We will discuss this problem with the scheduling of traditional motion estimation in detailed, in Section 4.2.4.

4.1.3 Memory Access of Interpolation and Differential Values

The third problem is that a lot of interpolation operations are required, because the corresponding position of current pixel is usually not an integer pixel. The corresponding reference value is interpolated based on the four neighboring pixels. The differential values, as shown in Eq. 9, are also required in this algorithm, in order to approach the final global motion parameters. Then, four neighboring reference pixels are necessary for computing one current pixel. If the memory access is regular and predictable, we can adopt the data reuse scheme to solve this problem easily. However, the data reuse between pixels is limited because of irregular memory access, and then the operating frequency is seriously dominated by the memory access of interpolation. How to efficiently access the required pixels is a challenge of hardware design.

4.1.4 Wordlength of Accumulated Values

The last problem is that the error and differential values of the whole frame have to be accumulated, as shown in Eqs. 7 and 8. These values have large magnitudes and variances. The required precision of these data is high in order to keep the performance. However, the hardware cost is increased as the wordlength increases, and due to this reason, a fixed-point number is not efficient for hardware implementation. On the other hand, although a floating-point number can provide a lower hardware cost, it sacrifices the precision in the process of accumulation. In short, how to provide a high precision with a low hardware cost is also a design challenge for hardware implementation of GME.

In the following subsections, we proposed a hardware architecture system of GME [31], and these design challenges are overcome. A new scheduling, Reference-Based Scheduling, is proposed to solve the irregular memory access. An interleaved memory arrangement is adopted to achieve that four neighboring pixels can be accessed in one cycle to reduce the operating frequency. A two-stage accumulator is also applied to reduce the wordlength and preserve the precision of accumulated values.

4.2 Proposed Hardware Architecture System

Based on the proposed ARC-Gradient Descent algorithm, a hardware architecture of GME for MPEG-4 ASP is proposed, as shown in Fig. 6. There are four major components, *GME controller*, *3-Tap Filter & Subsample*, *Initial Matching*, and *Gradient Descent with Local Memory*. An off-chip frame memory is required in this architecture to store the reference and current frames. *GME controller* controls other modules and decides which module takes action. Each module is described in detailed in the following subsections.

4.2.1 3-Tap Filter & Subsample

3-tap filter and frame subsampling of the proposed ARC-Gradient Descent algorithm are implemented in this module. Current frame is filtered by a 2-D 3-tap

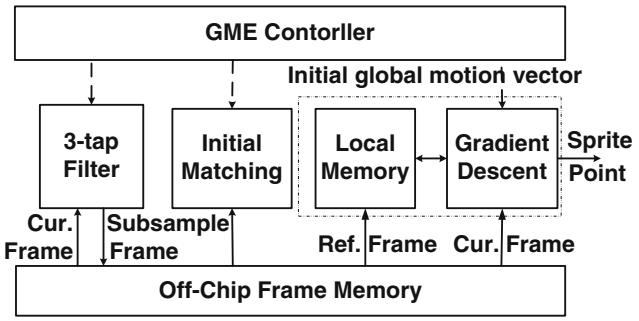

Figure 6 The system overview of hardware architecture for GME.

filter and subsampled by two in horizontal and vertical directions, respectively. The 2-D 3-tap filter can be decomposed into two 1-D separated filters, vertical and horizontal filters, as shown in Fig. 7. The data are filtered by horizontal filter and subsampled in horizontal direction before the vertical filter. Because of the vertical filter, two delay lines are required to store temporary data after horizontal filter. In general cases, the delay lines are implemented by dual ports memories. However, because of the horizontal subsample, the data rate of vertical filter is only half of the horizontal filter, and then a single port memory is enough to be the delay line. The data are filtered by the horizontal filter and written into the delay line in one cycle. In the next cycle, the data in the delay line are read and filtered by the vertical filter. At the same time, the output in the horizontal filter can be discarded because of subsampling in the horizontal direction. After the vertical filter, the data are outputted to the off-chip memories.

4.2.2 Initial Matching

In the proposed ARC-Gradient Descent algorithm, *Initial Matching* is executed once in every 30 frames, and it only applied on the 1/16 frames. Therefore, considering hardware efficiency and computation complexity of this function, one processing element is sufficient. The hardware architecture of *Initial Matching* is similar to other conventional motion estimation architectures and it consists of two modules, as shown in Fig. 8a. One is *Processing Element*, and the other is *Error Histogram*. *Processing Element* is responsible for calculating the absolute difference and accumulating the total error of whole frame for each candidate. The searching range is $[-8, 7]$ at the smallest frame size. After examining all searching candidates, the motion vector which has the smallest total error is selected as the initial prediction for the next stage. *Error Histogram* is responsible for calculating the distribution of the error in the whole frame and decide the error threshold for the next stage.

4.2.3 Gradient Descent with Local Memory

The detailed architecture of *Gradient Descent with Local Memory* is shown in Fig. 8b. *Controller* generates the address of current pixel and calculates the corresponding position of current pixel in the reference frame according to Eqs. 10 and 11 with global motion parameters at the last iteration. The luminance of reference pixel is interpolated in *Basic Vector*, where the basic differential values used to compute the matrix elements in Eq. 9 are also calculated. The distribution of errors is estimated in *Error Histogram*, and then the error threshold is derived. The elements of matrix A and B in Eqs. 7 and 8, are computed and accumulated in *Matrix Element*. After finishing the processing of the accumulation, the inverse matrix A^{-1} and matrix multiplication $A^{-1}B$ in Eq. 5 are calculated in *Floating-Point Matrix Processor*. *Sprite Point* checks if the global motion parameters converge or not and calculates the corresponding points of reference points, sprite points, for GME/GMC mode in MPEG-4 ASP.

4.2.4 Reference-Based Scheduling

In GME algorithms, the irregular memory access is inevitable. This problem leads to the difficulty of memory access and reduces the hardware utilization with the scheduling of traditional motion estimation. In the

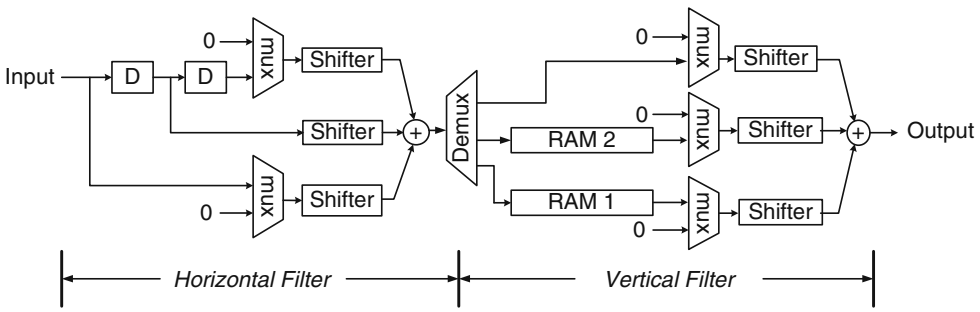

Figure 7 The hardware architecture of *3-tap filter & subsample*.

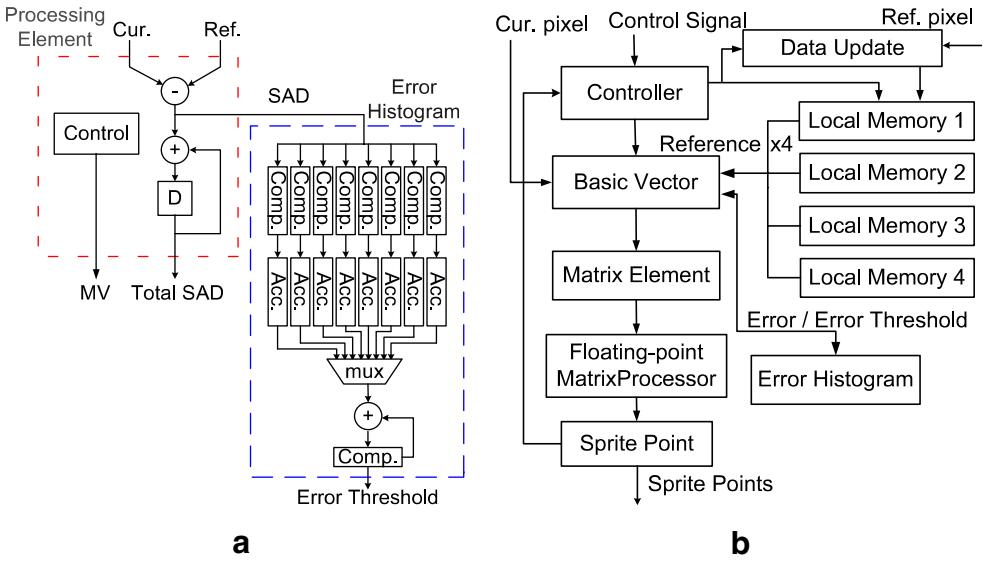

Figure 8 The hardware architectures of **a** *Initial Matching*; **b** *Gradient Descent with Local Memory*.

scheduling of traditional motion estimation, the reference data in local memory are loaded based on the current macroblock. Figure 9a shows the result of GME with the scheduling of traditional motion estimation. Because of the shear factor, the corresponding region is a sheared square, when the global motion model is isotropic model. Consequently, the penalty of the external memory access is large due to the irregular memory access. Besides, because scaling and shear factors are variable, the size of the corresponding region is also variable. A larger local memory for the worst case is required, but the hardware utilization is low in the average or the best case.

Figure 9b shows the proposed scheduling, Reference-Based Scheduling. Compared to the traditional scheduling of motion estimation, the roles of reference and current frames are exchanged. In the proposed Reference-Based Scheduling, based on the region of the reference data in the local memory, the current data are decided to be processed or not. In the beginning, the reference frame is segmented into several sections. For each section, the data are loaded into local memories,

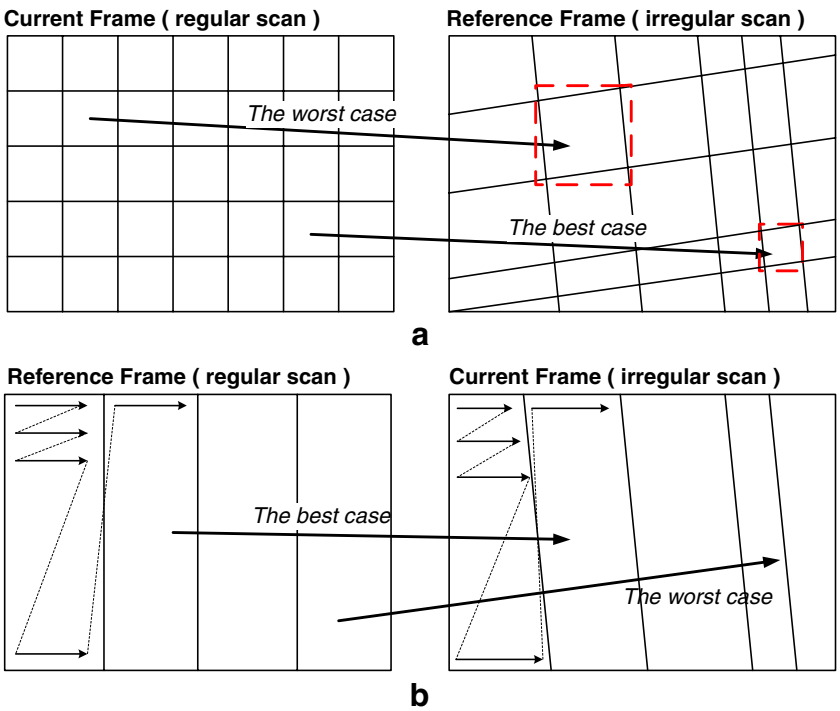

Figure 9 The different schedulings of GME; **a** The scheduling of traditional motion estimation; **b** The proposed scheduling, Reference-Based Scheduling.

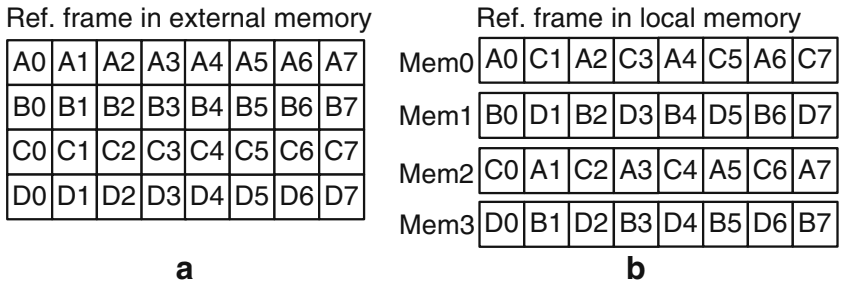

Figure 10 The data arrangement of reference frame in **a** External memory; **b** Local memory.

row by row. At the same time, those current pixels, whose corresponding positions are located in this section, are calculated row by row. If the corresponding position of current pixel in one row is not located in this section, the computation of this row is terminated, and the position of current pixel is recorded. After finishing the computation in one section, the procedure goes to the next section until all sections have been computed.

By this way, the utilization of local memory is almost 100% regardless of the best or worst case. The data reuse of reference frame is nearly achieved the maximum. In a word, although irregular memory access is inevitable, we proposed Reference-Based Scheduling so that the scan order of the reference frame becomes regular, and that of the current frame is also nearly regular. Not only the impact of irregular memory access can be reduced, but also the data in the reference frame can be reused as much as possible.

4.2.5 Interleaved Memory Arrangement

Because of the interpolation and differential values, four neighboring reference pixels are required for computing one current pixel. If only one neighboring reference pixel is gotten in one cycle, the operating frequency should be four times of the total number of current pixels in one second, at least. If we can get four neighboring pixels in one cycle, 75% cycles can be saved compared to the previous scheme. An interleaved memory arrangement is applied to solve this problem. Figure 10a and b show the data arrangements and relationship between external and local memories. The data arrangement of reference frame in external memory is raster scan. The data arrangement of reference frame in local memories is interleaved. The local memory consists of four dual port memories. By the proposed data arrangement of local memories, four neighboring reference pixels are located into different memory banks. Therefore, we can access them in one cycle.

4.2.6 Two-Stage Accumulator

In *Matrix Element*, elements of matrix A and B, as shown in Eqs. 7 and 8, are accumulated. Because these accumulated elements are the related information of whole frame, they have large variances and magnitudes. Therefore, if a fixed-point number is applied to represent these data, the wordlength is large. Then the hardware cost is high. On the contrary, if a floating-point number is adopted, the precision of these data is lost in the process of accumulation. The performance of GME is degraded. Considering the hardware cost and performance, a two-stage accumulator is proposed to accumulate them. In the first stage, a fixed-point accumulator is used to accumulate the matrix element. When the partial result of the first stage is larger than the accumulated threshold, a floating-point accumulator is adopted to accumulate the partial results of the

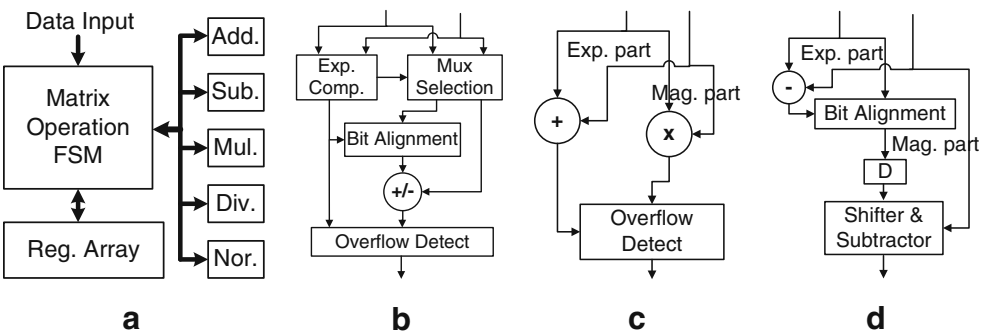

Figure 11 The hardware architecture of **a** *Floating-point matrix operation* system; **b** Floating-point adder and subtractor; **c** Floating-point multiplier; **d** Floating-point divider.

first stage in the second stage. By this accumulator, the wordlength can be reduced, and the precision can be preserved.

4.2.7 Floating-Point Matrix Processor

Floating-Point Matrix Processor is responsible for the calculation of inverse matrix and matrix multiplication in Eq. 5. In *Floating-Point Matrix Processor*, there are three major modules, *Matrix Operation FSM*, *Reg. Array*, and *Operators*, which includes adder (Add.), subtractor (Sub.), multiplier (Mul.), divider (Div.), and normalizer (Nor.), as shown in Fig. 11a. *Matrix Operation FSM* is the finite state machine which executes the operations of a 4×4 inverse matrix and a multiplication of a 4×4 and 4×1 matrixes. *Reg. Array* is the data buffer for storing the input values and partial results. Figure 11b, c, and d show the detailed architectures of *Operators*. Because they are floating-point operations, the operations of exponential part, bit alignment, and overflow detection are required in these architectures. Except the divider, the others finish one operation in one cycle. The divider is a multi-cycle-shift-and-subtract divider, and it can generate one bit of the quotient per cycle. *Floating-Point Matrix Processor* executes five operators at the same time and takes 137 cycles to finish the operations of a 4×4 inverse matrix and a multiplication of a 4×4 and 4×1 matrixes.

5 Hardware Implementation Result

The implementation results and our specification of the proposed GME hardware accelerator are listed in Table 6. The target specification is MPEG-4 ASP@L3, which is 352×288 with 30 fps. The working frequency is 30 MHz. The hardware is implemented with Verilog-HDL and synthesized with SYNOPSYS Design Compiler. ARTISAN 0.18*um* cell library is adopted to design the hardware. Figure 12 shows the layout of our proposed GME hardware accelerator. The detailed distribution of gate count and memory usages is shown in Table 7. The total gate count is about 131 K. Local memory size is 2.8 Kbits and 5.1 Kbits in the *3-Tap Filter & Subsample* and *Gradient Descent*, respectively. Although we have proposed two-stage accumulators to reduce the hardware cost of *Matrix Element*, the gate count of *Matrix Element* is still large. The gate count of *Floating-point Matrix Processor* is also 26 K.

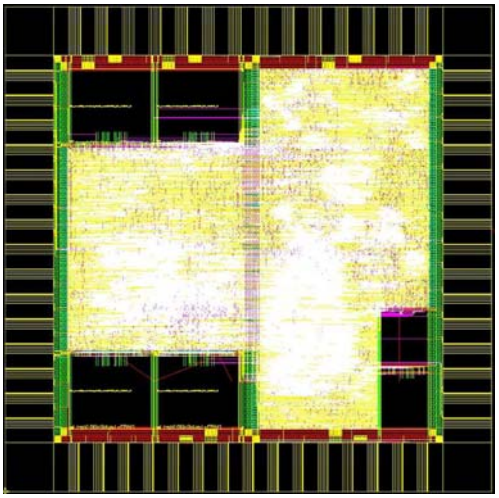

Figure 12 The layout of the proposed hardware accelerator for GME.

Compared with the other GME hardware architecture, SPM [33], whose target is also MPEG-4 ASP@(L2, L3), the gate count is 66 K, and internal memory is 31 Kb with AVANT! 0.35*um* cell library. The comparison is shown in Table 8. Although our gate count is double of that in SPM, the required operating frequency is only 1/3 of that in SPM. Moreover, our internal memory size is only 26.7% of SPM, and the required memory

Table 6 The specification of proposed hardware accelerator.

Technology	UMC 0.18um CMOS 1P6M
Package	68CLCC
Die size	1.94916 × 1.94628 mm
Core size	1.44936 × 1.44648 mm
Gate count	130,935
On-chip memory	2 176 × 8 single port SRAM
	4 160 × 8 dual ports SRAM
Work clock rate	30 MHz
Processing ability	CIF Format with 30 fps @ 30 MHz
Power consumption	29.59 mW @ 30 MHz, 1.8 V

Table 7 The hardware implementation results for GME.

Module	Gate count	On-chip memory
3-tap filter & subsample	1,144	2,816 bits
Initial matching	3,718	0
Sprite point	3,348	0
Gradient descent		
Controller	18,921	0
Basic vector	2,215	0
Error histogram	2,370	0
Matrix element	72,911	0
Floating-point matrix processor	26,308	0
Local memory	0	5,120 bits
Total	130,935	7,936 bits

Table 8 The comparison between our proposed and SPM at MPEG-4 ASP@L3.

Architecture	Our proposal	SPM
Gate counts	131 K	66 K
Frequency	30 MHz	100 MHz
On-chip memory	7.9 Kbits	30.8 Kbits
Memory bandwidth	19.1 Mbyte/s	198.0 Mbytes/s

bandwidth is only 19.10 MB/s in the average case, which is 9.65% of SPM. Hence our design is better and more suitable for the integration of video coding systems than SPM.

6 Conclusion

GME/GMC plays an important role in many applications including video segmentation, MPEG-7 descriptors, and video coding. Many GME algorithms are proposed to be applied in various applications. Based on our analyses, Gradient Descent is better than other algorithms, if the tradeoff between the computation complexity and the performance of an algorithm is considered. Moreover, because of the huge computation complexity and ultra large memory bandwidth, a hardware accelerator is required for realtime applications of GME. There are few hardware architectures, because there are several hardware design challenges of GME. The first problem is the huge memory bandwidth. This problem is solved by our proposed ARC-Gradient Descent algorithm. In this algorithm, 91.2% memory bandwidth and 80% iterations can be saved. The second problem is the irregular memory access. Although the irregular memory access is inevitable, the impact of the irregular memory access is largely reduced by the proposed Reference-Based Scheduling. Finally, an interleaved memory arrangement is applied to access four neighboring reference pixels. This technique satisfies the memory access requirement of the interpolation and differential values, and it can also reduce the cycles of memory access. Based on the above techniques, a hardware architecture for GME in MPEG-4 ASP@L3 is proposed. The gate count is 131 K with 7.9 Kb on-chip memory, the operating frequency is 30 MHz, and the required memory bandwidth is only 10% of the previous work, which is much suitable to be integrated into MPEG-4 ASP encoder. Besides, the provided information from GME hardware can be also utilized in may applications, such as image stabilizer, scene change detection, MPEG-7 descriptor, and so on.

References

1. Dufaux, F., & Konrad, J. (2000). Efficient, robust, and fast global motion estimation for video coding. *IEEE Transactions on Image Processing, 9*, 497–501.
2. Erturk, S. (2003). Digital image stabilization with sub-image phase correlation based global motion estimation. *IEEE Transactions on Consumer Electronics, 49*, 1320–1325.
3. Hoetter, M. (1989). Differential estimation of the global motion parameters zoom and pan. *Signal Processing, 16*, 249–265, March.
4. Chen, H. H., Kiang, C.-K., Peng, Y.-C., & Chang, H.-A. (2007). Integration of digital stabilizer with video codec for digital video cameras. *IEEE Transactions on Circuits and Systems for Video Technology, 17*(7), 801–813.
5. Lu, Y., Gao, W., & Wu, F. (2001). Sprite generation for frame-based video coding. In *Proc. of IEEE int. conf. on image processing* (Vol. 1, pp. 473–476).
6. Lu, Y., Ga, W., & Wu, F. (2002). Automatic video segmentation using a novel background model. In *Proc. of IEEE int. symp. on circuits syst.* (Vol. 3, pp. 807–810), May.
7. Qi, B., & Amer, A. (2005). Robust and fast global motion estimation oriented to video object segmentation. In *Proc. of IEEE int. conf. on image processing.* (Vol. 1, pp. 153–156), September.
8. ISO/IEC (2001). *Text of ISO/IEC 15938-3/FCD information technology – multimedia content description interface - part 3 visual*. ISO/IEC JTC 1/SC 29/WG11 N4062.
9. Manjunath, B. S., Salembier, P., & Sikora, T. (2002). *Introduction to MPEG-7*. New York: Wiley.
10. Smolic, A., Sikora, T., & Ohm, J.-R. (1999). Long-term global motion estimation and its application for sprite coding, content description, and segmentation. *IEEE Transactions on Circuits and Systems for Video Technology, 9*(8), 1227–1242.
11. Irani, M., Anandan, P., & Hsu, S. (1995). Mosaic based representations of video sequences and their applications. In *Proc. of IEEE int. conf. on comput. vision* (pp. 605–611).
12. Chien, S.-Y., Chen, C.-Y., Chao, W.-M., Hsu, C.-W., Huang, Y. W., & Chen, L.-G. (2002). A fast and high subjective quality sprite generation algorithm with frame skipping and multiple sprites techniques. In *Proc. of IEEE int. conf. on image processing* (Vol. 1, pp. 193–196).
13. Lu, Y., Gao, W., & Wu, F. (2003). Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(8), 394–405, May.
14. Chen, C.-Y., Chien, S.-Y., Chen, Y.-H., Huang, Y.-W., & Chen, L.-G. (2003). Unsupervised object-based sprite coding system for tennis sport. In *Proc. of IEEE int. conf. multimedia expo* (Vol. 1, pp. 337–340).
15. ISO/IEC (1999). *Information technology – coding of audio-visual objects – part 2: visual*. ISO/IEC 14496-2.
16. Dufaux, F. (1996). *Results for video coding using dynamic sprite* (*core experiment N3*). ISO/IEC JTC1/SC29/WG11 M1458.
17. Steinbach, E., Wiegand, T., & Girod, B. (1999). Using multiple global motion models for improved block-based video coding. In *Proc. of IEEE int. conf. on image processing* (Vol. 2, pp. 56–60).
18. Keller, Y., & Averbuch, A. (2003). Fast gradient methods based on global motion estimation for video compression. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(8), 300–309.
19. Stolberg, H.-J., Berekovic, M., Pirsch, P., & Runge, H. (2002). The MPEG-4 advanced simple profile - a complexity

study. In *Proc. of 2002 workshop and exhibition on MPEG-4* (pp. 33 –36).
20. Fu, M.-F., Au, O., & Chan, W.-C. (2003). Fast global motion estimation based on local motion segmentation. In *Proc. of IEEE int. conf. on image processing* (Vol. 3, pp. 367–370).
21. Moscheni, F., Dufaux, F., & Kunt, M. (1995). A new two-stage global/local motion estimation based on a background/foreground segmentation. In *Proc. of IEEE int. conf. on acoust., speech, and signal processing* (pp. 2261–2264).
22. Kim, E. T., & Kim, H.-M. (1998). Fast and robust parameter estimation method for global motion compensation in the video coder. *IEEE Transactions on Consumer Electronics, 45*(1), 76–83.
23. MPEG Video Group (2001). *The MPEG-4 video standard verification model version 18.0*. ISO/IEC JTC 1/SC 29/WG11 N3908.
24. Wu, S. F., & Kittler, J. (1990). A differential method for simultaneous estimation of rotation, change of scale and translation. *Signal Processing: Image Communication, 2*(1), 69–80, May.
25. Adolph, D., & Buschmann, R. (1991). 1.15Mbit/s coding of video signals including global motion compensation. *Signal Processing: Image Communication, 3*(2–3), 259–274, June.
26. Chan, W.-C., Au, O., & Fu, M.-F. (2002). A novel predictive global motion estimation for video coding. In *Proc. of IEEE int. symp. on circuits syst.* (Vol. 3, pp. 5–8), May.
27. Berekovic, M., Stolberg, H.-J., & Pirsch, P. (2002). Multicore system-on-chip architecture for MPEG-4 streaming video. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(8), 688–699.
28. Irani, M., & Peleg, S. (1993). Motion analysis for image enhancement: Resolution, occulusion and transparency. *Journal of Visual Communication and Image Representation, 4*(4), 324–335.
29. Badawy, W., & Bayoumi, M. (2002). A multiplication-free algorithm and a parallel architecture for affine transformation. *Journal of VLSI Signal Processing, 31*(2), 173–184, June.
30. Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics, 11*, 431–441.
31. Chen, C.-Y., Chien, S.-Y., Chao, W.-M., Huang, Y.-W., & Chen, L.-G. (2004). Hardware architecture for global motion estimation for MPEG-4 advanced simple profile. In *Proc. of IEEE int. symp. on circuit and system* (Vol. 2, pp. 23–26), May.
32. Richter, H., Smolic, A., Stabernack, B., & Muller, E. (2001). Real time global motion estimation for an MPEG-4 video encoder. In *Proceedings of picture coding symposium* (pp. 25–27).
33. Chien, S.-Y., Chen, C.-Y., Chao, W.-M., Huang, Y.-W., & Chen, L.-G. (2003). Analysis and hardware architecture for global motion estimation in MPEG-4 advanced simple profile. In *Proc. of int. symp. on circuits syst.* (Vol. 2, pp. 720–723), May.

Yi-Hau Chen was born in Taipei, Taiwan, R.O.C., in 1981. He received the B.S.E.E degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., in 2003. Now he is working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University. His major research interests include the algorithm and related VLSI architectures of global/local motion estimation, H.264/AVC, scalable video coding.

Shao-Yi Chien received the B.S. and Ph.D. degrees from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, in 1999 and 2003, respectively. During 2003 to 2004, he was a research staff in Quanta Research Institute, Tao Yuan Shien, Taiwan. In 2004, he joined the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, as an Assistant Professor. His research interests include video segmentation algorithm, intelligent video coding technology, image processing, computer graphics, and associated VLSI architectures.

Ching-Yeh Chen was born in Taipei, Taiwan, R.O.C., in 1980. He received the B.S., and Ph.D. degrees in electrical engineering from National Taiwan University (NTU), Taipei, Taiwan, R.O.C. in 2002, and 2006, respectively. He joined MediaTek Inc. from 2006. His research interests include intelligent video signal processing, global/local motion estimation, scalable video coding, and associated VLSI architectures.

Yu-Wen Huang was born in Kaohsiung, Taiwan, in 1978. He received the B.S. degree in electrical engineering and Ph.D. degree in the Graduate Institute of Electronics Engineering from National Taiwan University (NTU), Taipei, in 2000 and 2004, respectively. He joined MediaTek, Inc., Hsinchu, Taiwan, in 2004, where he develops integrated circuits related to video coding systems. His research interests include video segmentation, moving object detection and tracking, intelligent video coding technology, motion estimation, face detection and recognition, H.264/AVC video coding, and associated VLSI architectures.

Liang-Gee Chen was born in Yun-Lin, Taiwan, in 1956. He received the BS, MS, and Ph.D. degrees in Electrical Engineering from National Cheng Kung University, in 1979, 1981, and 1986, respectively.

He was an Instructor (1981–1986), and an Associate Professor (1986–1988) in the Department of Electrical Engineering, National Cheng Kung University. In the military service during 1987 and 1988, he was an Associate Professor in the Institute of Resource Management, Defense Management College. From 1988, he joined the Department of Electrical Engineering, National Taiwan University. During 1993 to 1994 he was Visiting Consultant of DSP Research Department, AT&T Bell Lab, Murray Hill. At 1997, he was the visiting scholar of the Department of Electrical Engineering, University, of Washington, Seattle. Currently, he is Professor of National Taiwan University. From 2004, he is also the Executive Vice President and the General Director of Electronics Research and Service Organization (ERSO) in the Industrial Technology Research Institute (ITRI). His current research interests are DSP architecture design, video processor design, and video coding system.

Dr. Chen is a Fellow of IEEE. He is also a member of the honor society Phi Tan Phi. He was the general chairman of the 7th VLSI Design CAD Symposium. He is also the general chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He serves as Associate Editor of IEEE Trans. on Circuits and Systems for Video Technology from June 1996 until now and the Associate Editor of IEEE Trans. on VLSI Systems from January 1999 until now. He was the Associate Editor of the Journal of Circuits, Systems, and Signal Processing from 1999 until now. He served as the Guest Editor of The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, November 2001. He is also the Associate Editor of the IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing. From 2002, he is also the Associate Editor of Proceedings of the IEEE.

Dr. Chen received the Best Paper Award from ROC Computer Society in 1990 and 1994. From 1991 to 1999, he received Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in VLSI design track. In 1993, he received the Annual Paper Award of Chinese Engineer Society. In 1996, he received the Out-standing Research Award from NSC, and the Dragon Excellence Award for Acer. He is elected as the IEEE Circuits and Systems Distinguished Lecturer from 2001–2002.