Contour-Based Binary Motion Estimation Algorithm and VLSI Design for MPEG-4 Shape Coding

Tsung-Han Tsai, Chia-Pin Chen, and Yu-Nan Pan
Department of Electronic Engineering
National Central University, Chung-Li, Taiwan, R.O.C

1. Introduction

MPEG-4 is a new international standard for multimedia communication [1]. It provides a set of tools for object-based coding of natural and synthetic videos/audios. MPEG-4 also enables content-based functionalities by introducing the concept of video object plane (VOP), and such a content-based representation is a key to enable interactivity with objects for a variety of multimedia applications. The VOP is composed of texture components (YUV) and an alpha component [2]-0. The texture component contains the colorific information of video object, and the alpha component contains the information to identify the pixels. The pixels which are inside an object are opaque and the pixels which are outside the object are transparent. MPEG-4 supports a content-based representation by allowing the coding of the alpha component along with the object texture and motion information. Therefore, MPEG-4 shape coding becomes the key technology for supporting the content-based video coding.
MPEG-4 shape coding mainly comprises the following coding algorithms: binary-shaped motion estimation/motion compensation (BME/BMC), context-based arithmetic coding (CAE), size conversion, mode decision, and so on. As full search (FS) algorithm is adopted for MPEG-4 shape coding, most of the computational complexity is due to binary motion estimation (BME). From the profiling on shape coding in Fig. 1, it can be seen that BME contributes to 90% of total computational complexity of MPEG-4 shape encoder. It is well known that an effective and popular technique to reduce the temporal redundancy of BME, called block-matching motion estimation, has been widely adopted in various video coding standard, such as MPEG-2 0, H.263 [5] and MPEG-4 shape coding [1]. In block-matching motion estimation, the most accurate strategy is the full search algorithm which exhaustively evaluates all possible candidate motion vectors over a predetermined neighborhood search window to find the global minimum block distortion position.

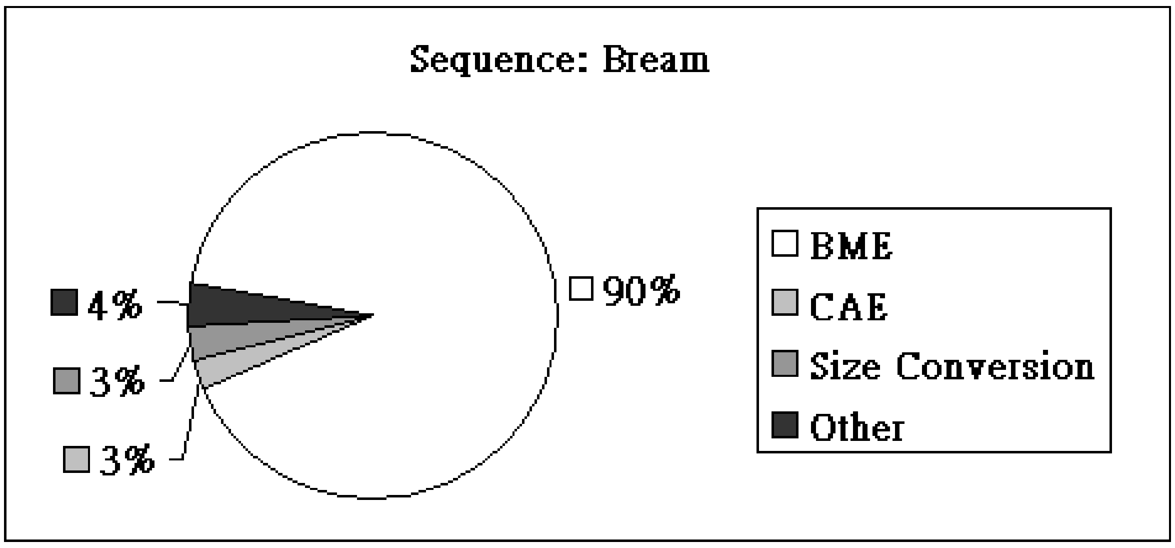


Fig. 1. Computational complexity of MPEG-4 shape encoder.

Fast BME algorithms for MPEG-4 shape coding were presented in several previous papers [6]-[8]. Among these techniques, our previous work, contour-based binary motion estimation (CBBME), largely reduced the computational complexity of shape coding [9]. It is applied with the properties of boundary search for block-matching motion estimation, and the diamond search pattern for further improvement. This algorithm can largely reduce the number of search points to 0.6% compared with that of full search method, which is described in MPEG-4 verification model (VM) [2].

In contrast with algorithm-level developments, architecture-level designs for shape coding are relatively less. Generally, a completed shape coding method should include different types of algorithms. In CAE part, it needs some bit-level operation. However, in binary motion estimation part, a high speed search method is needed. With these algorithm combinations on the shape coding, implementation should not be as straightforward as expected and it offers some challenges especially on architecture design. Since MPEG-4 shape coding has features of high-computing and high-data-traffic properties, it is suitable with the consideration of efficient VLSI architecture design. Most literatures have also been presented to focus on the main computation-expensive part, BME, to improve its performance [10]. Additionally, CAE is also an important part for architecture design and discussed in [11]-[12]. They utilized the multi-symbol technique to accelerate the arithmetic coding performance. As regards the complete MPEG-4 shape coding, some of these designs utilized array processor to perform the shaping coding algorithm [13]-[15], while others used pipelined architecture [16]. They can reach the relative high performance at the expense of these high cost and high complexity architectures. All of them intuitively apply the full search algorithm for easy realization on architecture design. However, the algorithm-level achievement on the large reduction of computation complexity is attractive and not negligible. This demonstrates that, without the supporting on an efficient algorithm, the straightforward implementation based on full search algorithm is hard to reach a cost-effective design.

In this paper, we proposed a fast BME algorithm, diamond boundary search (DBS), for MPEG-4 shape coding to reduce the number of search points. By using the properties of block-matching motion estimation in shape coding and diamond search pattern, we can skip a large number of search points in BME. Simulation results show that the proposed algorithm can marvelously reduce the number of search points to 0.6% compared with that of full search method, which is described in MPEG-4 verification model (VM)[2]. Compared with other fast BME in [6]-[7], the proposed BME algorithm uses less search points especially in high motion video sequences, such as 'Bream' and 'Forman'. We also present an efficient architecture design for MPEG-4 shape coding. This architecture is elaborated based on our fast shape coding algorithm with the binary motion estimation. Since this block-matching motion estimation can achieve the high performance based on the information of boundary mask, the dedicated architecture needs some optimization and consideration to reduce the memory access and processing cycles. Experimental results also demonstrate the equal performance on full-search based approach. This paper contributes a comprehensive exploration of the cost-effective architecture design of shaping coding, and is organized as follows.

In Section 2 the binary motion estimation in shape coding is described. We describe the highlights of the proposed fast BME algorithm for MPEG-4 shape coding in Section 3. The design exploration on CBBME is described in Section 4. In Section 5, the architecture design based on this BME algorithm is proposed. In Section 6, we present the implementation results and give some comparisons. Finally we summarize the conclusions in Section 7.

2. BME for MPEG-4 Shape Coding

The MPEG-4 VM [2] describes the coding method for binary shape information. It uses block-matching motion estimation to find the minimum block distortion position and sets the position to be motion vector for shape (MVS). The procedure of BME consists of two steps: first to determine motion vector predictor for shape (MVPS) and then to compute MVS accordingly.

MVPS is taken from a list of candidate motion vectors. As indicated in Fig. 2, the list of candidate motion vectors includes the shape motion vectors (MVS) from the three binary alpha blocks (BABs) which are adjacent to the current BAB and the texture motion vectors (MV) associated with the three adjacent texture blocks. By scanning the locations of MVS1, MVS2, MVS3, MV1, MV2 and MV3 in this order, MVPS is determined by taking the first encountered MV which is valid. Note that if the procedure fails to find a defined motion vector, the MVPS is set to (0, 0).

Based on MVPS determined above, the motion compensated (MC) error is computed by comparing the BAB indicated by the MVPS and current BAB. If the computed MC error is less or equal to 16xAlphaTH for any 4x4 sub-blocks, the MVPS is directly employed as MVS and the procedure terminates. Otherwise, MV is searched around the MVPS while computing sum of absolute difference (SAD) by comparing the BAB indicated by the MV and current BAB. The search range is ±16 pixels around MVPS along both horizontal and vertical directions. The MV that minimizes the SAD is taken as MVS and this is further interpreted as MV Difference for shape (MVDS), i.e. MVDS=MVS-MVPS.

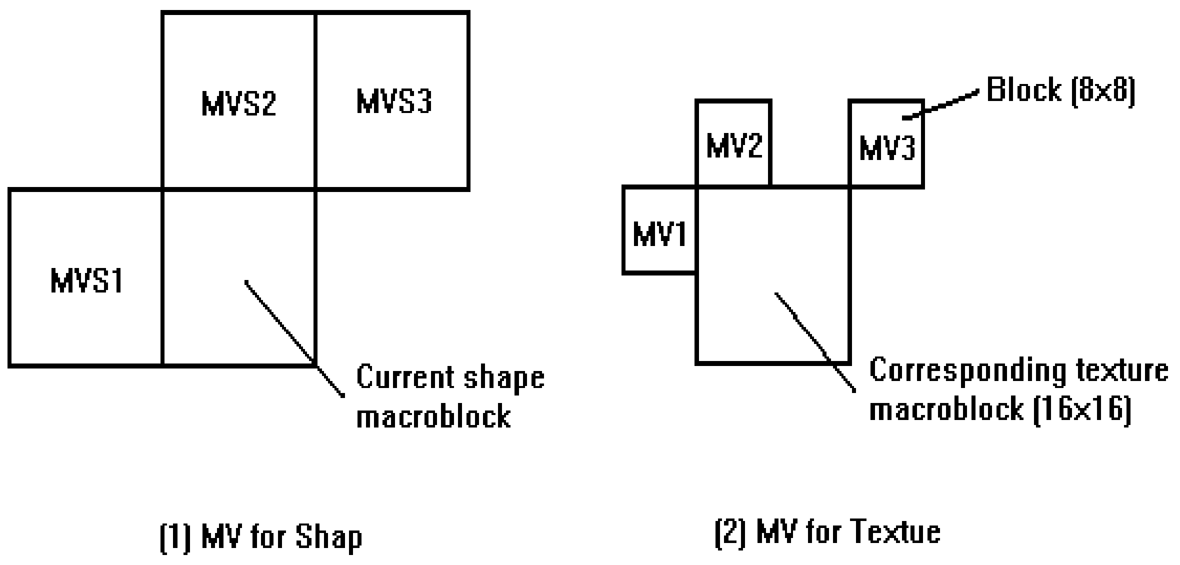


Fig. 2. Candidates for MVPS in shape VOP and texture VOP.

If more than one MVS minimize SAD by an identical value, the MVDS that minimizes the code length of MVDS is selected. If more than one MVS minimize SAD by an identical value with an identical code length of MVDS, MVDS with smaller vertical element is selected. If the vertical elements are also the same, MVDS with smaller horizontal element is selected.
After binary motion estimation, motion compensated block is constructed from the 16x16 BAB with a border of width 1 around the 16x16 BAB (bordered MC BAB). Then, context-based arithmetic encoding (CAE) is adopted for shape coding.

3. Proposed Contour-Based Binary Motion Estimation (CBBME) Method

The basic concept of the proposed BME method is that the contour of video objects in current BAB should overlap that in the motion compensated BAB, which is determined by binary motion estimation [9]. Therefore, those search positions, which contour lays apart from the contour of video objects in current BAB, can be skipped and the reduced number will be enormous. Moreover, based on the property that most real-world sequences have a central biased motion vector distribution [23], we use weighted SAD and diamond search pattern for furthermost improvement.

3.1 Definition of Boundary Pixel

In order to decide whether the pixel is on the contour of VOP, the boundary pixel is determined by the following procedure:

- If current pixel is opaque and one of its four adjacent pixels is transparent, the current pixel directly employed as boundary pixel.

Fig. 3 shows the correlation between current pixel and its four adjacent pixels. In the figure, the light grey area corresponds to the pixels outside the binary alpha map. It can be seen that the pixels outside the binary alpha map will not be taken into consideration, and the number of adjacent pixels will change into two or three.

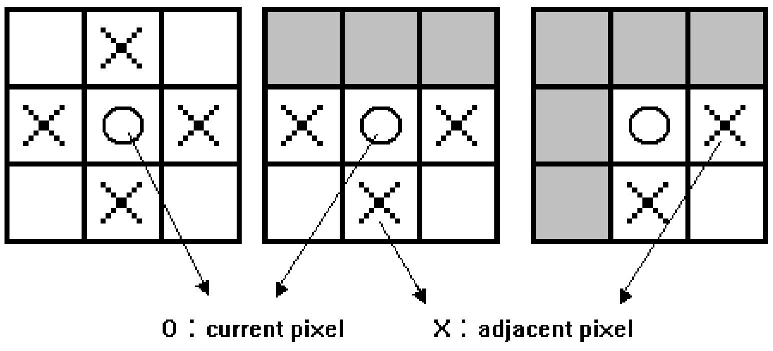


Fig. 3. The correlation between current pixel and adjacent pixel. Where gray part denotes pixels outside the VOP.

3.2 Boundary Search (BS)

According to the boundary pixels in VOP, we build a mask for BME process in MPEG-4 shape coding. Fig. 4 shows an example of boundary mask from the 'Foreman' sequence. In this figure, the white area denotes the efficient search position for fast BME, and it is much more efficient than the fast algorithm 0 illustrated in Fig. 5.

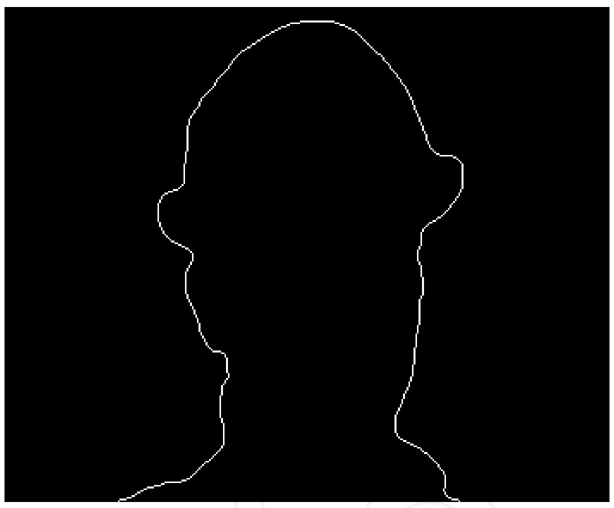


Fig. 4. Example of a boundary mask for shape coding. White area denotes boundary pixel.

A suggested implementation of the proposed Boundary search (BS) algorithm for shape coding is processed as follows:

Step 1. Perform a pixel loop over the entire reference VOP. If pixel (x,y) is an boundary pixel, set the mask at (x,y) to '1'. Otherwise set the mask at (x,y) to '0'.

Step 2. Perform a pixel loop over the entire current BAB. If pixel (i,j) is a boundary pixel, set (i,j) to be "reference point", and terminate the pixel loop. This step is illustrated in Fig. 6(b). Therefore, there is only one reference point in current BAB.

Step 3. For each search point within ±16 search range, check the pixel (x+i, y+j) which is fully aligned with the "reference point" from the current BAB. If the mask value at (x+i, y+j) is '1', which means that the reference point is on the boundary of reference VOP, the procedure will compute SAD of the search point (x, y). Otherwise, SAD of the search point (x, y) will not be computed, and the processing continues at the next position. Fig. 6(a) shows an example of this step. The search points in (x1, y1) and (x2, y2) will be skipped by this procedure, while the SAD will be computed in (x3, y3).

Step 4. When all the search points within ±16 search range is done, the MV that minimizes the SAD will be taken as MVS. Fig. 7 illustrates the overall scheme of proposed BS algorithm for MPEG-4 shape coding.

In the worst case, the proposed BS algorithm needs (256+ $(16+1)^2$) determinations to check whether the pixel is a boundary pixel. For each non-skipped search point, the SAD obtained by 256 exclusive-OR operations and 255 addition operations was taken as the distortion measure. However, based on BS algorithm, the number of non-skipped search points was reduced significantly, and the additional computational load due to BS algorithm was negligible.

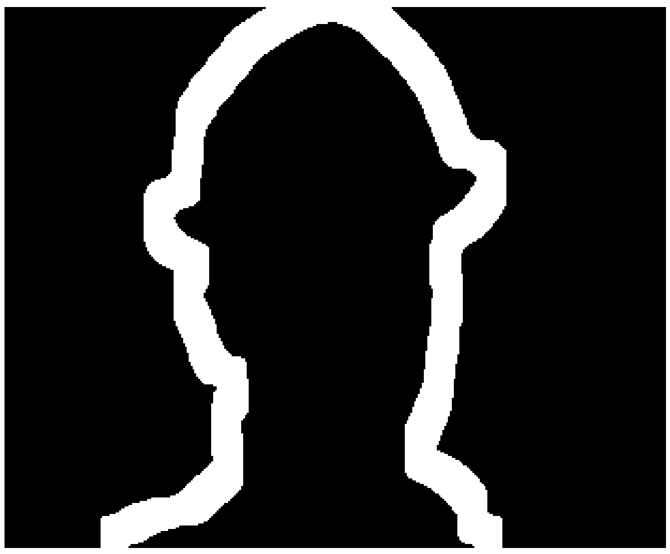


Fig. 5. Example of an effective search area, which has been proposed in reference 0.

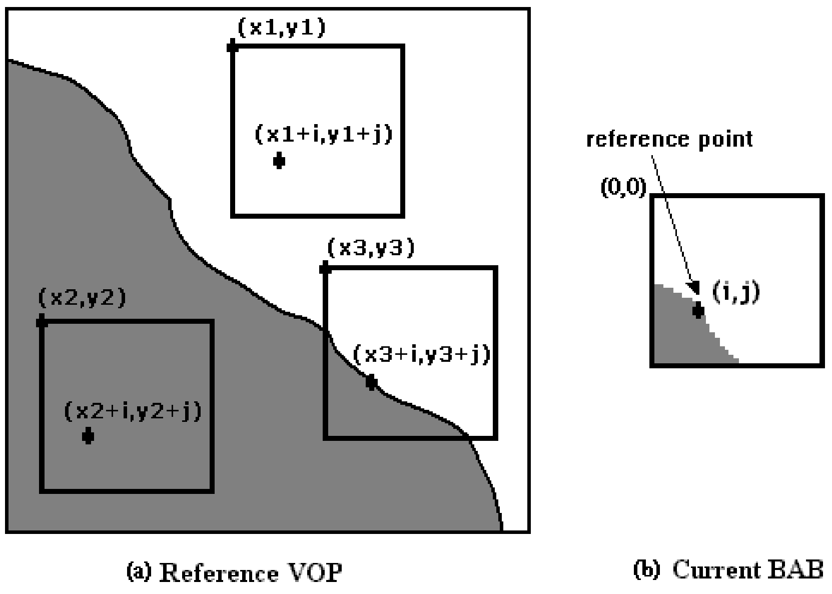


Fig. 6. The illustration of the proposed BS algorithm.

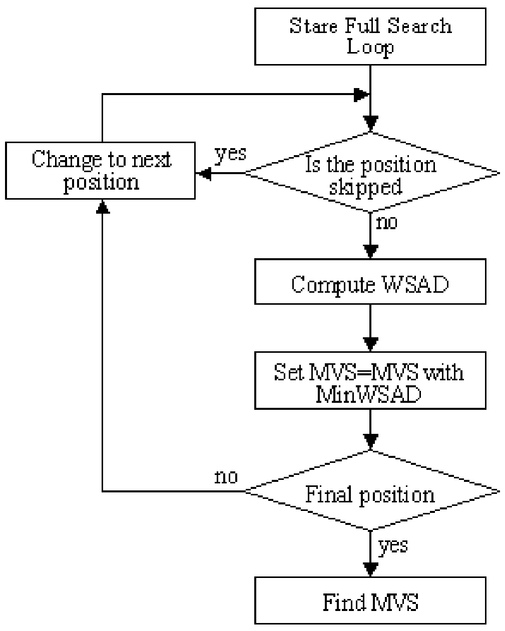


Fig. 7. Flow chart for proposed BS algorithm.

3.3 Diamond Boundary Search (DBS)

A better solution for block-matching motion estimation is to perform the search using a diamond pattern because of center-biased motion vector distribution characteristic. This is achieved by dividing the search area into diamond shaped zones and using half-stop criterion [17]. Fig. 8 shows an example of diamond shaped zones in a ±5 search window, and each number denotes the search zone in the search procedure.

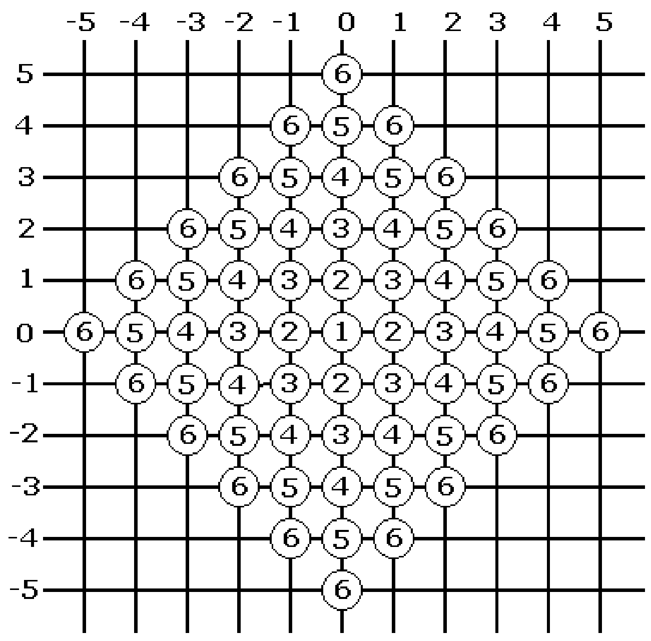


Fig. 8. A ±5 search window using diamond-shaped zones.

We combine the proposed BS algorithm with diamond-shaped zones, called DBS, and give different thresholds (Th_n) for each search zone for furthermost improvement. The procedure is explained in below.

Step 1. Construct diamond-shaped zones around MVPS within ±16 search window. Se t n=1.

Step 2. Calculate SAD for each search point in zone n. Let MinSAD be the smallest SAD up to now.

Step 3. If MinSAD≦Th_n, goto Step 4. Otherwise, set n=n+1 and goto Step 2.

Step 4. The motion vector is chosen according to the block corresponding to MinSAD.

3.4 Weight SAD (WSAD)

In MPEG-4 shape coding, the reference BAB with minimum SAD will be selected as motion compensated BAB. However, the BAB with minimum SAD may be far away from the original. It means that encoder should waste much more bits to code MVDS. Some previous motion estimation algorithms for color space used the concept of the weighted SAD (WSAD) to compensate the distortion [24]. In this paper, we proposed the similar concept of WSAD which takes both SAD and MVDS into consideration as the distortion measure. The WSAD is given by

$$WSAD=W_1*SAD+W_2*(|mvds\_x|+|mvds\_y|) \quad (1)$$

$$SAD=\Sigma\ \Sigma\ |p_{i-1}(i+u,j+v)-p_i(i,j)| \quad \dots \quad (2)$$

where *mvds_x* is MVDS in the horizontal direction and *mvds_y* is MVDS in the vertical direction. *W1* and *W2* denote the weighting values for SAD and MVDS, respectively. The WSAD is evaluated in every search points and the BAB with minimum WSAD is selected as motion compensated BAB. Based on the experimentation, *W1* and *W2* are determined as 10 and 7, respectively. The improvement of WSAD on bit-rate can compensate for the drawback when fast BME algorithm was adopted. Since the number of search points has been reduced significantly, the computational power due to calculate the WSAD will increase negligibly.

4. Design Exploration on CBBME

In this section, two approaches are developed based on CBBME. One is the center-biased motion vector distribution. The other is the search range shrinking. Both of them can further reduce the computation complexity in BME.

4.1 Center-biased motion vector distribution

The property is obvious that real-world sequences have a central biased motion vector distribution. This can be achieved by dividing the search area into diamond shaped zones and using half-stop criterion [17]. Diamond shaped zone has the higher center-biased distribution. The closer the search area to the center position, the less the number of the search zone in the search procedure.

4.2 Search Range Shrinking

With the property on Section 4.1, it is possible to reduce the default search range to a less size of search range. Default search range is ±16 in standard and it is straightforward used in conventional works. With the aid of WSAD in CBBME and the diamond shaped zones on search range, we make a search range shrinking technique from ±16 to ±13 pixels in our experimentation. Table 1 shows the simulation result of the proposed binary motion estimator using the above two techniques. This result supports the usage of the ±13 shrinking search range. An important contribution is that the WSAD makes some improvement on bit-rate. Therefore it takes similar even less bits to represent the shape per VOP in the same quality.

Sequence	Full Search (SR=±16)		Proposed Architecture with limited SSB (SR=±13)	
	Bits/VOP	%	Bits/VOP	%
Bream	1599.80	100	1599.27	99.97
News	890.22	100	890.18	100.00
Foreman	1186.94	100	1183.12	99.68
Children	2056.03	100	2055.43	99.97

Table 1. Performance comparison of FS and proposed architecture.

Fig. 9 shows the algorithm flow of CBBME. First, we construct diamond-shaped zones around motion vector predictor for shape (MVPS) within shrinking search range. Second, we calculate WSAD for each search point. Let MinWSAD be the smallest SAD up to now. If MinWSAD is smaller or equal to a threshold (*Th*) for each search zone, then the motion vector is chosen according to the block corresponding to MinWSAD; otherwise, it changes to next position and calculates the WSAD again.

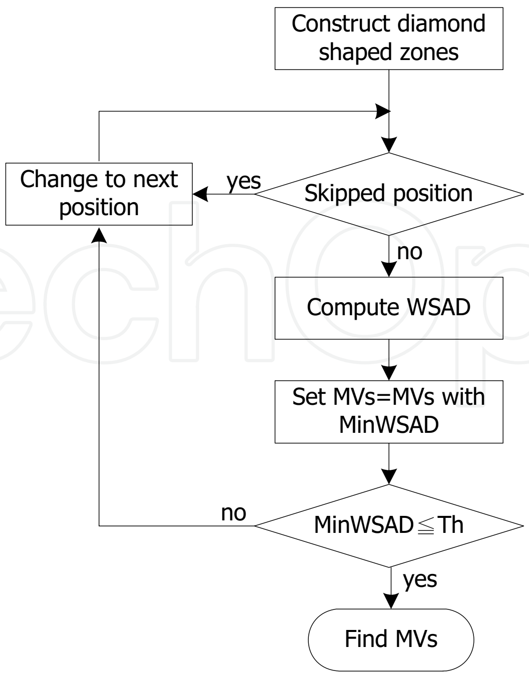


Fig. 9. Flow chart for the CBBME algorithm.

5. Architecture Design for MPEG-4 Shape Coding

The proposed MPEG-4 shape coding system, as illustrated in Fig. 10 consists of five major modules: BAB type decision, size conversion, BME, CAE and variable length coding (VLC). Based on the computational complexity analysis in Fig. 1, it can be seen that BME, size conversion and CAE take a great part of the processing time. In this section we propose an efficient architecture for these main modules with some design optimization and consideration.

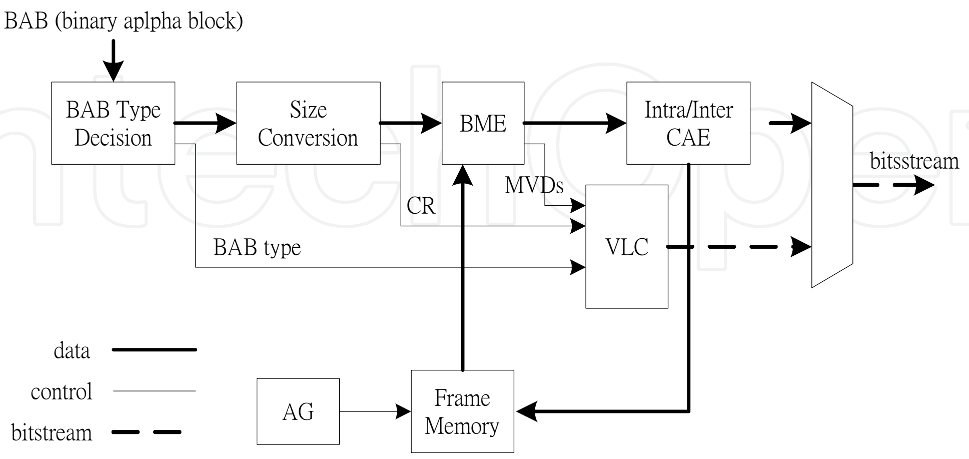


Fig. 10. Block diagram of MPEG-4 shape encoding.

5.1 Binary Motion Estimation

Fast motion estimation architectures were presented in several previous papers [12]-[15]. The performance could be high but high cost and high complexity architectures are always needed. Therefore, a novel and efficient architecture for binary motion estimation using proposed CBBME algorithm is proposed. Fig. 11 shows the block diagram of the proposed BME architecture. It mainly consists of a Boundary Pixel Detector, a processing element (PE) Array, a Compare and Selection (CAS) module and two main memories for search range (SR) and BAB buffer. Boundary Pixel Detector finds the "reference point" in current BAB and checks whether the candidate search positions are non-skipped positions. PE Array performs the binary motion estimation. CAS finds the minimum WSAD and its MVDS for shape coding.

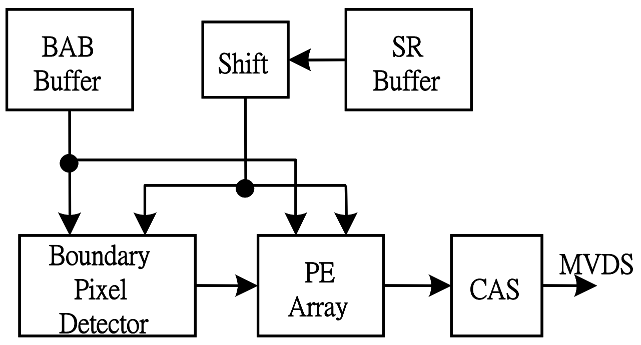


Fig. 11. Block diagram of the proposed BME architecture.

A. Boundary Pixel Detector

Boundary Pixel Detector is a design dedicated to our CCBME algorithm. From the analysis in Section 4, since the search range has been reduced to ±13, the search points with one dimension is 13+13+1=27 and induce a detected region with 27×27 search points. Including the BAB size of 16×16, totally 42×42 pixels are used as the total search area.

According to the effect on 27×27 search points, a multiple of 3 is considered on architecture design. In our binary motion estimator, a 3×3 PE array is used. Therefore, we separate the search positions into a 9×9 sub-search-block (SSB) array structure, and each SSB contains 3×3 search positions for the calculation of PE array. Fig.12 illustrates the detected region and its related array structure with totally 9x9= 81 SSBs.

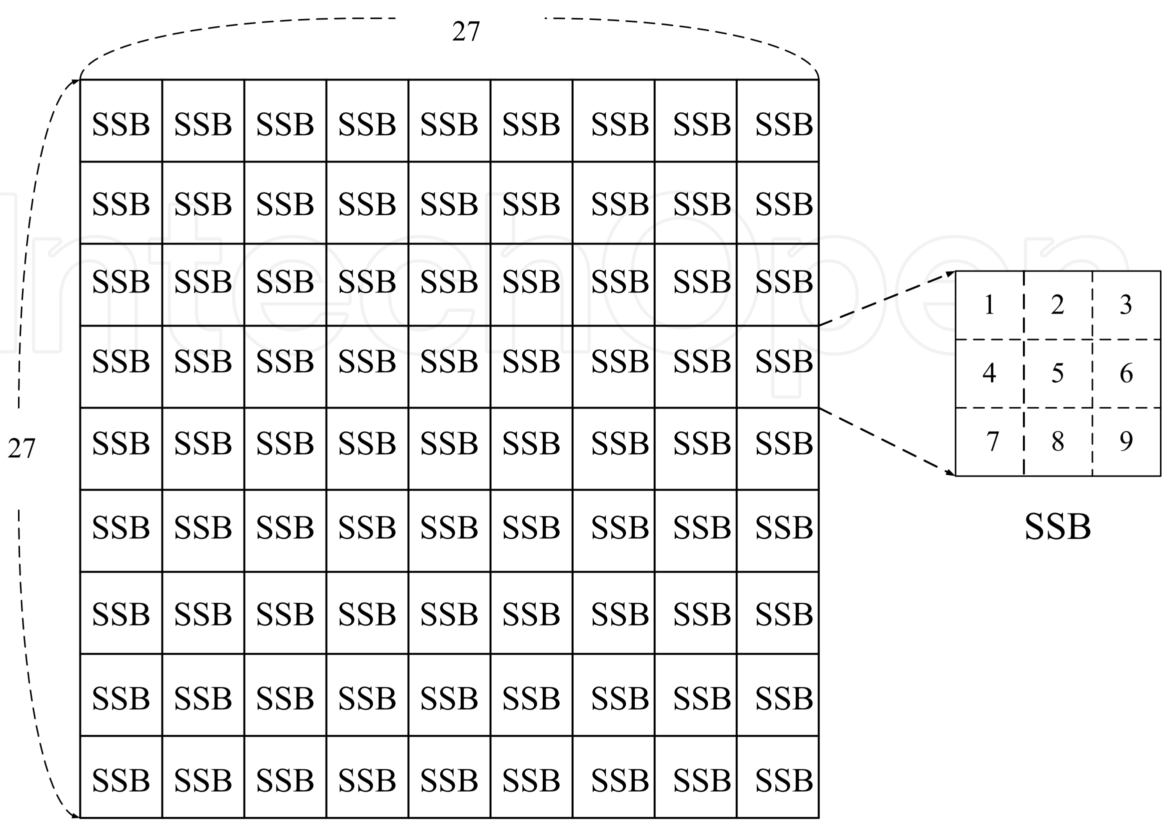


Fig. 12. The 27x27 detected region composes of 9x9=81 SSBs; each SSB contains 9 checking pixel.

In each SSB, 9 checking pixels are included. In addition, in order to detect boundary pixels and obtain the bordered MC-BAB, a border with the width of one pixel around the 42×42 search area, called bordered search area, is applied. As depicted in Fig. 13, the pixels in the grey area are the border pixels. This range indicates the total needed pixels in memory.
Boundary Pixel Detector first selects a boundary pixel *(i,j)* as "reference point" in 16×16 current BAB. Based on the reference point, a 27×27 detected region is generated as shown in Fig. 14. Each pixel in detected region, called checking pixel, denotes whether the correlative search position is non-skipped search position or not. If the checking pixel belongs to a boundary pixel, the correlative search position is denoted as non-skipped search position. Hereafter, Boundary Pixel Detector detects 9 checking pixels, which are relative to the coding SSB in detected region. As shown in Fig. 14, the pixels in the dark area are used for detecting boundary pixel. If all of the checking pixels in light grey area are not boundary pixels, it means that all search positions in the SSB are skipped search positions. Therefore, the coding SSB will not be processed in PE Array. Otherwise, the PE Array calculates WSAD of these 9 search positions in coding SSB.

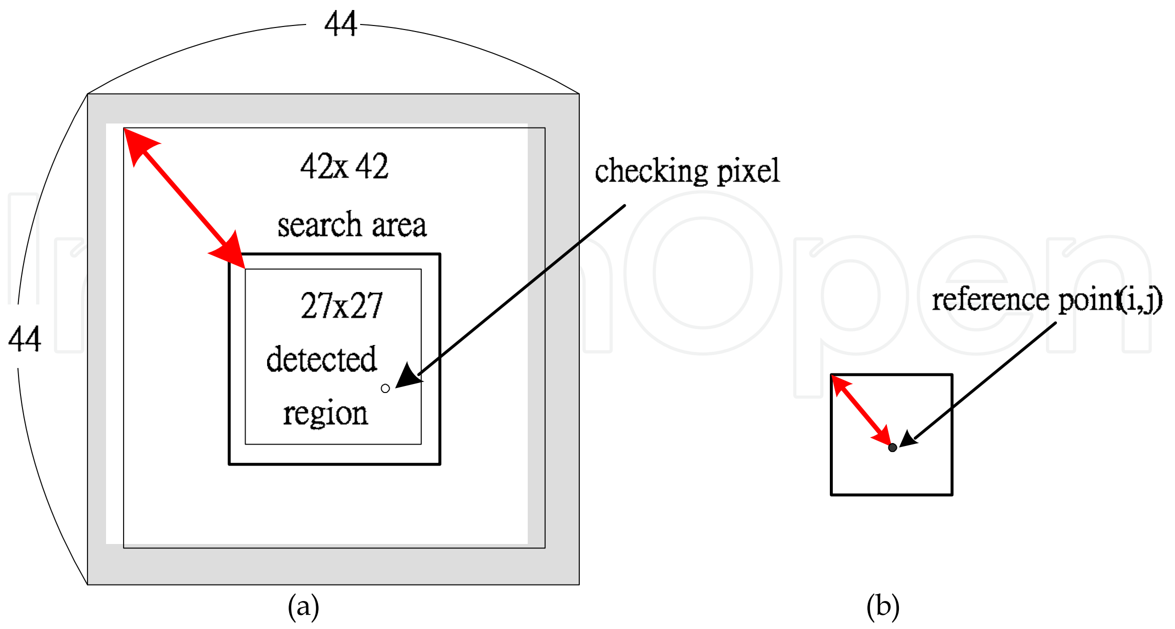


Fig. 13. (a) 44×44 bordered search area, (b) 16×16 current BAB.

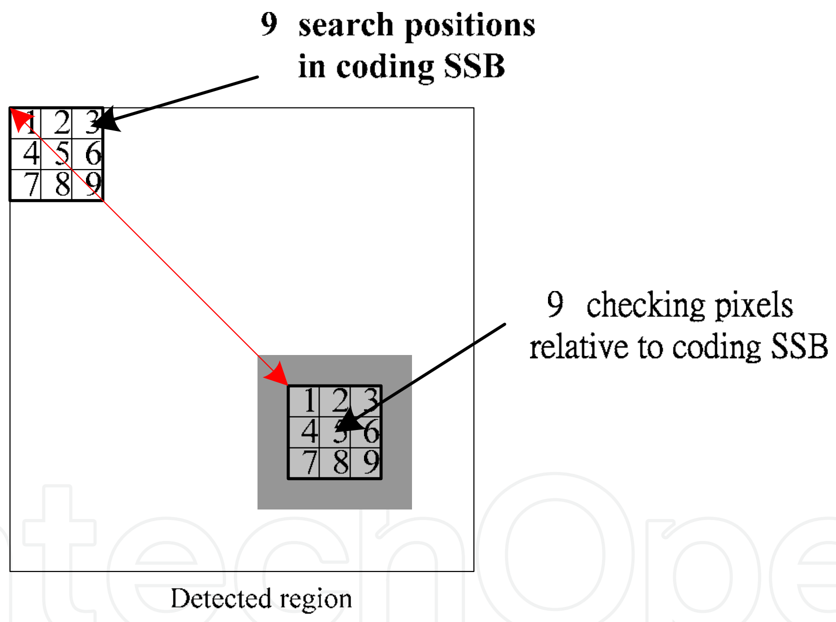


Fig. 14. The relation between search positions and detected region.

B. PE Array and CAS

PE array architecture which performs the binary motion estimation algorithm is shown in Fig. 15(a). The array is the 3×3 architecture and totally consists of 9 PEs. In this architecture, 18-bits reference data (denoted as *Ref[17:0]*) are read from SR buffer, and 16-bits current data are read from BAB buffer. The reference data are broadcasted to all PEs (*Ref[17:2]* for PE1, PE4 and PE7; *Ref[16:1]* for PE2, PE5 and PE8; *Ref[15:0]* for PE3, PE6 and PE9 respectively), while the current data is delayed and fed to the corresponding PE. Each PE calculates the WSAD of a search position in coding SSB. It is noted that only the non-skipped SSB, which is detected from Boundary Pixel Detector, is processed in PE Array.

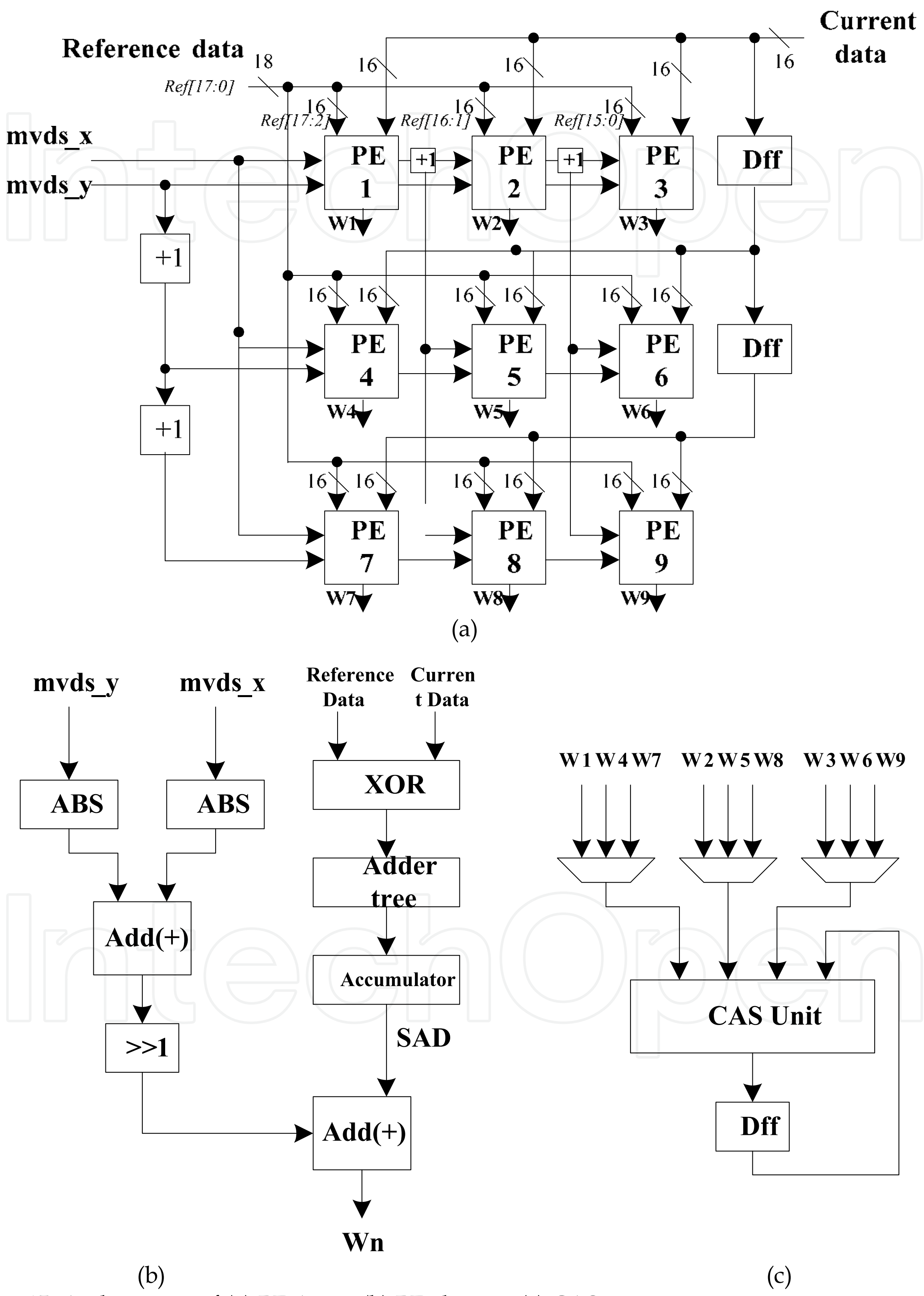


Fig. 15. Architecture of (a) PE Array (b) PE element (c) CAS unit.

The weighted data is calculated by adding absolute MVDS in both horizontal and vertical directions and shifting right one bit. From the analysis in our previous paper [9], the ratio for *V2/V1* will not make large difference from 0.5 to 0.8. And in that range our WSAD is indeed better than the result for SAD. For reducing the computational complexity, *V1* and *V2* in (1) are determined as 1 and 0.5, respectively. The architecture of PE element is shown in Fig. 15(b). It produces WSAD with the sum of weighted data and SAD, where *Wn* means the WSAD value for each PE element from *n*=1 to 9.
The architecture of Compare and Selection (CAS) module is shown in Fig. 15(c). It finds the smallest WSAD and its MVS in coding SSB and feedbacks the smallest WSAD as the input for the next SSB.

5.2 Size Conversion

In MPEG-4 shape coding, rate control and rate reduction are realized through size conversion. Fig. 16 shows the block diagram of size conversion module.

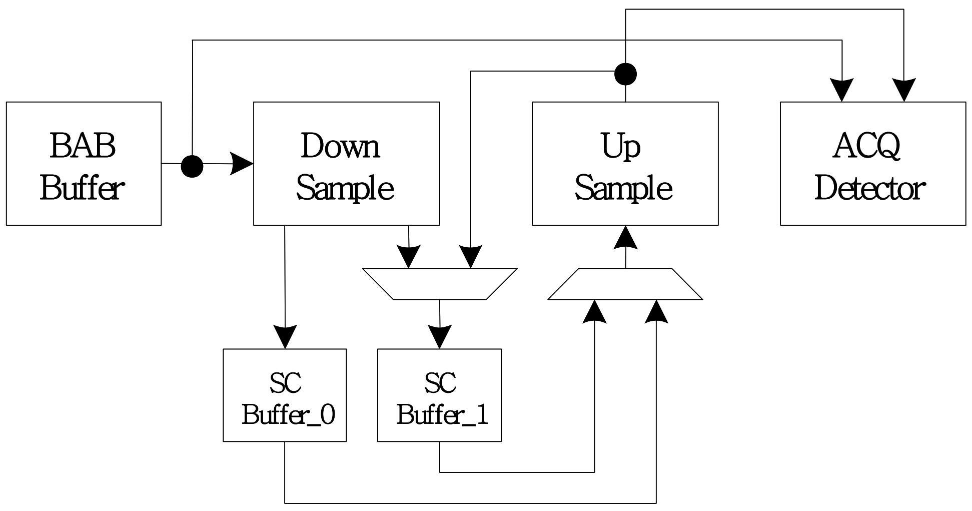


Fig. 16. Block diagram of size conversion module.

It consists of three major units: down-sample, up-sample and accepted quality (ACQ) detector. The bordered BAB is read from BAB buffer and down-sampled to 4×4 and 8×8 BAB in "SC Buffer_0" and "SC Buffer_1" respectively. Then, the 4×4 BAB is up-sampled to SC Buffer_1, and 8×8 BAB is up-sampled to ACQ detector by up-sample unit. ACQ detector calculates the conversion error between the original BAB and the BAB which is down-sampled and reconstructed by up-sample unit. ACQ also needs to determine the conversion ratio. In down-sample procedure, several pixels are down-sampled to one pixel, while interpolated pixels are produced between original pixels in up-sample procedure. To compute the value of the interpolated pixel, a border with the width of two around the current BAB is used to obtain the neighboring pixels (A~L), and the unknown pixels are extended from the outermost pixels inside the BAB. The template and pixel relationship used for up-sampling operation can be referred as in MPEG-4 standard and [15].
Since the implementation of the down-sample and ACQ detector is relatively simple, we only address the design of up-sample unit here. The block diagram of up-sample unit is

shown in Fig. 17. Due to the window-like slicing operations, the up-sample can be easily mapped into a delay line model. A delay line model is used to obtain the pixels in A~L, which determine interpolated pixels (P1~P4). Based on the pixels in A~L, four 4-bits threshold values are obtained from "Table CF". "UP_PE" generates corresponding values for comparison. After comparison between threshold values and the values from "UP_PE", four interpolated pixels (P1~P4) are stored in "Shift Register" and outputted later.

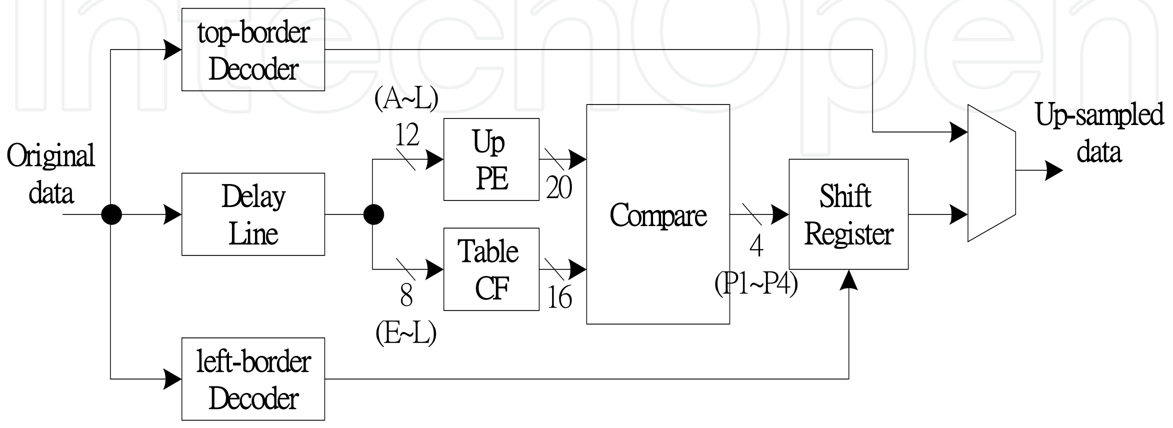


Fig. 17. Block diagram of up-sample unit.

5.3 Context Based Arithmetic Encoder (CAE)

CAE architecture mainly comprises the context generation unit and the binary arithmetic coder [18]-[19]. Fig. 18(a) shows the block diagram of CAE module for our design.

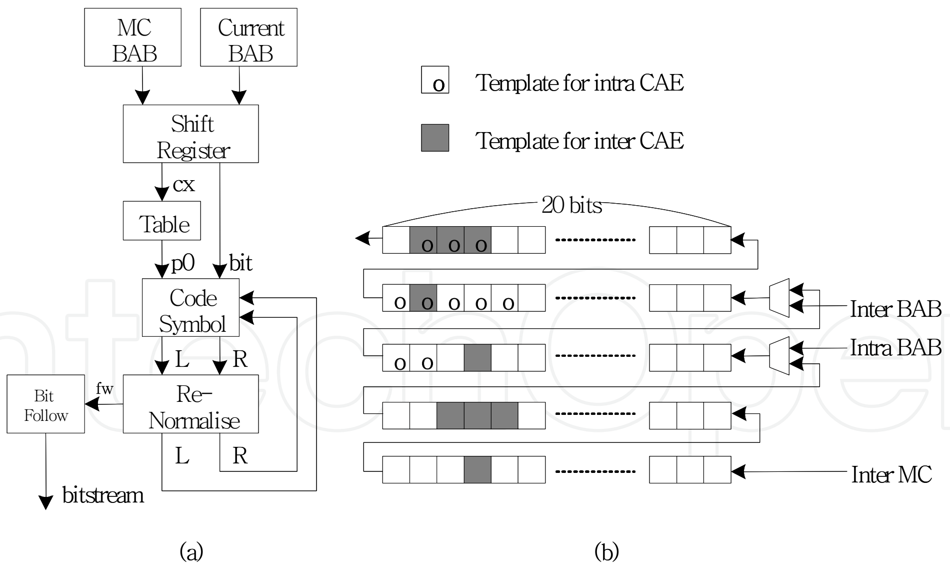
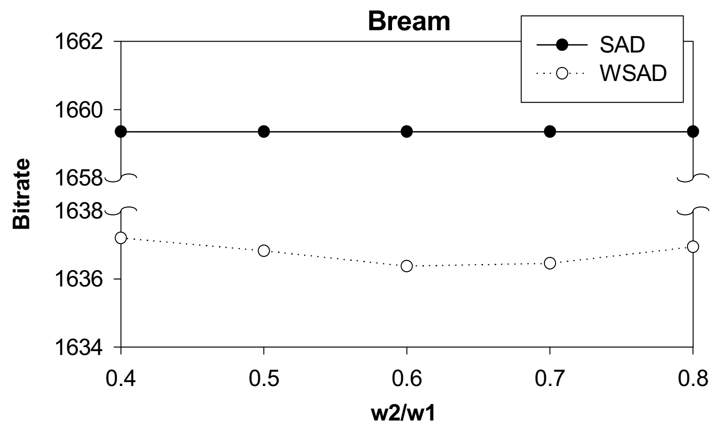


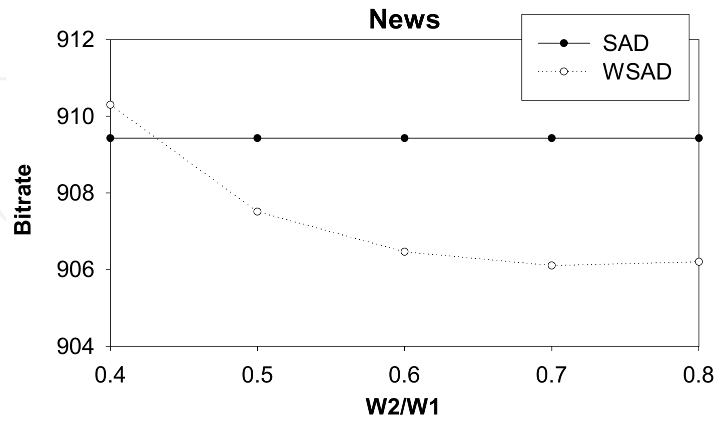
Fig. 18. (a) Block diagram of CAE module. (b). Illustration of the Shift Register.

As mentioned before, for pixel-by-pixel processing in CAE, it basically uses the raster scan order. Since most of the execution time is spent on the context generation in the CAE, the "Shift Register" is used to obtain context and the related operation is illustrated in Fig. 18(b) [20]. Data in the shift registers can be effectively reused and thus this redundant data accesses can be removed. In Fig. 18(b) all the rectangles are represented as registers. Pixels in current BAB and MC-BAB are first loaded into the Shift Register, and then shifted left one bit at every clock cycle. Registers in context box are arranged such that various contexts can be achieved. For intra-CAE mode, the first three rows of Shift Register are used to store current BAB. The first two rows of Shift Register are used to store current BAB and the last three rows are used to store MC-BAB in inter-CAE mode. Therefore, the context (*cx*) and coded bit (*bit*) are obtained from Shift Register per cycle.

6. Implementation Results and Comparisons

We use three MPEG-4 test video sequences of CIF (352×288) format for experiment: Bream, News and Foreman. The three sequences characterize a variety of spatial and motion activities. Each sequence consists of 300 VOP's of arbitrary shape. A search range of ±16 pixels is used and frame-based lossless shape coding is performed for all the test sequence.
The comparisons of SAD and WSAD using full search algorithm are shown in Fig. 19. In this figure, the correlations between bit-rate and the ratio of *W2* to *W1* (*W2*/*W1*) are also shown in Fig. 19.





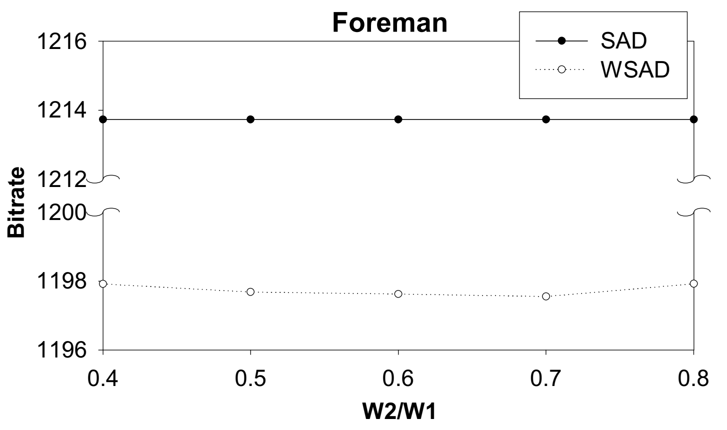


Fig. 19. Performance comparisons of SAD and WSAD using full search algorithm.

It can be seen that the result of using WSAD as the distortion measure takes less bits than that of using SAD except “News” in *W2*/*W1*= 0.4. This is because the “News” sequence is a low motion video sequence, and WSAD is of no benefit in such sequence. As the result of Fig. 19, *W1* and *W2*, which is derived from (1), are determined as 10 and 7, respectively. Based on the weighting values, the average number of bits to represent the shape per VOP is shown in Table 2. The percentages compared to the results of full search, which uses SAD as the distortion measure, are also shown in Table 2. An important contribution is that the WSAD makes some improvement on bit-rate, and it can compensate for the inaccuracy of motion vector when the fast search algorithm is used.

Sequence	Full Search with SAD		Full Search with WSAD	
	Bits/VOP	%	Bits/VOP	%
Bream	1659.35	100	1636.46	98.62
News	909.43	100	906.10	99.63
Foreman	1213.73	100	1197.55	98.67

Table 2. Performance comparison of SAD and WSAD based on full search algorithm in bit-rate (*W1*=10, *W2*=7).

Fig. 20 shows the comparison of various search algorithms. It can be seen that the proposed BS and DBS algorithm take less search points than FS algorithm and the algorithms described in 0-0.

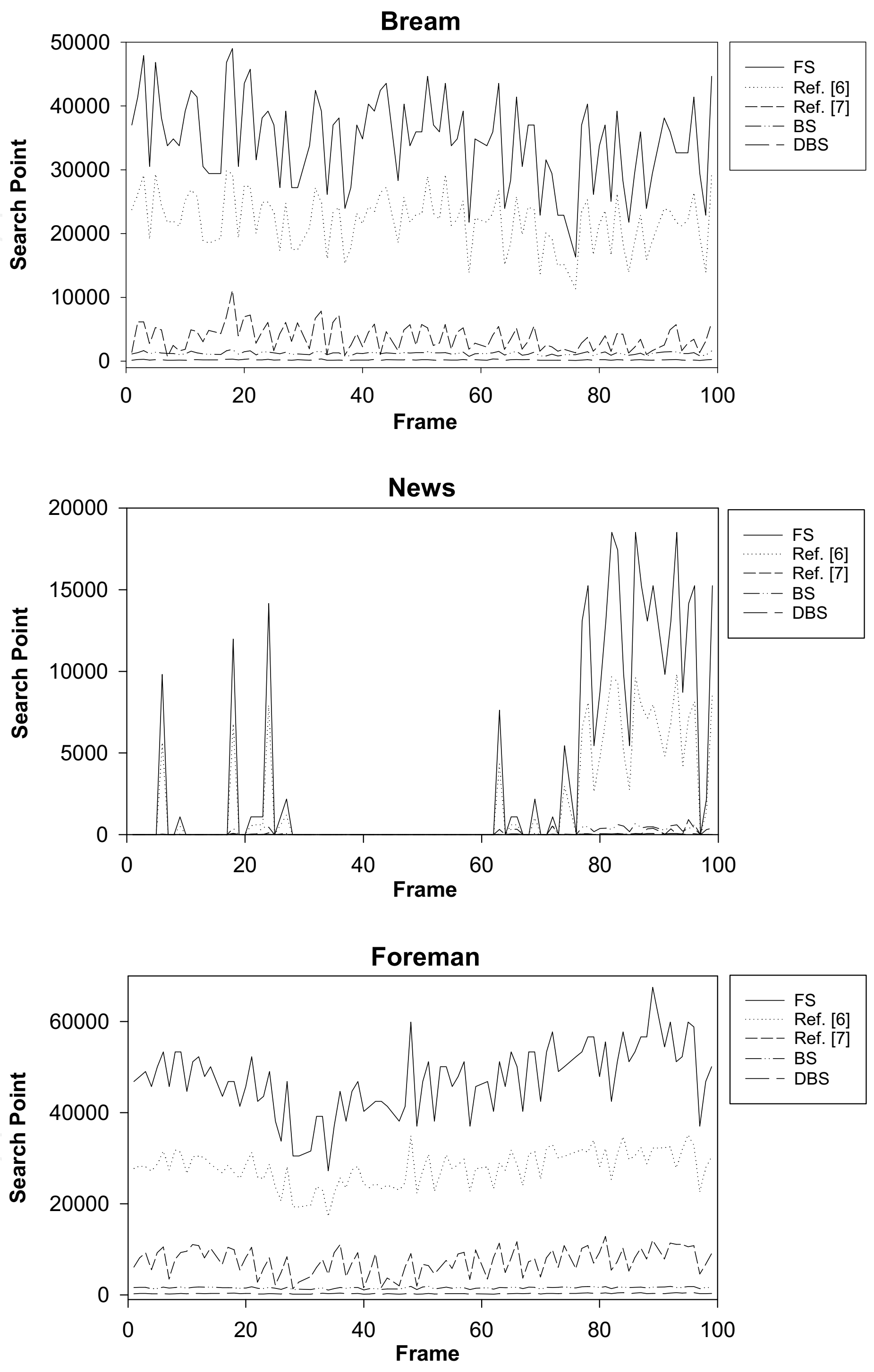


Fig. 20. Performance comparisons of various search algorithms.

Table 3 shows the number of search points (SP), and Table 4 shows the average number of bits to represent the shape per VOP. The percentages compared to the results corresponding to the full search in MPEG-4 VM are also shown in these tables. “BS” denotes the proposed algorithm without using diamond search pattern and “DBS” denotes the proposed algorithm using diamond search pattern. Table 5 shows the runtime simulation results of various BME algorithms. It is noted that the BME algorithm 0 takes much more computational complexity because of the generation of the mask for effective search area.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	SP	SP	%	SP	%	SP	%	SP	%
Bream	10,504,494	6,495,838	61.84	1,560,221	14.85	367,115	3.49	67,232	0.64
News	747,054	403,131	53.96	6,568	0.88	24,923	3.36	2,048	0.27
Foreman	9,085,527	5,093,409	56.06	1,564,551	17.22	287,272	3.16	57,214	0.63

Table 3. Total search points for various search algorithms.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	Bits/VOP	Bits/VOP	%	Bits/VOP	%	Bits/VOP	%	Bits/VOP	%
Bream	1659.35	1659.35	100.00	1683.28	101.44	1655.78	99.78	1669.58	100.62
News	909.43	909.43	100.00	902.68	99.26	910.82	100.15	908.39	99.89
Foreman	1213.73	1213.73	100.00	1219.47	100.47	1209.35	99.64	1219.23	100.45

Table 4. Average bit-rate for various search algorithms.

Compared with the full search method, the proposed fast BME algorithm (BS) needs 3.5% search points and takes equal bit rate in the same quality. By using the diamond-shaped zones, the proposed algorithm (DBS) needs only 0.6% search points. Compared with other fast BME 0-0, our algorithm uses less search points, especially in high motion video sequences, such as 'Bream' and 'Foreman'.

Sequence	Full Search	Ref. 0		Ref. 0		Proposed (BS)		Proposed (DBS)	
	ms	ms	%	ms	%	ms	%	ms	%
Bream	57718.67	62776.36	108.76	7782.83	13.48	8678.02	15.04	1738.24	3.01
News	3830.13	4202.36	109.72	35.05	0.92	574.04	14.99	22.16	0.58
Foreman	44877.19	52574.24	117.15	7507.74	16.73	6487.52	14.46	1703.88	3.80

Table 5. Runtime simulation results of various search algorithms.

Table 6 shows the number of non-skipped SSB per PE. Due to the contribution of the proposed CBBME algorithm, the number of non-skipped SSB is reduced largely. It can be seen that the average number of non-skipped SSB is much less than the total number of SSB, which is denoted as 81 per PE. In the worst case, the additional non-skipped SSB is usually in the positions with large motion vector, which does not tend to be the adoptive MV.

Sequence	Average non-skipped SSB per PE	Maximum non-skipped SSB per PE
Bream	12.78	33
News	10.73	16
Foreman	10.94	24
Children	13.40	46

Table 6. Average and maximum number of non-skipped SSB per PE.

Therefore, in the binary motion estimator, the number of non-skipped SSB is limited to 32 from experimentation in maximum situation. For average situation, we set the number as 12.In our architecture, the processing cycles for various BAB types are shown in Fig. 21.

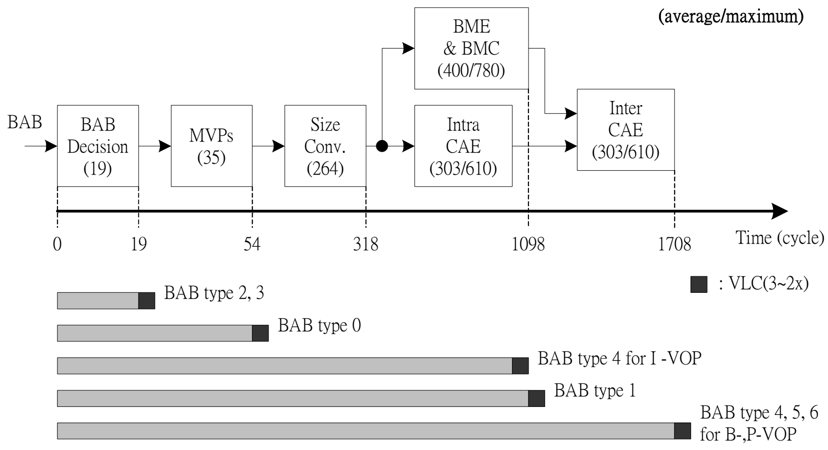


Fig. 21. Processing cycles for various BAB types.

Totally seven types of mode are described for BAB. The number in parentheses indicates the latency of that module with average and maximum number of cycles. Notice that the processing cycles of BME and CAE depend on the content of BAB. To complete one BAB processing in the worst case scenario, our architecture requires 1708 clock cycles, including: 19 clock cycles for mode decision, 35 clock cycles for identifying MVPS, 264 clock cycles for size conversion, 780 clock cycles for BME, and 610 clock cycles for inter CAE.
Actually, few literatures explored the architecture design of shape coding. In [10], they only designed the BME. We can extract our data for BME part as comparison in Table 7. In terms of the whole shape coding,

	E. A. Al_Qaralleh's [10]	Proposed
Gate count	11582	10523
One BAB processing (cycles)	563	780
Comment	Partial design (Only BME implemented)	Completed design

Table 7. Comparison with BME only.

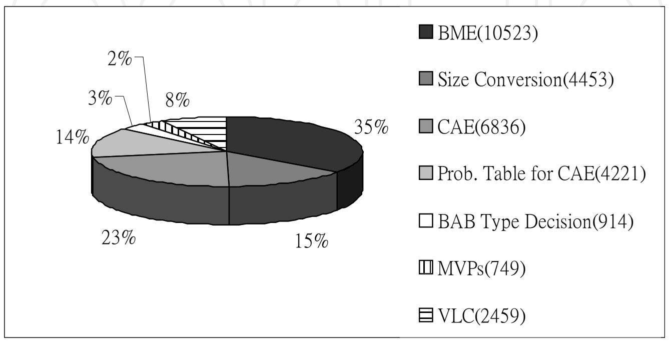
Table 8 illustrates some results with different architectures. For our design the size of BAB buffer and SR buffer are 16×16 and 44×44 respectively. The average and maximum numbers of non-skipped SSB are determined as 12 and 32 from experiments.

	DDBME [16]	Natarajan's [21]	Proposed
Current BAB size	16×16 bits RAM	16×16 bits SRAM	16×16 bits SRAM
SR buffer size	16×32 bits RAM	47×32 bits SRAM	44×44 bits SRAM
Access from SR buff to obtain one MV (Bytes)	4096	5828	Average: 1348 Maximum: 3328
Latency to obtain one MV (cycles)	1039	1039	Average: 360 Maximum: 740
One BAB processing (cycles)	3034 (without pipelinig)	N/A	1708 (without pipelining)

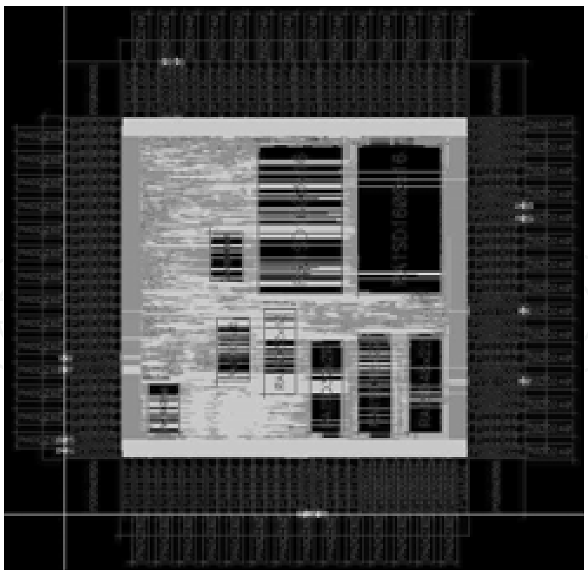
Table 8. Architecture analysis and comparison for various binary motion estimations.

Table 8 also lists the architecture comparisons between the proposed and some previous works in [16] and [21]. In [16] it adopts the data-dispatch technique and is named as data-dispatch based BME (DDBME). [21] is Natarajan's architecture which is modified from BME-based Yang's 1-D systolic array [22]. In their design, they use extra memory, SAP module, to process the bit shifting and bit packing for the alignment of BAB. It also results in a computation overhead. In our design, we have used the Boundary Pixel Detector for the alignment of boundary of BAB. Accordingly, no SAP memory is needed. Furthermore, the proposed CBBME design needs less data transfer and latency to obtain one motion vector compared with [16] and [21], because we consider the skipping on redundant searches. Compared with the implementation for one BAB processing in the worst case, our design also requires less cycles than [16] with the same base of non-pipelining work. Only 56% cycles of [16] is needed in our approach.

Fig. 22(a) shows the synthesized gate count of each module and Fig. 22(b) shows the chip layout using synthesizable Verilog HDL. There are 7 synchronous RAMs in the chip. Two 1600×16 bits RAMs are used for frame buffer. Two 48×22 bits RAMs are used for SR buffer. One 32×20 bits RAM is used for SC buffer, one 32×18 bits RAM is used for MC buffer and one 32×20 bits RAM is used for BAB buffer, respectively. The chip feature is summarized in Table 9. Total gate count is 40765. The chip size is 2.4×2.4 mm^2 with TSMC 0.18μm CMOS technology and the maximum operation frequency is 53 MHz



(a)



(b)

Fig. 22. (a) Synthesized gate count of each module. (b) Chip layout of shape coding encoder.

Technology	**TSMC 0.18µm CMOS (1P6M)**
Package	**128 CQFP**
Die size	**2.4×2.4 mm ^{2}**
Core size	**1.4×1.4 mm ^{2}**
Clock rate	**53 MHz**
Power dissipation	**35mW**
Gate count	**40765**
Memory size (bits)	**Frame buffer: 2×1600×16** **SR buffer: 2×48×22** **SC buffer: 32×20** **MC buffer: 32×18** **BAB buffer: 32×20**

Table 9. Chip Features.

7. Conclusion

MPEG-4 has provided a well-adopted object-based coding technique. When people migrate from compressed coding domain to object coding domain, the complexity issue on shape coding is converged. In this paper we propose a fast binary motion estimation algorithm using diamond search pattern for shape coding and an efficient architecture for MPEG-4 shape coding. By using the properties of shape information and diamond shaped zones, we can reduce the number of search points significantly, resulting in a proportional reduction of computational complexity. The experimental results show that the proposed method can reduce the number of search points of BME for shape coding to only 0.6% compared with that of the full search method described in MPEG-4 verification model. Specifically, the fast algorithm takes equal bit rate in the same quality compared with full search algorithm. The proposed algorithm is simple, efficient and suitable for real-time software and hardware applications. This architecture is based on the boundary search fast algorithm which accomplishes the large reduction on computation complexity. We also apply the approaches on center-biased motion vector distribution and search range shrinking for further improvement. In this paper we report a comprehensive exploration on each module of shape coding encoder. Our architecture completely elaborates the advantages of the proposed fast algorithm with a high performance and regular architecture. The result shows that our design can reduce the memory access and processing cycles largely. The average number of clock cycles for one binary alpha block processing is only 1708, which is far less than other designs. The system architecture is implemented by synthesizable Verilog HDL with TSMC 0.18μm CMOS technology. The chip size is 2.4 × 2.4 mm^2 and the maximum operation frequency is 53 MHz.

8. Acknowledgements

This work was supported by the CIC and the National Science Council of Taiwan, R.O.C. under Grant *NSC97-2220-E-008-001*.

9. References

B. Natarajan, V. Bhaskaran, and K. Konstantinides, “Low-complexity block-based motion estimation via one-bit trasforms,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 702 -707, Aug. 1997.

D. Yu, S. K. Jang, and J. B. Ra, “A fast motion estimation algorithm for MPEG-4 shape coding,” *IEEE Int. Conf. Image Processing*, vol. 1, pp. 876-879, 2000.

E. A. Al_Qaralleh, T. S. Chang, and K. B. Lee, “An efficient binary motion estimation algorithm and its architecture for MPEG-4 shape encoding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 17, pp. 859-868, Jul. 2006.

G. Feygin, P. Glenn and P. Chow, “Architectural advances in the VLSI implementation of arithmetic coding for binary image compression,” *Proc. Data Compression Conf. (DCC’94)*, pp. 254 -263, 1994.

G. Sullivan and T. Wiegand, “Rate-Distortion optimization for video compression”, *IEEE Signal Processing Magazine*, pp. 74-90, Nov. 1998.

H. C. Chang, Y. C. Chang, Y. C. Wang, W. M. Chao, and L. G. Chen, “VLSI architecture design for MPEG-4 shape coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 741-751, Sep. 2002.

ISO/IEC 13818-2, “Information technology-generic coding of moving pictures and associated audio information-Video,” 1994.

ISO/IEC JTC1/SC29/WG11 N 2502a, “Generic coding of audio-visual objects: Visual 14492-2,” Atlantic City Final Draft IS, Dec. 1998.

ISO/IEC JTC1/SC29/WG11 N 3908, “MPEG-4 video verification model version 18.0,” Jan. 2001.

J. L. Mitchell and W. B. Pennebaker, "Optimal hardware and software arithmetic coding procedures for the Q-Coder," *IBM J. Res. Devel.*, vol. 32, pp. 717-726, Nov. 1998.

Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, and Ashraf Ali Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 169-177, Aug. 1998.

K. B. Lee, J. Y. Lin and C. W. Jen, "A Multisymbol Context-Based Arithmetic Coding Architecture for MPEG-4 Shape Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 283-295, Feb. 2005.

K. B. Lee, J. Y. Lin, and C. W. Jen, "A fast dual symbol context-based arithmetic coding for MPEG-4 shape coding," *IEEE International Symposium on Circuits and Systems (ISCAS2004)*, pp. 317-320, 2004.

K. M. Yang, M. T. Sun, and L. Wu, "A family of VLSI design for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 36, pp. 1317 -1325, Oct. 1989.

K. Panusopone, and X. Chen, "A fast motion estimation method for MPEG-4 arbitrarily shaped objects," *IEEE Int. Conf. Image Processing*, vol. 3, pp. 624-627, 2000.

L. K. Liu, and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 419-422, Aug. 1996.

M. Tourapis, O. C. Au, and M. L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 934-947, Oct. 2002.

N. Brady, "MPEG-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1170-1189, Dec. 1999.

Recommendation H.263: Video coding for low bit-rate communication, ITU-T H.263, 1998.

S. Dutta and W. Wolf, "A flexible parallel architecture adapted to block-matching motion estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no 1, pp. 74-86, Feb. 1996.

S. H. Han, S. W. Kwon, T. Y. Lee, and M. K. Lee, "Low power motion estimation algorithm based on temporal correlation and its architecture," *IEEE Internal Symposium on Signal Processing and its Applications (ISSPA)*, vol. 2, pp.647-650, Aug. 2001.

T. H. Tsai and C. P. Chen, "A fast binary motion estimation algorithm for MPEG-4 shape coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 908-913, Jun. 2004.

T. Komarek and P. Pirsch, "Array architectures for block-matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, pp.1301-1308, Oct. 1989.

T. M. Liu, B. J. Shieh and C. Y. Lee, "An efficient modeling codec architecture for binary shape coding," *IEEE International Symposium on Circuits and Systems (ISCAS2002)*, vol. 2, pp. 316 -319, 2002.

VLSI
Edited by Zhongfeng Wang

ISBN 978-953-307-049-0
Hard cover, 456 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The process of Integrated Circuits (IC) started its era of VLSI (Very Large Scale Integration) in 1970's when thousands of transistors were integrated into one single chip. Nowadays we are able to integrate more than a billion transistors on a single chip. However, the term "VLSI" is still being used, though there was some effort to coin a new term ULSI (Ultra-Large Scale Integration) for fine distinctions many years ago. VLSI technology has brought tremendous benefits to our everyday life since its occurrence. VLSI circuits are used everywhere, real applications include microprocessors in a personal computer or workstation, chips in a graphic card, digital camera or camcorder, chips in a cell phone or a portable computing device, and embedded processors in an automobile, et al. VLSI covers many phases of design and fabrication of integrated circuits. For a commercial chip design, it involves system definition, VLSI architecture design and optimization, RTL (register transfer language) coding, (pre- and post-synthesis) simulation and verification, synthesis, place and route, timing analyses and timing closure, and multi-step semiconductor device fabrication including wafer processing, die preparation, IC packaging and testing, et al. As the process technology scales down, hundreds or even thousands of millions of transistors are integrated into one single chip. Hence, more and more complicated systems can be integrated into a single chip, the so-called System-on-chip (SoC), which brings to VLSI engineers ever increasingly challenges to master techniques in various phases of VLSI design. For modern SoC design, practical applications are usually speed hungry. For instance, Ethernet standard has evolved from 10Mbps to 10Gbps. Now the specification for 100Mbps Ethernet is on the way. On the other hand, with the popularity of wireless and portable computing devices, low power consumption has become extremely critical. To meet these contradicting requirements, VLSI designers have to perform optimizations at all levels of design. This book is intended to cover a wide range of VLSI design topics. The book can be roughly partitioned into four parts. Part I is mainly focused on algorithmic level and architectural level VLSI design and optimization for image and video signal processing systems. Part II addresses VLSI design optimizations for cryptography and error correction coding. Part III discusses general SoC design techniques as well as other application-specific VLSI design optimizations. The last part will cover generic nano-scale circuit-level design techniques.