IEICE TRANS. FUNDAMENTALS, VOL.E91–A, NO.12 DECEMBER 2008

PAPER Special Section on VLSI Design and CAD Algorithms

Wide-Range Motion Estimation Architecture with Dual Search Windows for High Resolution Video Coding

Lan-Rong DUNG[†a)], *Member and* **Meng-Chun LIN**[†], *Nonmember*

SUMMARY This paper presents a memory-efficient motion estimation (ME) technique for high-resolution video compression. The main objective is to reduce the external memory access, especially for limited local memory resource. The reduction of memory access can successfully save the notorious power consumption. The key to reduce the memory accesses is based on center-biased algorithm in that the center-biased algorithm performs the motion vector (MV) searching with the minimum search data. While considering the data reusability, the proposed dual-search-windowing (DSW) approaches use the secondary windowing as an option per searching necessity. By doing so, the loading of search windows can be alleviated and hence reduce the required external memory bandwidth. The proposed techniques can save up to 81% of external memory bandwidth and require only 135 MBytes/sec, while the quality degradation is less than 0.2 dB for 720 p HDTV clips coded at 8 Mbits/sec.

key words: *motion estimation, MPEG, video compression, bandwidth*

1. Introduction

Motion estimation (ME) has been notably recognized as the most critical part of video compression, such as MPEG standards and H.26x [1]–[5]. It tends to dominate the computational and hence power requirements. As the demand for high-resolution, high-quality video system increases, the implementation of motion estimation is becoming more costly and power-consuming. Among the hardware components of motion estimation, the on-chip memory is the one that dominates power consumption and cost. Because the on-chip memory size is too small to store a high-resolution frame, there exists a tradeoff between the external memory bandwidth and on-chip memory size. The less the on-chip memory is used in motion estimation, the higher the external memory bandwidth is required. There are three factors that affect the tradeoffs: the data reuse mechanism, the size of search window, and the efficiency of external memory access. The first two factors can be exploited at the architecture level while the last can be improved in the DRAM controller.

In the past decade, various algorithms have been proposed to improve the performance of ME in terms of compression ratio and computational cost; however, very few works present solutions for data reusability while analyzing the required external memory bandwidth. Paper [6] is the one. [6] defines data reuse levels for an Full-Search Block-Matching (FSBM) ME architecture to minimize the external memory bandwidth. The FSBM algorithm with the Sum of Absolute Difference (SAD) is the most popular criterion for motion estimation because of its considerably good quality [23], [35]–[37]. It is particularly attractive to those who require extremely high quality. However, the full search algorithm needs high computational load and large memory size which are a major problem in the implementation of motion estimation.

To reduce the computational complexity of FSBM, researchers have proposed various fast block-matching algorithms (FBMAs), by either reducing the number of search steps [15]–[21] or simplifying the calculation of error criterion [22]–[31]. We categorize the former as the center-biased algorithms, and the latter as the criterion-simplifying algorithms. By combining step-reduction and criterion-simplifying, some researchers proposed two-phase algorithms to balance the performance between complexity and quality [32]–[34].

It has been shown that these fast algorithms can significantly reduce the computational load with little quality degradation. The center-biased algorithms are good for reducing the external memory bandwidth, while the center-biased algorithms, which are motivated by statistical observation show that most of motion vectors are centered around (0,0) and, hence, only a small portion of the search window needs to be accessed most of the time. For high-resolution applications, this nice feature can help us reduce the external memory bandwidth and the local memory requirement.

This paper presents a new windowing technique, called dual-search-windowing (DSW), for center-biased ME algorithms. The DSW requires smaller on-chip memory than full search-windowing while maintaining high data reusability that significantly reduces the external memory bandwidth requirement. The DSW consists of a primary windowing and a secondary windowing. The primary windowing is necessary for all MV searches and the secondary windowing is only called for when needed. The primary windowing is sliding with the macro-block (MB) changing, so each move only requires an update of a single slice. This leads to a high degree of reusability. When the center-biased algorithm moves outside the primary window, the secondary window will be loaded. Although the secondary window is not be reused for its occasional occurrence, thanks to the center-biased algorithm, the secondary windowing is seldom needed and the impact on external memory bandwidth requirement is low. For 720 p HDTV clips, the proposed

Manuscript received March 24, 2008.
Manuscript revised June 24, 2008.

[†]The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 30010, Taiwan.
a) E-mail: lennon@faculty.nctu.edu.tw

DOI: 10.1093/ietfec/e91–a.12.3638

techniques save up to 81.41% of external memory bandwidth and require only 135.20 MBytes/sec, with less than 0.2 dB quality degradation for video coded at 8 Mbits/sec.

The paper is organized as follows. Section 2 introduces the primary windowing techniques and exploits their reusability. Section 3 describes the proposed DSW algorithms with comparisons. Section 4 describes the experimental results and the performance analysis in terms of external memory bandwidth, local memory size, and visual quality. Finally, the Sect. 5 concludes the contributions of this work.

2. The Primary Windowing Techniques and Their Reusability

Center-biased ME algorithms are developed based on the observation that most of MVs are located near the center-point of the search window. In this paper, we use diamond search (DS) [11]–[13] and small diamond search (SDS) [7] as the target algorithms; however, the proposed approaches are not limited to these two algorithms. Figure 1 shows the search pattern for DS and SDS algorithms. By simulating with D1 videos, Figs. 2 and 3 illustrate that more than 98% MVs are located within ±32 search range. Hence, we can use ±32 search range for the primary window to save the external memory accesses.

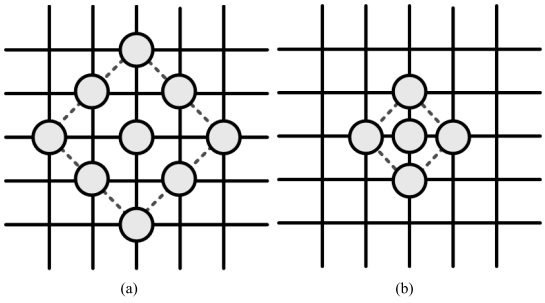


Fig. 1 (a) Large diamond search pattern. (b) Small diamond search pattern.

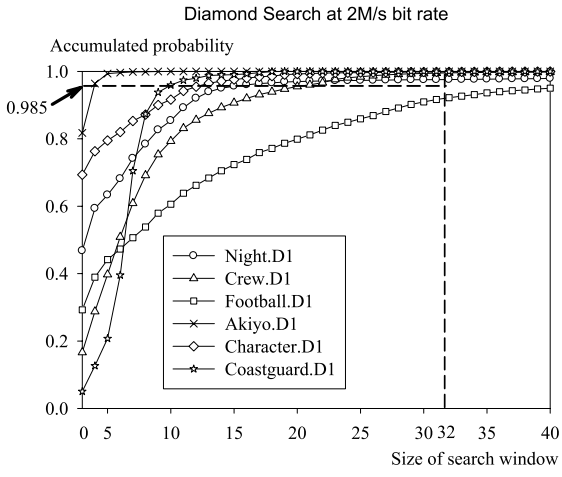


Fig. 2 The analysis of accumulated probability versus size of search window for DS algorithm under the rate control for 2 Mbits/sec.

The primary windowing is used to load a smaller search window for most MV searches. For instance, given a D1 video sequence, the typical search window size is ± 64 and we can choose ± 32 as the primary window. Note that the MB size in MPEG4/AVC is 16×16. Therefore, the local memory size can be ideally reduced by a factor of $81/25$ and Fig. 4(a) shows the windowing technique with a single MB. The bolded box indicates the data in local memory and the centered square is the current MB. For the MB of SW_1, as shown in the upper-left corner, we first load three slices labelled by 3, 4, and 5 while the slices 1 and 2 are the padding data which is generated internally by the ME engine without consuming external memory bandwidth. When the MV search performs for the MB of SW_1, the primary windowing simultaneously loads slice 6 for the next MV search. The following steps of windowing show the parallel operations of MV searches and updates of slices. Comparing with full search windowing, the local memory size is reduced by the factor of $81/30$ (or 2.7) and the external bandwidth requirement can be reduced by a factor of $9/5$. To increase the degree of reusability, one can process more MBs at a time because the data of the primary window can be used more than once for each data loading. However, the penalty is the increase of local memory size. Figures 4(b)–(d) illustrate the other schemes for primary windowing; the symbol $type_p$ is used to label the schemes.

We make a modification on PMV search for schemes of Figs. 4(c) and (d). When the MV of upper right macroblock has not determined yet, we temporarily use the motion vector (0,0) as the MV of upper right macroblock. The modification might result in quality degradation under the constant bit-rate control. Fortunately, the degradation is little for the high occurrence of null MV. As shown in Table 1, the quality degradation is up to 0.16 dB. In video coding community empirically 0.5 dB is considered a threshold below which the perceptual quality difference cannot be perceived by subjects.

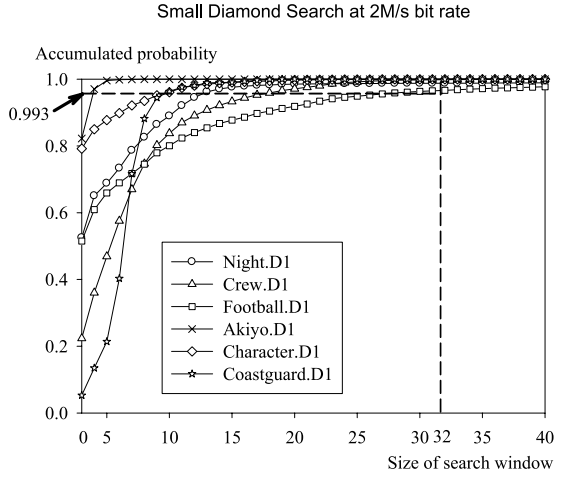


Fig. 3 The analysis of accumulated probability versus size of search window for SDS algorithm under the rate control for 2 Mbits/sec.

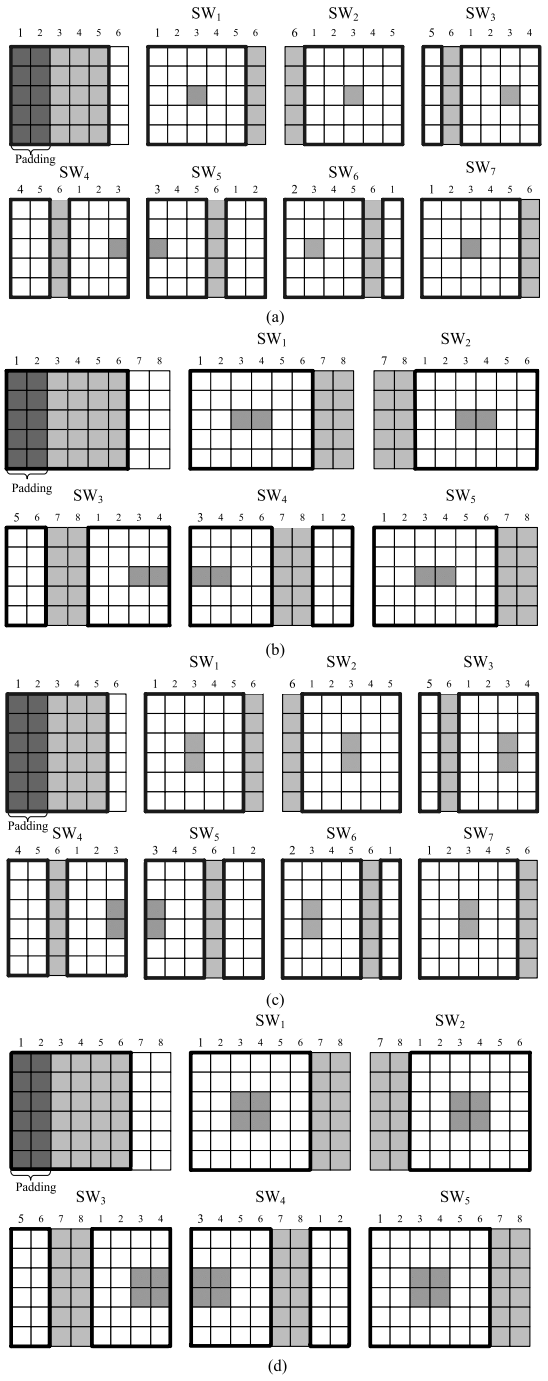


Fig. 4 (a) The primary windowing with single MB ($type_p = 1$). (b) The primary windowing with two horizontal MBs ($type_p = 2$). (c) The primary windowing with two vertical MBs ($type_p = 3$). (d) The primary windowing with four MBs ($type_p = 4$).

According to the windowing scheme, we can use the following equation to calculate the number of pixel accesses for each frame during the primary windowing and evaluate the external memory bandwidth:

$$N_{acess,p} = N_f \times N_1 - 2m_x \times N_y \times \sum_{i=1}^{\frac{(N_1-1)}{2}} i, \quad (1)$$

Table 1 The performance analysis of visual quality versus different PMV accuracy.

DS Algorithm			
Bit-rate	2M bits/s		
Search range	[−64,+64]		
Motion degree	Fast motion		Moderate motion
Video sequences	Night	Football	Crew
	PSNRY	PSNRY	PSNRY
All PMV	33 dB	36.03 dB	35.97 dB
The primary windowing with two vertical MBs ($type_p = 3$)	32.9 dB (−0.1 dB)	35.83 dB (−0.2 dB)	35.86 dB (−0.11 dB)
The primary windowing with four MBs ($type_p = 4$)	32.96 dB (−0.04 dB)	35.99 dB (−0.04 dB)	35.92 dB (−0.05 dB)
SDS Algorithm			
Bit-rate	2M bits/s		
Search range	[−64,+64]		
Motion degree	Fast motion		Moderate motion
Video sequences	Night	Football	Crew
	PSNRY	PSNRY	PSNRY
All PMV	32.84 dB	35.43 dB	35.81 dB
The primary windowing with two vertical MBs ($type_p = 3$)	32.68 dB (−0.16 dB)	35.23 dB (−0.2 dB)	35.65 dB (−0.16 dB)
The primary windowing with four MBs ($type_p = 4$)	32.78 dB (−0.06 dB)	35.33 dB (−0.1 dB)	35.78 dB (−0.03 dB)

where N_f is the frame pixel count, N_1 is the regular access times of each pixel, m_x is number of vertical pixels of target MB set for each primary windowing and N_y is the number of horizontal pixels of each frame. N_1 and m_x are defined as follows.

$$\mathrm{N}_1 = \begin{cases} 5, & for\ type_p = 1, 2. \\ 3, & for\ type_p = 3, 4. \end{cases} \quad (2)$$

$$\mathrm{m}_x = \begin{cases} 16, & for\ type_p = 1, 2. \\ 32, & for\ type_p = 3, 4. \end{cases} \quad (3)$$

3. Dual-Search-Window Motion Estimation Algorithms

Based on the proposed two-step memory access mechanisms, we developed three ME algorithms for SDTV/HDTV applications. The first algorithm, named the fully-expanding dual-search-window algorithm (FEDSW), expands the search range to full search window when the MV search reaches or locates beyond the boundary of the primary window. The FEDSW may have the least quality degradation, but it requires high memory bandwidth for loading the secondary windows to local SRAM. Since the center-biased ME seldom goes too far from the starting point, the secondary window can be set to a smaller size to save the external memory accesses. Hence, we propose

the second algorithm, called the fixed-secondary-window dual-search-window algorithm (FSDSW). The FSDSW limits the size of the secondary window to cut the redundant external memory accesses and save local SRAM size. The range of the secondary window is determined by simulating testcases with full-sized search window. Given a range to cover most MV results, the FSDSW requires low memory bandwidth while the average quality loss is little. Nevertheless, its transient quality loss could be high for some high-motion clips. To deliver a quasi-static video quality, we further proposed the third algorithm to adaptively adjust the range of the secondary window. The third algorithm is called the variable-secondary-window dual-search-window algorithm (VSDSW). The VSDSW can adaptively adjust the size of the secondary window to keep the transient quality loss low and save unnecessary memory accesses. The following gives descriptions of the proposed algorithms.

3.1 The Fully-Expanding Dual-Search-Window Algorithm (FEDSW)

The FEDSW defines the primary window and four extra search windows, as shown in Fig. 5, where (2N+1) × (2N+1), (2P+1) × (2P+1) and (2N+1)/2 × (2N+1)/2 indicate the ranges of total, primary and secondary window, respectively. In Fig. 5, the primary window is at the center of the full search window and the secondary windows are located at four quadrants. During the ME process, the predicted MV (PMV) is first calculated to decide the initial search point. If PMV is located inside of the primary window, the FEDSW performs the MV searching within the primary window. As shown in Fig. 5(a), when both PMV and MV are within the primary window, the secondary window will not be needed. When the searching point reaches the boundary of primary window, the secondary window will be loaded to expand the search range for the right MV, as shown in Fig. 5(b). The secondary window is selected according to in which quadrant the searching point reaches the boundary. If PMV is out of the primary window at the beginning, the MV search will start in the secondary window, as shown in Fig. 5(c).

Although the FEDSW can efficiently decide whether a secondary window is used to find the candidate motion vector or not according to the direction of PMV or position of searching point for each MB, the range of secondary window is still wide-ranging for high resolution video sequences. For example, the range of original search window is [−64, +64] for horizontal and vertical directions, the primary window is [−32, +32] for both ones and the secondary window is quarter of original search window, namely [−32, +32]. The range of secondary window is the same as the one of primary window; however, based on statistical results, the candidate motion vectors of average 98.5% MB and ones of average 99.3% MB can be searched in the primary window ([−32, +32]) by using DS algorithm and SDS algorithm for six testing D1 video sequences respectively and therefore reducing the range of secondary window to efficiently saving memory access from DRAM to SRAM is necessary. To achieve this target, we further propose two optimal methods to find the suitable secondary window, the one is to support a fixed range of secondary window through the statistical analysis and the other can adaptively adjust the range of search window by using the curve fitting skill for different kinds of motion degree video sequences. Therefore, this paper presents two more algorithms for the secondary window; they are FSDSW and VSDSW.

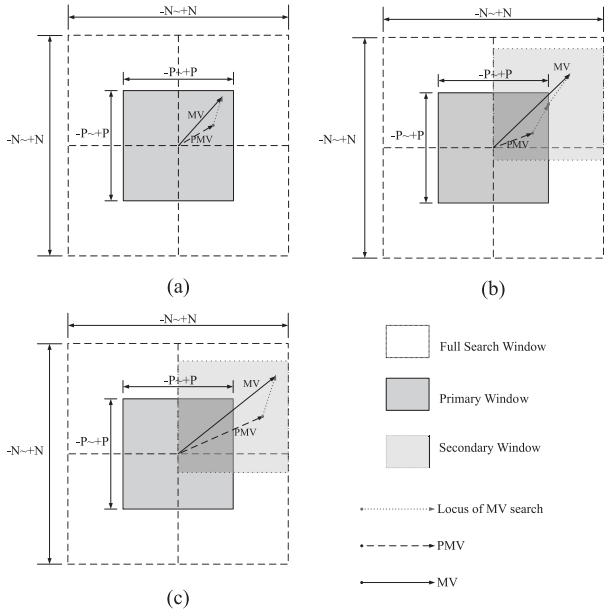


Fig. 5 (a) The windowing strategy for the case when both PMV and MV are in the primary window. (b) The windowing strategy for the case, given the PMV in the primary window, when the mv searching reaches the boundary of the primary window. (c) The windowing strategy for the case when the PMV is out of the primary window.

3.2 The Fixed-Secondary-Window Dual-Search-Window Algorithm (FSDSW)

Figure 6 shows the schemes for FSDSW and VSDSW, where (2S+1)×(2S+1) is the range of secondary window. There are four cases for the secondary windowing. Figure 6(a) shows the first case where the PMV is in the primary window and the motion vector can be reached within the primary window. In this case, the secondary windowing is not needed. However, if the searching point touches the boundary of the primary window, the secondary windowing will be called and the search scheme becomes the second case, as shown in Fig. 6(b). Note that the motion vector will be searched until the searching point reaches the boundary of the secondary window. The third and forth cases occur when the PMV is out of the primary window. In the third case, we perform the secondary windowing at the beginning while the primary window is loaded. Since the primary window and secondary window are not overlapped the MV searching is running within the secondary window only, as shown in Fig. 6(c). If both windows are overlapped, we go

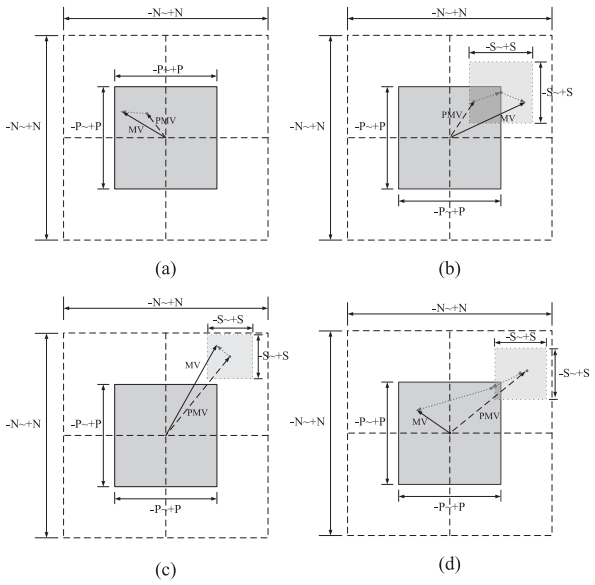


Fig. 6 Four cases for sizing of secondary window. (a) Case 1: Both PMV and motion vector are within the primary window and the secondary is not needed. (b) Case 2: The PMV is located in the primary window and the tracking of motion vector reaches the boundary of primary window. (c) Case 3: The PMV is out of the primary window and the motion vector search is not returning into the primary window. (d) Case 4: The PMV is out of the primary window and the motion vector search can go into the primary window when two sub-windows are overlapped.

Table 2 Size of secondary window versus coverage for Night D1 sequence.

Size of secondary window	Case 1	Case 2	Case 3	Case 4	Total
0	100%	21%	13%	0%	98.88%
1	100%	43%	61%	23%	99.47%
2	100%	53%	80%	33%	99.72%
3	100%	61%	89%	57%	99.84%
4	100%	65%	93%	70%	99.90%
5	100%	67%	96%	77%	99.93%
6	100%	71%	97%	80%	99.95%
7	100%	76%	98%	80%	99.97%
18	100%	100%	100%	100%	100%
. . .	. . .	. . .	. . .	. . .	. . .
127	100%	100%	100%	100%	100%
128	100%	100%	100%	100%	100%

with the forth case, as shown in Fig. 6(d). For the last case the MV searching will perform within the range covered by the primary and secondary windows.

In FSDSW, the range of the secondary window is deterministic and fixed based on statistical results. We simulated full-range ME on six D1 video sequences and intended to set the size of the secondary window for covering most of motion vectors. Table 2, for instance, illustrates the statistical results of the D1 clip "Night" with the small diamond search algorithm under the rate control of 2 Mbits/sec. In Table 2, we counted the occurrence times of four cases and calculated the coverage for given window size. As shown in the result, when the size of secondary window is 4, the total coverage is 99.90%. It means that one can set ±4, instead of ±32 as the size of secondary window and has the coverage as high as 99.90%.

Table 3 The quality degradation without the secondary windowing.

2M bit-rate control		
Video sequences	FSBM PSNRY (dB)	FSDSW (0-size secondary window) ΔPSNRY (dB)
Night	33.05	−0.36
Football	36.27	−1.12
Crew	35.87	−0.31
Character	31.41	−0.6
Akiyo	41.31	−0.1
Coastguard	34.91	−0.22

Table 4 Sizing of secondary window with DS/SDS algorithms in different bit-rates and motion types.

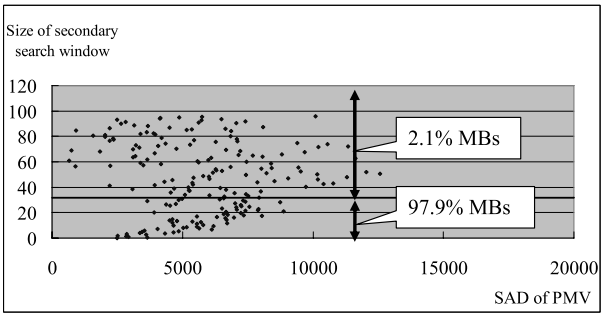
	Fast motion	Moderate motion	Slow motion	Average for all video sequences
Bit-rate	DS algorithm			
2 Mbits/s	10	6	1	8
4 Mbits/s	14	10	6	12
6 Mbits/s	16	11	9	14
Bit-rate	SDS algorithm			
2 Mbits/s	5	1	1	4
4 Mbits/s	8	3	2	7
6 Mbits/s	9	3	4	8

Table 3 shows the degradation without the secondary windowing. For fast-motion clips, without the secondary window, the quality degradation might be greater than 0.5 dB. The quality degradation of greater than 0.5 dB is sensible for human perception. The degradation of "football" clip, for instance, can be as high as 1.12 dB, because its coverage without the secondary windowing is 94.5%. To determine the size of secondary windows, we use 0.3 dB as the threshold of quality degradation. We intend to find the minimum size of secondary windows for less than 0.3 dB of the quality degradation. Table 4 shows the sizes of secondary windows based on calculating average values of sizes for different motion degrees of video sequences or all testing video sequences in different bit-rate control and motion estimation algorithms.

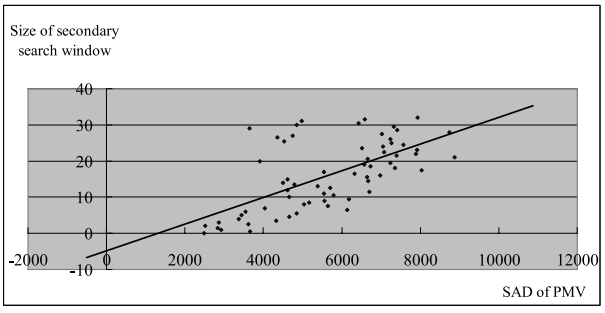
3.3 The Variable-Secondary-Window Dual-Search-Window Algorithm (VSDSW)

Instead of applying fixed size for the secondary window, we developed VSDSW to adaptively adjust the size of secondary window based on the SAD value of PMV for a specific MB. As shown in Fig. 6, motion estimation process starts after PMV stage because the PMV can efficiently predict a good starting point for each MB. Hence, the range of MV searching is limited and depended on the SAD values of PMVs. To formulate the relation between SAD value of PMV and the required size of secondary window, we

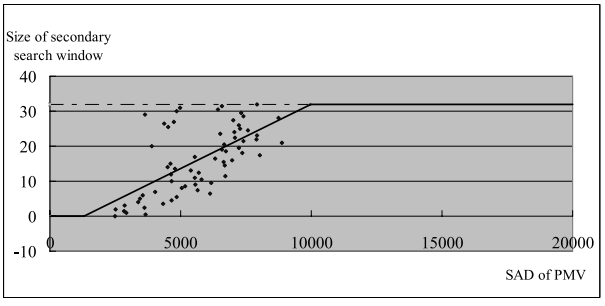
collected the SAD-size data as shown in Fig. 7, where the search algorithm is DS algorithm and bit-rate is 2M bits/s. From Fig. 7(a), we observed that the larger the SAD value, the larger the size of secondary window. When the size of secondary window is less than 32, there are 97.9% MBs can be covered for correct MV searches. Thereafter, we first applied the least mean square (LMS) method for the region of the 97.9% MBs to find the optimally-fitting, first-order curve, which is a line shown in Fig. 7(b). Since the maximum size of the secondary window is 32, we made the transfer curve, $Size_{secondary\_window} = f(SAD_{PMV})$, a piecewise-linear curve, as shown in Fig. 7(c). There are two boundary conditions; one is zero while the SAD value of PMV is less than 1280.7, and the other is 32 while the SAD of PMV value is greater than 9929.4. Finally, we concluded the parameters of optimal curve and boundary conditions in different bit-rate control and motion estimation algorithms for six testing D1 sequences as illustrated in the Table 5.



(a)



(b)



(c)

Fig. 7 (a) Distribution of $Size_{secondary\_window}$ versus SAD_{PMV}. (b) The optimal fitting line using LMS method. (c) The final piece-wise linear curve for VSDSW sizing. (Note: the search algorithm is DS algorithm and bit rate is 2M bits/s)

4. Experimental Results

The proposed architecture has been fabricated in the MPEG-4 CODEC chip, named AVS-1008, using UMC 0.13 μm 1P8M. The core size of the chip is $5490 \times 4950\,\mu m^2$ and the chip photograph is shown in Fig. 8. The memory sizes of the primary and secondary windows are 0.506 mm^2 and 0.272 mm^2, respectively. They cost 2.87% of the whole chip. The specifications of AVS 1008 are summarized in Table 6.

In AVS-1008, the motion estimator is not idle while the data is being loaded into the secondary window. The data loading and motion estimation are performed in parallel. Their executions are pipelined. Figure 9 illustrates the pipelined schedule of data loading and motion estimation operations. The data loading operations are performed in the direct memory access (DMA) and the motion estimation is performed in the motion estimator (ME). The DMA loads image data from DRAM to four SRAM blocks. The four

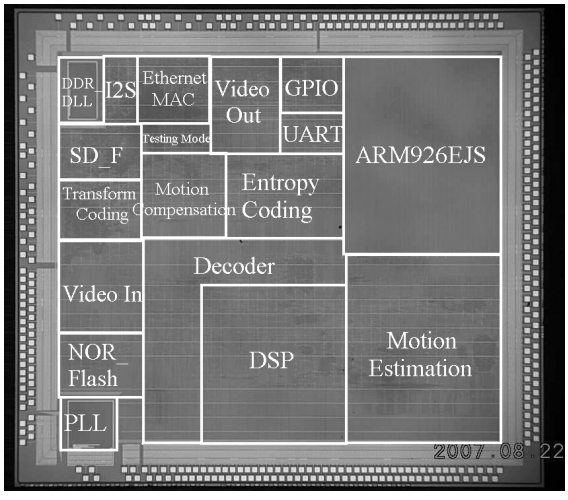


Fig. 8 Chip photograph of AVS-1008.

Table 5 Parameters of optimal curve for VSDSW sizing with D1 video sequences.

Size of secondary window=a×(SAD_{PMV})+b						
Motion estimation algorithms	Bit-rate					
	2M bits/s		4M bits/s		6M bits/s	
DS algorithm	1280.7≤SAD_{PMV}≤9929.4		1535.8≤SAD_{PMV}≤10184		1857.7≤SAD_{PMV}≤8814.2	
	a=0.0037	b=−4.7836	a=0.0037	b=−5.6826	a=0.0046	b=−8.5453
SDS algorithm	6.8235≤SAD_{PMV}≤18830.3		1534.8≤SAD_{PMV}≤10947		922.233≤SAD_{PMV}≤11589	
	a=0.0017	b=−0.0116	a=0.0034	b=−5.2183	a=0.003	b=−2.7667

Table 6 Specifications of AVS-1008.

Function	MPEG4 ASP Real Time CODEC
Video input	NTSC:720×480 @ 30 fps PAL: 720×576 @ 25 fps
Video resolution	QCIF, CIF, VGA, 16x, D1(max)
CPU	32-bit RISC ARM926EJ embedded 16-KB I-cache and 16-KB D-cache
ME search area	primary windowing(max): H:[−32~+32], V:[−32~+32] secondary windowing(max): H:[−16~+16], V:[−16~+16]
External memory	64-MB DDR / 16-MB NOR-type flash
Internal memory	868k bits SRAM
Technology	UMC 0.13 μm 1P8M
Supply voltage	1.2 V
Clock	external clock input : 27 MHz internal AHB clock : 75.6, 81, 87.75, 90, 94.5, 100 MHz internal APB clock : 37.8, 40.5, 43.9, 45, 47.25, 50 MHz
Power consumption	234 mW
Core size	$5490 \times 4950\,\mu m^2$
Chip size	$17 \times 17\,mm^2$
Transistor count	6.4 million
Package	256-pin BGA

SRAM blocks are mb_1, mb_2, sw_p, and sw_{sec}. The mb_1 and mb_2 store two consecutive macro-blocks. In Fig. 9, $mb_1(r_1)$ and $mb_2(r_2)$ stand for loading of macro-blocks r_1 and r_2 of the current frame. The SRAM blocks mb_1 and mb_2 behave as the ping-pong buffer; that is, when the SRAM mb_1 is being processed in ME the DMA is loading data to the SRAM mb_2, and vice versa. The SRAM sw_p stores the primary search window as shown in Fig. 4. As mentioned in Sect. 2, the primary search window only needs to update one or more slices for each new motion vector search. Thus, the operations $sw_p(SW_1)$ and $sw_p(SW_2)$ express the incremental loading of primary search windows of the macroblocks r_1 and r_2. For $type_p = 1$ and 3, the incremental loading is the update of the slice 6. For $type_p = 2$ and 4, the incremental loading is the update of the slices 7 and 8. As shown in Fig. 9, there are two cases. When the secondary search window is not required, the pipeline of DMA and ME is running regularly. When the secondary search window is required, the loading of secondary search window is added in DMA and the pipeline becomes irregular. The MV search in secondary search window is activated right after the loading; meanwhile, the data loading of the following macroblock is being performed. Note that the operation $ME(SW_{p1}, r_1)$ stands for the motion estimation of the macroblock r_1 in the primary search window and the operation $ME(SW_s, r_1)$ for the motion estimation of the macroblock r_1 in the secondary search window.

In this work, we use six D1-sized (or 720×480) video sequences to validate the design, listed in Table 7, to analyze the memory bandwidth, the size of local SRAM and visual quality with our proposed algorithms. The standard for the picture quality evaluations is based on H.264/AVC

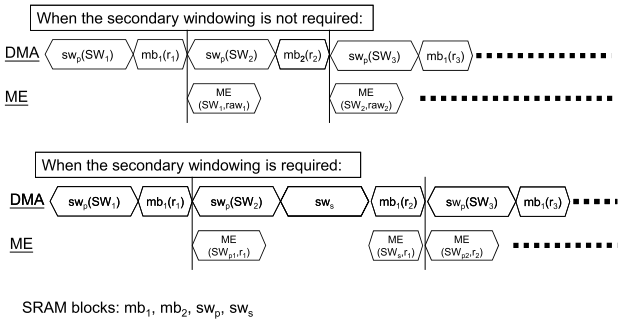


Fig. 9 The pipelined schedule of DMA and ME operations.

Table 7 Video sequences for D1 at 30 fps.

	Video Sequence	Number of Frames
Fast Motion	Night	230
	Football	260
Moderate Motion	Crew	300
Slow Motion	Character	260
	Akiyo	30
	Coastguard	300

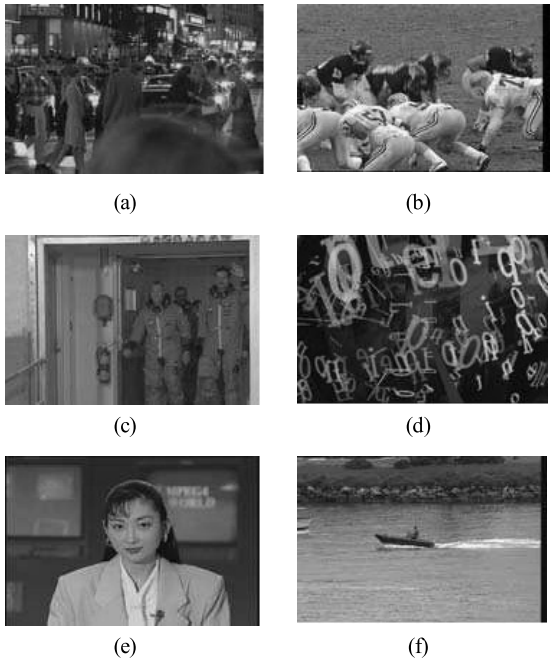


Fig. 10 D1 test clips: (a) Night, (b) Football, (c) Crew, (d) Character, (e) Akiyo, and (f) Coastguard.

software model. Figure 10 illustrates the first frames of D1 clips. In the simulation, the full-search-range is ±64 and the MB size is 16-by-16. We use ±32 as the size of primary window and three bit-rates (say 2 Mbits/sec, 4 Mbits/sec and 6 Mbits/sec) to measure the quality degradation. We use DSW($type_p$,$type_s$) to clarify the dual-search-window methods, where $type_p$ is the type of primary windowing (1: single MB, 2: two horizontal MBs, 3: two vertical MBs, and 4: four MBs) and $type_s$ is the type of second windowing (1:

Table 8 Bandwidth requirements for D1 clips with DS algorithm under 2M bit-rate control.

DS algorithm at 2M bit-rate						
DSW	Night	Football	Crew	Character	Akiyo	Coastguard
methods	Bandwidth	Bandwidth	Bandwidth	Bandwidth	Bandwidth	Bandwidth
DSW(1,1)	51.7 MB/s	55.8 MB/s	50.8 MB/s	49.8 MB/s	50.1 MB/s	49.8 MB/s
DSW(1,2)	50.7 MB/s	52.4 MB/s	49.9 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(1,3)	50.9 MB/s	51.3 MB/s	49.9 MB/s	49.8 MB/s	50 MB/s	49.8 MB/s
DSW(2,1)	51.7 MB/s	55.8 MB/s	50.8 MB/s	49.8 MB/s	50.1 MB/s	49.8 MB/s
DSW(2,2)	50.7 MB/s	52.4 MB/s	49.9 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(2,3)	50.9 MB/s	51.3 MB/s	49.9 MB/s	49.8 MB/s	50 MB/s	49.8 MB/s
DSW(3,1)	31.7 MB/s	35.8 MB/s	30.4 MB/s	29.7 MB/s	30 MB/s	29.7 MB/s
DSW(3,2)	30.7 MB/s	32.4 MB/s	29.9 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s
DSW(3,3)	30.8 MB/s	31.3 MB/s	29.8 MB/s	29.7 MB/s	30 MB/s	29.7 MB/s
DSW(4,1)	31.7 MB/s	35.8 MB/s	30.4 MB/s	29.7 MB/s	30 MB/s	29.7 MB/s
DSW(4,2)	30.7 MB/s	32.4 MB/s	29.9 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s
DSW(4,3)	30.8 MB/s	31.3 MB/s	29.8 MB/s	29.7 MB/s	30 MB/s	29.7 MB/s

Table 9 Bandwidth requirements for D1 clips with SDS algorithm under 2M bit-rate control.

SDS algorithm at 2M bit-rate						
DSW	Night	Football	Crew	Character	Akiyo	Coastguard
methods	Bandwidth	Bandwidth	Bandwidth	Bandwidth	Bandwidth	Bandwidth
DSW(1,1)	50.6 MB/s	52.3 MB/s	50 MB/s	49.8 MB/s	49.9 MB/s	49.8 MB/s
DSW(1,2)	50 MB/s	50.3 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(1,3)	50.8 MB/s	50.3 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(2,1)	50.6 MB/s	52.3 MB/s	50 MB/s	49.8 MB/s	49.9 MB/s	49.8 MB/s
DSW(2,2)	50 MB/s	50.3 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(2,3)	50.8 MB/s	50.3 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s	49.8 MB/s
DSW(3,1)	30.5 MB/s	32.2 MB/s	30 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s
DSW(3,2)	30 MB/s	30.2 MB/s	29.7 MB/s	29.7 MB/s	29.7 MB/s	29.7 MB/s
DSW(3,3)	30.1 MB/s	30.3 MB/s	29.7 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s
DSW(4,1)	30.5 MB/s	32.2 MB/s	30 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s
DSW(4,2)	30 MB/s	30.2 MB/s	29.7 MB/s	29.7 MB/s	29.7 MB/s	29.7 MB/s
DSW(4,3)	30.1 MB/s	30.3 MB/s	29.7 MB/s	29.7 MB/s	29.8 MB/s	29.7 MB/s

FEDSW, 2: FSDSW, 3: VSDSW).

The following equations are used to evaluate the external memory bandwidth, where fps is the frame rate, $N_{s,discont}$ is the number of discontinuous secondary windowing times, $N_{s,cont}$ is the number of continuous secondary windowing times, m_y is the number of horizontal pixels of target MB set for each primary windowing, $N_{secondary}$ is the number of secondary windowing times, and $f(s_i)$ expresses the number of pixels loaded in the ith secondary windowing.

$$BW_{DSW(type_p,1)} = [N_{access,p} + (2P+1)^2 \times N_{s,discont} + (2P+1) \times m_y \times N_{s,cont}] \times fps \quad (4)$$

$$BW_{DSW(type_p,2)} = [N_{access,p} + (2S+1)^2 \times N_{secondary}] \times fps \quad (5)$$

$$BW_{DSW(type_p,3)} = \left(N_{access,p} + \sum_{i=1}^{N_{secondary}} f(s_i) \right) \times fps \quad (6)$$

First of all, we estimate the required external memory bandwidth for D1 clips with DS and SDS algorithms. Table 8 and Table 9 shows that using the single-MB windowing has the same result as using two-horizontal-MB windowing, and the other two windowing techniques has the same bandwidth requirement as well. The DSW methods with $type_p = 3$ and $type_p = 4$ are better than the other two, and can save the external memory bandwidth up to 40.36%.

Next, we further calculate the size of local memory as shown in Table 10. In Table 10, the required sizes of local memory for primary windowing are 60k bits ((5×6)×(16×16)×8)/1024), 80k bits ((5×8)×(16×16)×8)/1024), 72k bits ((6×6)×(16×16)×8)/1024), 96k bits ((6×8)×(16×16)×8)/1024) for single-MB, two-horizontal-MB, two-vertical-MB, and four-MB windowing techniques, respectively. For the secondary windowing, with the four-MB primary windowing, FEDSW requires 96 Kbits ((6×8)×(16×16)× 8)/1024) local memory and VSDSW requires 50 Kbits ((5×5)×(16×16)×8/1024). Comparing with the others, FSDSW requires the minimum local memory. It requires 8 Kbits $((8+8+16)^2 \times 8/1024)$ with DS algorithm and 4.5 Kbits $((4+4+16)^2 \times 8/1024)$ with SDS algorithm, when the bit-rate control is set to 2 Mbits/sec. Table 10 sums up the total SRAM size based on the memory requirements of primary and secondary window and also calculates the memory increasing ratio normalized with single-MB windowing technique. From the analysis of memory bandwidth and local memory requirements, DSWs with $type_p = 1$ and $type_p = 2$ has the same bandwidth requirement while the latter with DS algorithm requires 33.3%, 29.41% and 18.18% more local memory than the former for FEDSW, FSDSW, and VSDSW, respectively. With SDS algorithm, the latter requires 33.3%, 31.01% and 18.18% more local memory than the former. Also, as shown

Table 10 Results of local SRAM sizes.

Motion etimation algorithm	DSW methods	The SRAM size of primary window	The SRAM size of secondary window	Total SRAM size	Increasing ratio
DS algorithm	DSW(1,1)	60 Kbits	60 Kbits	120 kbits	—
	DSW(1,2)	60 kbits	8 Kbits	68 Kbits	—
	DSW(1,3)	60 Kbits	50 Kbits	110 Kbits	—
	DSW(2,1)	80 Kbits	80 Kbits	160 Kbits	33.3%
	DSW(2,2)	80 Kbits	8 Kbits	88 Kbits	29.41%
	DSW(2,3)	80 Kbits	50 Kbits	130 Kbits	18.18%
	DSW(3,1)	72 Kbits	72 Kbits	144 Kbits	20%
	DSW(3,2)	72 Kbits	8 Kbits	80 Kbits	17.65%
	DSW(3,3)	72 Kbits	50 Kbits	122 Kbits	10.91%
	DSW(4,1)	96 Kbits	96 Kbits	192 Kbits	60%
	DSW(4,2)	96 Kbits	8 Kbits	104 Kbits	52.94%
	DSW(4,3)	96 Kbits	50 Kbits	146 Kbits	32.73%
SDS algorithm	DSW(1,1)	60 Kbits	60 Kbits	120 Kbits	—
	DSW(1,2)	60 Kbits	4.5 Kbits	64.5 Kbits	—
	DSW(1,3)	60 Kbits	50 Kbits	110 Kbits	—
	DSW(2,1)	80 Kbits	80 Kbits	160 Kbits	33.3%
	DSW(2,2)	80 Kbits	4.5 Kbits	84.5 Kbits	31.01%
	DSW(2,3)	80 Kbits	50 Kbits	130 Kbits	18.18%
	DSW(3,1)	72 Kbits	72 Kbits	144 Kbits	20%
	DSW(3,2)	72 Kbits	4.5 Kbits	76.5 Kbits	18.6%
	DSW(3,3)	72 Kbits	50 Kbits	122 Kbits	10.91%
	DSW(4,1)	96 Kbits	96 Kbits	192 Kbits	60%
	DSW(4,2)	96 Kbits	4.5 Kbits	100.5 Kbits	55.81%
	DSW(4,3)	96 Kbits	50 Kbits	146 Kbits	32.73%

Table 11 Comparisons of visual quality for DS algorithm in PSNRY.

2M bit-rate control					
Video sequences	FSBM PSNRY (dB)	DS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	33.05	−0.05	−0.07	−0.07	−0.07
Football	36.27	−0.24	−0.25	−0.26	−0.25
Crew	35.87	0.1	0.07	0.09	0.1
Character	31.41	0.01	0.01	0.01	0.01
Akiyo	41.31	0.01	0.01	0.01	0.01
Coastguard	34.91	−0.04	−0.04	−0.04	−0.04
4M bit-rate control					
Video sequences	FSBM PSNRY (dB)	DS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	36.34	−0.03	−0.03	−0.03	−0.01
Football	39.69	−0.1	−0.1	−0.1	−0.1
Crew	38.48	−0.05	−0.05	−0.06	−0.05
Character	35.13	0.03	0.03	0.03	0.03
Akiyo	41.31	0.01	0.01	0.01	0.01
Coastguard	37.95	−0.01	−0.01	−0.02	−0.01
6M bit-rate control					
Video sequences	FSBM PSNRY (dB)	DS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	38.18	0	0	−0.02	0
Football	41.57	−0.04	−0.04	−0.05	−0.04
Crew	39.88	0.04	0.05	0.05	0.05
Character	37.33	−0.02	−0.02	−0.02	−0.02
Akiyo	41.31	0	0	0	0
Coastguard	39.88	0	0	0	0

in the result, DSWs with $type_p = 3$ and $type_p = 4$ can save 40.36% external memory bandwidth when comparing with DSW with $type_p = 1$. Although DSWs with $type_p = 3$ and $type_p = 4$ have the same requirement for memory bandwidth, the latter needs larger local memory size than the former. The memory increasing ratios are 33.3% $(((192-144)/144)\times 100\%)$, 30% $(((104-80)/80)\times 100\%)$ and 19.67% $(((146-122)/122)\times 100\%)$ in DS algorithm and 33.3% $(((192-144)/144)\times 100\%)$, 31.37% $(((100.5-76.5)/76.5)\times 100\%)$ and 19.67% $(((146-122)/122)\times 100\%)$ in SDS algorithm for FEDSW, FSDSW, and VSDSW, respectively.

Table 11 shows the visual qualities for DS algorithm. For high motion video clips, such as "Night" and "Football,"

Table 12 Comparisons of visual quality for SDS algorithm in PSNRY.

2M bit-rate control					
Video sequences	FSBM PSNRY (dB)	SDS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	33.05	−0.21	−0.21	−0.23	−0.21
Football	36.27	−0.84	−0.84	−0.86	−0.84
Crew	35.87	−0.06	−0.06	−0.08	−0.06
Character	31.41	−0.41	−0.41	−0.41	−0.41
Akiyo	41.31	0	0	0	0
Coastguard	34.91	−0.07	−0.07	−0.07	−0.07
4M bit-rate control					
Video sequences	FSBM PSNRY (dB)	SDS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	36.34	−0.06	−0.06	−0.08	−0.06
Football	39.69	−0.26	−0.26	−0.34	−0.27
Crew	38.48	−0.02	−0.03	−0.06	−0.04
Character	35.13	−0.1	−0.12	−0.13	−0.12
Akiyo	41.31	0	0	0	0
Coastguard	37.95	−0.02	−0.02	−0.02	−0.02
6M bit-rate control					
Video sequences	FSBM PSNRY (dB)	SDS ΔPSNRY (dB)	FEDSW ΔPSNRY (dB)	FSDSW ΔPSNRY (dB)	VSDSW ΔPSNRY (dB)
Night	38.18	−0.02	−0.03	−0.04	−0.04
Football	41.57	−0.07	−0.07	−0.14	−0.07
Crew	39.88	0.02	0.02	0.02	0.02
Character	37.33	0.03	0.03	0.03	0.03
Akiyo	41.31	0	0	0	0
Coastguard	39.88	0	0	0	0

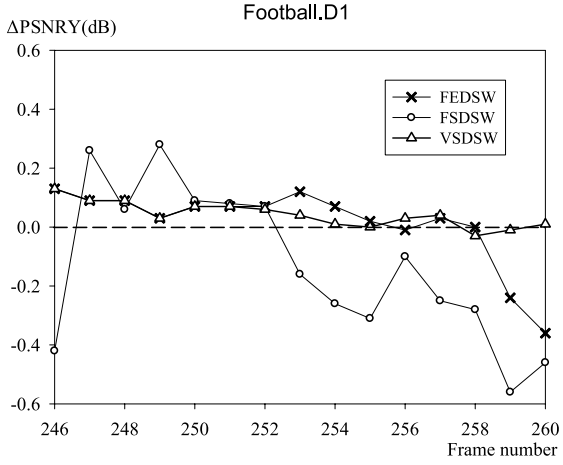


Fig. 11 The dynamic quality performance of D1 clip "Football" with DS algorithm and 2 Mbits/sec of bit-rate.

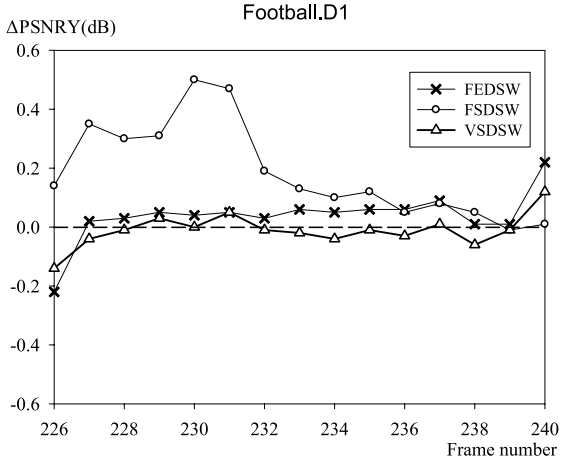


Fig. 12 The dynamic quality performance of D1 clip "Football" with SDS algorithm and 2 Mbits/sec of bit-rate.

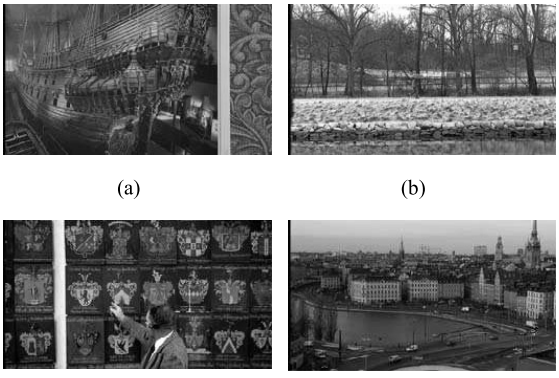
the quality degradation (△PSNRY) is less than 0.26 dB. It means that the quality degradation of proposed algorithms can be less than 0.5 dB which is considered as a minor degradation in the community of video compression. Note that the degradation is even less (say 0.02 dB) when comparing with full-range-windowing. We also applied the algorithms for SDS algorithm. From Table 12, the quality degradation is getting worse than DS algorithm; however, when comparing with full-range-windowing, the degradation is less than 0.02 dB.

To observe the dynamic degradation, we estimate the quality degradation of each frame for all D1 video clips at 2M bit-rate control. After exhaustive simulation, we conclude that the approach VSDSW can have the best visual quality among the proposed algorithms. Because of the space limitation of the manuscript, this paper does not present all results. Instead, we use the "football" sequence as an example. Figure 11 and Fig. 12 illustrate the variation of quality degradation. As shown in the results, FSDSW has worse transient degradation than FEDSW and VSDSW while the VSDSW is better than FEDSW.

To show the proposed algorithms can actually save the memory requirements for high resolution clips, we also use high-definition (HD) clips [38] to demonstrate the DSW techniques. The testcases are listed in Table 13 and their video format is 720 p (1280×720). Figure 13 illustrates the

Table 13 Video sequences for HDTV at 50 fps.

	Video Sequence	Number of Frames
Moderate Motion	Mobcal	504
Slow Motion	Shileds	504
	Stockholm	604
	Parkrin	504



(a) (b)

(c) (d)

Fig. 13 The first frames of HDTV video sequences: (a) Mobcal, (b) Parkrun, (c) Shield, and (d) Stockholm.

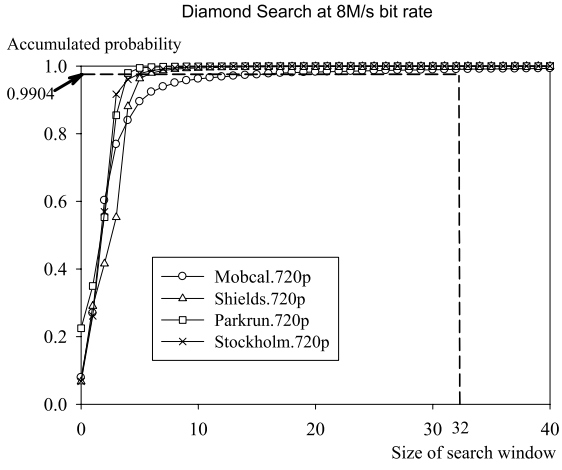


Fig. 14 The analysis of accumulated probability versus size of search window for DS algorithm under the rate control for 8 Mbits/sec.

first frames of four HDTV clips. We consider that the full search range is ±128 and the MB size is 16-by-16. From Fig. 14 and Fig. 15, we can also use ±32 as the size of primary window. One may note that the motion vectors of tested HD clips are shorter than those of D1 clips in average. This is because we hardly find fast-motion HD video clips and the clips under testing are all slow-motion ones. However, we can still use these HD clips to illustrate the capability of the proposed architecture for HD video. Using the VSDSW with parameters of Table 14, Table 15 and Table 16 illustrate that, comparing with FSBM, the quality degradation is as low as 0.16 dB. Its quality is very close to the full-range windowing technique, less than 0.07 dB

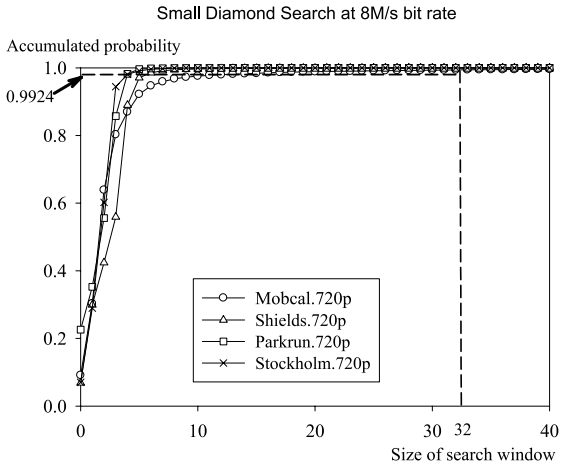


Fig. 15 The analysis of accumulated probability versus size of search window for SDS algorithm under the rate control for 8 Mbits/sec.

Table 14 Parameters of optimal curve for VSDSW sizing with HD video sequences.

Size of secondary search window=a×(SAD_{PMV})+b		
Motion estimation algorithms	Bit-rate	
	8M bits/s	
DS algorithm	$0 \le SAD_{PMV} \le 18060$	
	a=0.00016	b=3.1034
SDS algorithm	$0 \le SAD_{PMV} \le 23596$	
	a=0.0013	b=1.3246

Table 15 Comparisons of visual quality for HDTV clips with DS algorithm in PSNRY.

8M bit-rate control			
Video sequences	FSBM	DS	VSDSW
	PSNRY	ΔPSNRY	ΔPSNRY
Mobcal	32.97 dB	−0.06 dB	−0.06 dB
Shileds	32.83 dB	0.16 dB	0.16 dB
Stockholm	33.85 dB	0 dB	0 dB
Parkrin	26.06 dB	0 dB	−0.05 dB

Table 16 Comparisons of visual quality for HDTV clips with SDS algorithm in PSNRY.

8M bit-rate control			
Video sequences	FSBM	SDS	VSDSW
	PSNRY	ΔPSNRY	ΔPSNRY
Mobcal	32.97 dB	−0.04 dB	−0.05 dB
Shileds	32.83 dB	0.16 dB	0.16 dB
Stockholm	33.85 dB	0 dB	0 dB
Parkrin	26.06 dB	−0.02 dB	−0.09 dB

Table 17 Results of local SRAM sizes for HDTV clips.

Motion estimation algorithm	DSW methods	SRAM size of primary search window	SRAM size of secondary search window	Total SRAM size
DS	DSW(4,3)	96 Kbits	50 Kbits	146 Kbits
SDS	DSW(4,3)	96 Kbits	50 Kbits	146 Kbits

Table 18 Bandwidth requirements (in MB/s) for HD clips with DS and SDS algorithms under 8M bit-rate control.

DS algorithm at 8M bit-rate				
Methods	Mobcal	Shileds	Stockholm	Parkrin
DSW(4,3)	136.18	135.30	135.21	135.23
SDS algorithm at 8M bit-rate				
Methods	Mobcal	Shileds	Stockholm	Parkrin
DSW(4,3)	135.59	135.23	135.20	135.21

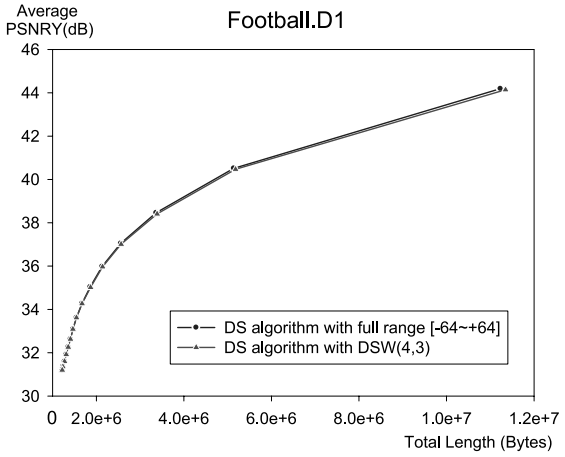


Fig. 16 The rate-distortion curve of the "football" D1 clip.

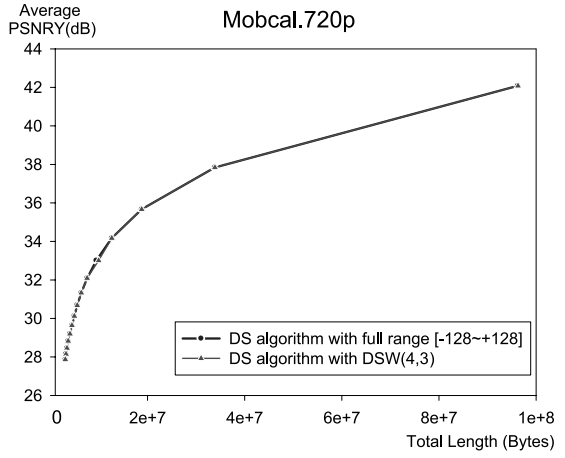


Fig. 17 The rate-distortion curve of the "mocal" HD clip.

degradation while the memory requirements are way less than the full-range windowing. Table 17 shows the total size of local memory using DSW(4,3) is 146 kbits and Table 18 shows the results of bandwidth performance for testing HD clips. The results show that the proposed approach can save 76.14% of local memory and 81.41% of external memory bandwidth while the traditional approach may require 612 Kbits local memory and 727.51 MBytes/sec.

5. Conclusion

As the demand of high-resolution video applications increases, to solve the notorious power-consuming problem, the memory requirements have been the most important factors for the CODEC performance and quality. Given the limited local memory size, this paper mainly focuses on the reduction of external memory bandwidth while the compression quality degradation is little. The reduction of memory bandwidth implies the save of power consumption. We proposed three windowing algorithms for center-biased motion estimations and take the advantage of minimizing the required data accesses in the center-biased motion estimations. At the same time, we also take the data reusability into account. Under the rate-control mechanism, the proposed windowing can significantly save the external memory bandwidth. As shown in Fig. 16 and Fig. 17, the quality degradation is very little for either D1 or HDTV video clips. For 720 p HDTV sequences, the proposed windowing algorithms only require the external memory bandwidth as low as 135.20 MBytes/Sec, while the quality degradation is less than 0.2 dB.

Acknowledgments

This work was supported by the National Science Council, R.O.C., under the grant number NSC 95-2221-E-009-337-MY3. The authors would like to acknowledge National Chip Implementation Center (CIC) for technical support.

References

[1] Information Technology — Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s — Part 2: Video, ISO/IEC 11172-2, 1993.
[2] Information Technology — Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC 13818-2 and ITU-T Recommendation H.262, 1996.
[3] Information Technology — Coding of Audio-Visual Objects — Part 2: Visual, ISO/IEC 14496-2, 1999.
[4] Joint Video Team, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC, May 2003.
[5] P. Kuhn, Algorithm, Complexity Analysis And VLSI Architecture for MPEG-4 Motion Estimation, Kluwer Academic Publishers, 1999.
[6] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Trans. Circuits Syst. Video Technol., vol.12, no.1, pp.61–72, Jan. 2002.
[7] J.-Y. Kim and S.-B. Yang, "An efficient search algorithm for BLOCK motion estimation," IEEE Workshop on Signal Processing Systems, pp.100–109, Oct. 1999.
[8] J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans. Commun., vol.29, no.12, pp.1799–1808, Dec. 1981.
[9] M.J. Chen, L.G. Chen, and T.D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," IEEE Trans. Circuits Syst. Video Technol., vol.4, no.5, pp.504–509, Oct. 1994.
[10] R. Li, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.4, no.4, pp.438–442, Aug. 1994.
[11] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching motion estimation," ICICS'97, pp.9–12, Singapore, Sept. 1997.
[12] J.Y. Tham, S. Ranganath, M. Ranganath, and A.A. Kassim, "A novel unrestricted center-biased dimond search algorithm for block motion

estimation," IEEE Trans. Circuits Syst. Video Technol., vol.8, no.4, pp.369–377, Aug. 1998.
[13] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Trans. Image Process., vol.9, no.2, pp.287–290, Feb. 2000.
[14] C. Zhu, X. Lin, and L.P. Chau, "Hexagon-based search pattern for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.12, no.5, pp.349–355, May 2002.
[15] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.6, no.3, pp.313–317, June 1996.
[16] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Trans. Image Process., vol.9, no.2, pp.287–290, Feb. 2000.
[17] X.-Q. Banh and Y.-P. Tan, "Adaptive dual-cross search algorithm for block-matching motion estimation," IEEE Trans. Consum. Electron., vol.50, no.2, pp.766–775, May 2004.
[18] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," IEEE Trans. Image Process., vol.11, no.12, pp.1442–1449, Dec. 2002.
[19] C.-H. Cheung and L.-M. Po, "Novel cross-diamond-hexagonal search algorithms for fast block motion estimation," IEEE Trans. Multimed., vol.7, no.1, pp.16–22, Feb. 2005.
[20] X. Jing and L.-P. Chau, "An efficient three-step search algorithm for block motion estimation," IEEE Trans. Multimed., vol.6, no.3, pp.435–438, June 2004.
[21] 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, no.4, pp.419–422, Aug. 1996.
[22] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," IEEE Trans. Image Process., vol.4, no.1, pp.105–107, Jan. 1995.
[23] V.L. Do and K.Y. Yun, "A low-power VLSI architecture for full-search block-matching motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.8, no.4, pp.393–398, Aug. 1998.
[24] J.H. Luo, C.N. Wang, and T. Chiang, "A novel all-binary motion estimation (ABME) with optimized haredware architectures," IEEE Trans. Circuits Syst. Video Technol., vol.12, no.8, pp.700–712, Aug. 2002.
[25] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," IEEE Trans. Circuits Syst. Video Technol., vol.3, no.2, pp.148–157, April 1993.
[26] C.K. Cheung and L.M. Po, "A hierarchical block motion estimation algorithm using partial distortion measure," IEEE ICIP, vol.3, pp.606–609, Oct. 1997.
[27] C.K. Cheung and L.M. Po, "Normalized partial distortion search algorithm for block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.10, no.3, pp.417–422, April 2000.
[28] C.N. Wang, S.W. Yang, C.M. Liu, and T. Chiang, "A hierarchical decimation lattice based on *N*-queen with an application for motion estimation," IEEE Signal Process. Lett., vol.10, no.8, pp.228–231, Aug. 2003.
[29] C.N. Wang, S.W. Yang, C.M. Liu, and T. Chiang, "A hierarchical *N*-queen decimation lattice and hardware architecture for motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.14, no.4, pp.429–440, April 2004.
[30] Y.-L. Chan and W.-C. Siu, "New adaptive pixel decimation for block motion vector estimation," IEEE Trans. Circuits Syst. Video Technol., vol.6, no.1, pp.113–118, Feb. 1996.
[31] Y.K. Wang, Y.Q. Wang, and H. Kuroda, "A globally adaptive pixel-decimation algorithm for block-motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol.10, no.6, pp.1006–1011, Sept. 2000.
[32] S. Lee and S.I. Chae, "Motion estimation algorithm using low-resolution quantization," Electron. Lett., vol.32, no.7, pp.647–648, March 1996.
[33] H.W. Cheng and L.R. Dung, "EFBLA: A two-phase matching algorithm for fast motion estimation," Advances in Multimedia Information Processing — PCM, vol.2532, pp.112–119, Dec. 2002.
[34] C.L. Su and C.W. Jen, "Motion estimation using msd-first processing," IEE Proc-G Circuits, Devices and Systems, vol.150, no.2, pp.124–133, April 2003.
[35] J.-F. Shen, T.-C. Wand, and L.-G. Chen, "A novel low-power full-search block-matching motion-estimation design for H.263+," IEEE Trans. Circuits Syst. Video Technol., vol.11, no.7, pp.890–897, July 2001.
[36] M. Brunig and W. Niehsen, "Fast full-search block matching," IEEE Trans. Circuits Syst. Video Technol., vol.11, no.2, pp.241–247, Feb. 2001.
[37] L. Sousa and N. Roma, "Low-power array architectures for motion estimation," 1999 IEEE 3rd Workshop on Multimedia Signal Processing, pp.679–684, 1999.
[38] http://www.ldv.ei.tum.de/liquid.php?page=70

Lan-Rong Dung was born in 1966. He received a BSEE and the Best Student Award from Feng Chia University, Taiwan, in 1988, an MS in electronics engineering from National Chiao Tung University, Taiwan, in 1990, and Ph.D. in electrical and computer engineering from Georgia Institute of Technology, in 1997. From 1997 to 1999 he was with Rockwell Science Center, Thousand Oaks, CA, as a Member of the Technical Staff. He joined the faculty of National Chiao Tung University, Taiwan in 1999 where he is currently an associate professor in the Department of Electrical and Control Engineering. He received the VHDL International Outstanding Dissertation Award celebrating in Washington DC in October, 1997. His current research interests include VLSI design, digital signal processing, hardware-software codesign, and System-on-Chip architecture. He is a member of Computer and Signal Processing societies of the IEEE.

Meng-Chun Lin received a B.S. degree in Electronic Engineering from Fu Jen Catholic University, Taipei, Taiwan, in 2001, and Ph.D. degree in the Electrical and Control Engineering, National Chiao Tung University Hsinchu, Taiwan, in 2007. He is currently working with Avisonic Technology Corp., Taiwan. His research interests are image processing, video processing, VLSI architecture and memory circuit design.