Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results — Source link

Journal of VLSI Signal Processing 42, 297–320, 2006
© 2006 Springer Science + Business Media, Inc. Manufactured in The Netherlands.
DOI: 10.1007/s11265-006-4190-4

Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results

YU-WEN HUANG, CHING-YEH CHEN, CHEN-HAN TSAI, CHUN-FU SHEN AND LIANG-GEE CHEN
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan

Received February 11, 2005; Revised February 11, 2005; Accepted August 4, 2005

Published online: 13 February 2006

Abstract. Block matching motion estimation is the heart of video coding systems. During the last two decades, hundreds of fast algorithms and VLSI architectures have been proposed. In this paper, we try to provide an extensive exploration of motion estimation with our new developments. The main concepts of fast algorithms can be classified into six categories: reduction in search positions, simplification of matching criterion, bitwidth reduction, predictive search, hierarchical search, and fast full search. Comparisons of various algorithms in terms of video quality and computational complexity are given as useful guidelines for software applications. As for hardware implementations, full search architectures derived from systolic mapping are first introduced. The systolic arrays can be divided into inter-type and intra-type with 1-D, 2-D, and tree structures. Hexagonal plots are presented for system designers to clearly evaluate the architectures in six aspects including gate count, required frequency, hardware utilization, memory bandwidth, memory bitwidth, and latency. Next, architectures supporting fast algorithms are also reviewed. Finally, we propose our algorithmic and architectural co-development. The main idea is quick checking of the entire search range with simplified matching criterion to globally eliminate impossible candidates, followed by finer selection among potential best matched candidates. The operations of the two stages are mapped to the same hardware for resource sharing. Simulation results show that our design is ten times more area-speed efficient than full search architectures while the video quality is competitively the same.

Keywords: block matching, motion estimation, global elimination algorithm, VLSI architecture

1. Introduction

Motion compensated transform coding has been adopted by all of the existing international video coding standards, such as the ISO MPEG series [1–3] and the ITU-T H.26X series [4–6]. Motion estimation (ME) removes temporal redundancy within frames and thus provides coding systems with high compression ratio. Since ME module is usually the most computationally intensive part (50–90% of the entire system) in a video encoder, efficient implementation of ME is a must. Block matching approach is mostly selected as the ME module in video codecs and is also adopted in all existing video coding standards because of its simplicity and good performance. The block matching algorithm (BMA) is described as follows. Each luma frame is divided into blocks of size $N \times N$, and each block in the current frame is matched with candidate blocks of size $N \times N$ within the search area in the reference frame. The best matched block has the lowest distortion among all of the candidate blocks. The displacement of the best matched block, or namely the motion vector (MV) of current block, will be transmitted with prediction residues to the decoder. The distortion is mostly evaluated by sum of absolute differences (SAD).

Among all the BMAs, full-search block matching algorithm (FSBMA) is the most popular. FSBMA can be described by:

$$\mathrm{SAD}(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |c(i,j) - s(i+m, j+n)| \tag{1}$$

$$\mathrm{MV} = \{(u,v) \mid \mathrm{SAD}(u,v) \le \mathrm{SAD}(m,n); \; -p \le m,n \le p-1\} \tag{2}$$

where SAD(m, n) is the distortion of the candidate block at search position (m, n), $\{c(x,y) \mid 0 \le x \le N-1, 0 \le y \le N-1\}$ means current block data, $\{s(x,y) \mid -p \le x \le p+N-2, -p \le y \le p+N-2\}$ stands for search area data, the search range is $[-p, p-1]$, the block size is $N \times N$, and MV expresses the motion vector of current block with minimum SAD among $(2p)^2$ search positions. FSBMA demands a lot of computation. For example, real-time ME for CIF (352×288) 30 frames per second (fps) video with $[-16, +15]$ search range requires 9.3 Giga-operations per second (GOPS). If the frame size is enlarged to D1 (720×480) 30 fps with $[-32, +31]$ search range, 127 GOPS is required. Clearly, such huge computational complexity is far beyond the processing capabilities of today's general purpose processors. Therefore, many fast algorithms and hardware architectures have been proposed.

The main purpose of this paper is to make a comprehensive study of ME algorithms and architectures. Comparisons in many directions are made for system designers to determine the best tradeoff. The rest of this paper is organized as follows. In Section 2, different categories of fast ME algorithms are discussed. Section 3 investigates both full search and fast search architectures. In Section 4, we propose our hardware-oriented algorithm and its corresponding architecture. Finally, Section 5 concludes this paper.

2. Exploration of Algorithms

We classify fast algorithms into six categories. The first five categories are lossy, which means that FSBMA outperforms them in video quality. The last category is lossless, which means that it produces the same results as FSBMA. A fast ME algorithm can belong to combination of the several categories.

2.1. Reduction in Search Positions

Under the assumption that the distortion monotonically increases as the search position moves away from the point corresponding to minimum distortion, convergence to the optimal position still can be achieved without matching all the candidates. Computation is thus significantly reduced by decimation of search positions. Since 1981 many algorithms of this type, e.g. two dimensional logarithmic search [7], three step search [8], conjugate direction search [9], modified logarithmic search [10], cross search [11], parallel hierarchical one dimensional search [12], one dimensional full search [13], new three step search [14], four step search [15], block-based gradient descent search [16], center-biased diamond search [17, 18], advanced diamond zonal search [19, 20], minimum bounded area search [21], one-dimensional gradient descent search [22], cross diamond search [23], predictive line search [24, 25], and many others, have been proposed. Roughly speaking, a newer algorithm requires less computation, achieves faster convergence, and results in higher video quality. In [12, 13], and [24], not only the number of searched candidates but also the feasibility of parallel processing, regularity of data flow, and efficiency of memory access were taken into consideration toward system optimization. The first diamond search [26], which is adopted by the reference software of MPEG-4, has significantly better performance in speed and quality than its prior algorithms. Although improvements still can be made after diamond search, the contribution is less recognized.

2.2. Simplification of Matching Criterion

The SAD matching criterion involves all pixels in the current block and the candidate block. In order to reduce the computational effort, a subsampling scheme was performed in [27]. Only every second pixel is taken into account for estimation of distortion in both horizontal and vertical directions, and the computational burden is reduced by a factor of four. Aliasing effects can be avoided by low-pass filtering. In [28, 29], periodic alternation of four subsampling patterns was adopted on different search positions to solve the aliasing without filtering. Adaptive pixel-decimation scheme was further proposed in [30]. It does not require an initial division of a block and selects pixels only when they have the features important in determining a match.

Pixel difference classification (PDC) [31] is to threshold every pixel absolute difference and classify it into match or mismatch. The best candidate block has the highest number of match pixels. The hardware complexity can be dramatically reduced because only one counter is required to replace the accumulators. However, the threshold value affects the quality a lot and is not easy to be decided automatically.

Minimax criterion [32] finds the maximum error among all pixels in a candidate block and then chooses the final MV by minimizing the maximum errors of all candidate blocks. Although the number of operations is not reduced in software implementation compared with FSBMA, minimax criterion can save 15% of hardware area because an eight-bit comparator is much smaller than a 16-bit accumulator. Boundary match [33] is also a simplified matching criterion. Moreover, it is often adopted for error concealment from loss of MVs.

The concept of integral projection was introduced in [34, 35]. For simple translational motion, the information on the axes in the Fourier transform domain is sufficient to estimate motion between two images. Computing the horizontal and vertical frequency information is equivalent to discrete approximation as integral projections at these two orientations. The simple equivalence between shifts in integral projection measurements and shifts in the corresponding segments of images suggests the comparison of integral projections as a computationally efficient technique for BMA since the pair of horizontal and vertical projections contain fewer data than the pixels in a candidate block.

2.3. *Bitwidth Reduction*

Originally, each pixel is represented with eight-bit resolution. In [36, 37], their algorithms involve transforming each pixel to one-bit representation and then applying conventional ME search strategies. In [38], they directly truncate the bitwidth of pixels. It is shown that on average more than four bits can be truncated without significantly affecting the picture quality. Pixel truncation can lead to substantial reduction in hardware complexity and power consumption. Fixed length truncation saves hardware areas and power consumption but runs the risk of losing too much quality. Adaptive truncation by masking the least significant bits of pixels to zero cannot reduce areas, but it has chances to save a lot of power without losing quality. Furthermore, pixel truncation can be also applied in software implementation for the popularity of single instruction multiple data (SIMD) of processors since fewer-bit representation may achieve higher degrees of parallel processing.

2.4. *Predictive Search*

For the video sequences with fast moving objects, the heuristic fast search algorithms of the first category perform poorly due to the frequent failure of monotonically increasing distortion model assumption. Algorithms belonging to decimation of search positions are often trapped in local minima of distortion, thus resulting in poor ME accuracy. Predictive motion estimation [39–43], which utilizes the motion information in the spatial and/or temporal neighboring blocks to form an initial estimate of current MV, can effectively reduce the search area as well as the computation. The reduced motion search area also provides an additional compression since the overhead information of MV is less. In [20], the MV predictors can be the MVs of the MBs on the left, top, and top right, their median, zero MV, the MV of the collocated MB in the previous frame, and the accelerated MV of the collocated MBs in the previous two frames. These predictors are most frequently adopted in predictive search.

2.5. *Hierarchical Search*

It is well known that a multiresolution structure, also known as a pyramid structure, is a very powerful computational configuration for image processing task. To save the computation of FSBMA, it is common to resort to the pyramid structure. The multiresolution scheme is based on the idea of predicting an initial estimate at the coarse level and refining the estimate at the fine level. Usually two- or three-level hierarchical search is adopted [44–46]. The search range at the fine level is much smaller than the original search range. Basically, more levels can save more computation, but the probability of being trapped in local minimum is higher because when the image is scaled down, the detailed textures will be lost. In fact, the multiresolution technique has been regarded as one of the most efficient methods in BMA and is mostly adopted in applications with very large frames and search areas.

2.6. *Fast Full Search*

The main idea of fast full search algorithms is stated as follows. In the early stage, a simple check is done to detect whether a candidate block is possible to be the optimal one. Then, only the potential candidate blocks are further processed for detailed distortion calculation. Thus, a large portion of unnecessary computation for impossible candidate blocks can be avoided. For example, successive elimination algorithm (SEA) [47] eliminates impossible candidate blocks by checking if the absolute difference between current block pixel sum and candidate block pixel sum is larger than the up-to-date minimum SAD, denoted as $\mathrm{SAD}_{\mathrm{min}}$. If the condition holds, it is proved that the SAD of the candidate block will be larger than $\mathrm{SAD}_{\mathrm{min}}$, and this search position should be skipped. If the condition fails, SAD calculation is still necessary for finding the global minimum distortion. The sum of all pixels in current block only has to be computed one time, and the sum of all pixels in a candidate block can be computed in a fast way by simple partial result reuse. Therefore, the computational overhead of the checking procedure is very small, and the skipping of impossible candidate blocks can speed up the whole BMA process. Note that a good initial guess of MV with smaller SAD is critical for SEA to increase the skipping ratio. Consequently, SEA is often combined with the use of MV predictors or spiral scanning order of search positions. Multilevel successive elimination algorithm (MSEA) [48–50] improves SEA by changing the checking procedure. The probability to skip candidate blocks can be greatly increased. In [51], MSEA is further improved by combination with SIMD for speeding up.

The concept of partial distortion elimination (PDE) [52] is simple and effective. Computation of FSBMA is reduced by using a halfway-stop technique in the SAD calculation. When the partial distortion of a candidate block is already larger than the current minimum distortion, this candidate block can be skipped. In [53], the partial distortions and the current minimum distortion were normalized to increase the probability of early rejection of non-possible candidate MVs. In addition, grouping of partial distortion and spiral scan of search positions also increase the probability of early termination. However, normalized PDE cannot guarantee exactly the same result as FSBMA. It may suffer a little quality loss but is still very close to FSBMA. In [54], an adaptive scanning order of pixels in a candidate block was proposed for distortion calculation to further speed up the PDE. In [55], models to describe the probability distribution of the total distortion given a measured partial distortion are introduced. In [56], analysis-based method for optimizing the timing of decisions regarding early termination was proposed for developing PDE algorithms on different platforms.

Winner-update algorithm [57] is a very interesting fast full search algorithm. Suppose there are five players in a game of poker cards, and each player is dealt four cards. The player with the minimum sum of the four card values is the winner. The basic idea is that one does not have to calculate the summation of all the card values for each player when determining the winner. In the beginning, every player shows one card, and the player with the smallest card value can show the second card. Then, only the player having the minimum sum of card values can reveal the next card, and the first player reaching the fourth card is the winner. Similarly, in the process of FSBMA, the candidate blocks and the pixels in a block can be regarded as the players and the number of cards dealt for each player, respectively. In [57], hash chain is used in the implementation of winner-update strategy to avoid the expensive sorting procedure of finding the minimum partial SAD value. Also, in order to reduce the size of hash table and thus achieve speed-up, MSEA is applied with normalized partial distortion.

In fact, the results of fast full search are not exactly the same as FSBMA. Sometimes, minor differences occur. For example, when two or more search positions have the same minimum SAD, the result will be dependent on scan order. However, these minor differences do not cause noticeable effect on quality.

2.7. *More Discussion and Comparison*

Combination with mode decision and encoding issues is another trend. In [58], a quantization parameter dependent threshold value is applied on SAD to detect all-zero residues and to early stop ME. In [59, 60], diversity-based method takes advantages of multiple algorithms. For example, diamond search or four step search has good performance when the motion field is small while three step search converges faster when the motion field is larger. Thus, a pre-checking can be developed to select what algorithm is more suitable for the current block. In [61], a computation-aware (CA) scheme for software-based BMA was introduced. In a computation-limited environment, the computation-distortion curve of a BMA is much more important than the rate-distortion curve for real-time applications. According to their experiments, CA diamond search is usually the best while CA three step search or CA

FSBMA is the worst. Recently, in the new video coding standard, H.264/AVC, multiple reference frames and variable block sizes make BMA much more complex, which becomes the hottest new topic of fast ME. However, we do not have enough space for this topic. Interested readers can refer to [46, 62–71], and many others. These papers were dedicated for the BMA in H.264/AVC. Other kinds of ME algorithms may include considerations for fractional pixel accuracy, bidirectional frames, fields, and deinterlacing.

Now we compare representative BMAs in terms of motion compensated PSNR and execution time. The platform is a personal computer (PC) with Pentium IV 2.53 GHz CPU and 1 GB DRAM (DDR 333 MHz), and the program is written in C language. The implemented BMAs are FSBMA, three step search (TSS), one dimensional full search (1DFS), center-biased diamond search (DS), predictor-biased diamond search (PDS), two-level hierarchical search (2-Level Hier.), three-level hierarchical search (3-Level Hier.), full search with 1/2-subsampling (in SAD computation), full search with 1/4-subsampling, full search with 1/8-subsampling, spiral SEA, spiral MSEA, spiral PDE, and winner-update strategy. The results of Foreman (QCIF [−16, +15]) and Stefan (CIF [−32, +31]) are shown in Tables 1 and 2, respectively. As can be seen, real-time encoding of QCIF 30 Hz video with FSBMA is not achievable even on such a high speed PC. If MMX/SSE instructions can be used, 2–3 times of speed-up can be further achieved, but the column of speed up will still be about the same. TSS, 1DFS, and DS belong to the category of reduction in search positions. DS has better tradeoff in speed and quality. PDS has better quality than DS with faster convergence, which shows the advantage of motion vector prediction. Hierarchical search BMAs provide moderate performance in both speed and quality. The quality drops of 1/2-, 1/4-, and 1/8-subsampling are insignificant, moderate, and unacceptable, respectively. As for fast full search, spiral MSEA is the fastest and is about 20 times faster than FSBMA. Note that the speed up of fast full search may vary a lot with different sequences. The earlier the global minimum distortion is found, the more computation is saved. For software implementations, lossy BMAs slower than spiral MSEA are out of popularity. Recently, prevailing lossy BMAs should be capable of providing more than 100 times of speed-up with acceptable quality loss (<0.5 dB).

Table 1. Comparison of BMAs for Foreman, QCIF 30 Hz, [−16, +15].

BMA	Time (ms)	Speed up	PSNR (dB)	Diff.
FSBMA	105.108	N/A	31.911	N/A
TSS	003.486	30.151	31.431	−0.480
1DFS	010.454	10.054	31.377	−0.534
DS	002.822	37.246	31.555	−0.356
PDS	002.506	41.943	31.730	−0.181
2-Level Hier.	009.008	11.668	31.752	−0.159
3-Level Hier.	007.130	14.742	31.634	−0.277
1/2-Subsample	048.439	02.170	31.868	−0.042
1/4-Subsample	024.987	04.207	31.739	−0.172
1/8-Subsample	014.882	07.063	30.355	−1.555
Spiral SEA	018.128	05.798	31.911	−0.000
Spiral MSEA	004.892	21.486	31.911	−0.000
Spiral PDE	024.083	04.364	31.911	−0.000
Winner-Update	009.905	10.612	31.903	−0.008

Time: Average milli-second per frame.
Platform: Pentium IV 2.53 GHz, 1 GB DDR 333 MHz DRAM, C language.

Table 2. Comparison of BMAs for Stefan, CIF 30 Hz, [−32, +31].

BMA	Time (ms)	Speed up	PSNR (dB)	Diff.
FSBMA	1,683.107	N/A	25.732	N/A
TSS	0,016.569	101.582	22.619	−3.113
1DFS	0,079.953	021.051	25.302	−0.430
DS	0,016.669	100.972	22.747	−2.985
PDS	0,014.736	114.217	24.507	−1.224
2-Level Hier.	0,122.649	013.723	25.700	−0.032
3-Level Hier.	0,059.886	028.105	25.186	−0.546
1/2-Subsample	0,770.482	002.184	25.724	−0.007
1/4-Subsample	0,393.656	004.276	25.513	−0.219
1/8-Subsample	0,225.806	007.454	24.878	−0.853
Spiral SEA	0,403.324	004.173	25.732	−0.000
Spiral MSEA	0,092.756	018.146	25.732	−0.000
Spiral PDE	0,470.268	003.579	25.732	−0.000
Winner-Update	0,164.505	010.231	25.713	−0.019

Time: Average milli-second per frame.
Platform: Pentium IV 2.53 GHz, 1 GB DDR 333 MHz DRAM, C language.

3. Investigation of Architectures

In this section, we will introduce FSBMA architectures and discuss on-chip memories for storing search area data, followed by using hexagonal plots to compare six aspects, and surveying fast BMA architectures.

3.1. FSBMA Architectures

Many FSBMA architectures were developed due to the regularity of data flow. In this subsection, we will try to bring a thorough survey of FSBMA designs. Most of them belong to systolic arrays [72] composed of locally connected processing elements (PEs). A pipelined simultaneous data flow via the local connections does not require any control overhead. The small load capacities to be driven permit high clock frequencies and thus higher processing speeds. Moreover, after a datum is accessed from memory, it is reused for each PE by propagating through the array, which significantly reduces the memory bandwidth. For FSBMA, each PE is responsible for computing the absolute difference of one current block pixel and one search area pixel. In the following discussion, we denote the block size and search range as $N \times N$ and $[-p, p-1]$, respectively.

In [73], Komarek and Pirsch contributed a detailed systolic mapping procedure to derive FSBMA architectures. The first step is to establish the dependence graph (DG) of FSBMA. In DG, a node represents a basic operation (absolute pixel difference), and an arc denotes data dependence. Second, a time schedule and assignment of multiple nodes to a single PE by projection are specified to provide a signal flow graph (SFG) of reduced dimension compared with DG. The time schedule must be carefully designed. Multiple nodes (operations) projected to the same PE should not be executed at the same time. Note that the DG is not unique. Different DGs, as well as different time schedules and projections, lead to different architectures. With a single projection, a 2-D DG and a 3-D DG can be transformed into a 1-D array and a 2-D array, respectively. DGs with dimensions higher than three have to be mapped on to systolic arrays by multiple projections. In this paper, two DGs were displayed, and two projections were attempted for each DG. The proposed architectures are AB1, AB2, AS1, and AS2. "A" denotes array, "1" represents 1-D, "2" stands for 2-D, "B" means that the number of PEs is in proportion to the block size (1-D: N; 2-D: N^2), and "S" indicates that the number of PEs is proportional to the search range (1-D: $2p$; 2-D: $N \times 2p$). The four architectures are fully systolic without any global routing, but the bitwidths of memory access are large.

In [74], Vos and Stegherr proposed a 2-D semi-systolic array with an adder tree. The most special idea is the scanning order of search positions, known as snake scan. The number of PEs is N^2, and the N^2 pixels of current block are stayed in the corresponding PEs. All pixels of a candidate block are also properly moved to the corresponding PEs. Each PE computes the absolute difference, add its own difference with the output of previous PE on the same row, and send the result to the next PE in the horizontal direction. The last PE on each row outputs the SAD of a row, and an adder tree calculates the final SAD of a candidate block. In the beginning, the first column of search positions is scanned from top to bottom. Then the second column is scanned from bottom to top, followed by the third column scanned from top to bottom, and so on. In order to successively compute SAD values without extra cycles to load pixels, two sets of $(2p-1) \times N$ registers are required to prepare the proper search area pixels beforehand, and the data path should be periodically configured as $2p$ cycles upward, one cycle leftward, $2p$ cycles downward, and one cycle leftward. The large register sets is a tradeoff for the bitwidth of memory access. The utilization of this architecture is higher (produces one SAD value of a search position per cycle) for large search range application. If $2p$ is smaller than N, there exist $(N-2p)$ bubble cycles when search positions are switched from one column to another.

In [75], Yang, Sun, and Wu implemented the first VLSI motion estimator in the world. The most important concept is data broadcasting. Two 1-D semi-systolic arrays were proposed based on data-flow designs which allow sequential inputs but perform parallel processing. One type is broadcasting reference frame data with current block data propagated through PEs while the other type is broadcasting current block data with reference frame data propagated through PEs. With broadcasting technique, although some global routing is required, the bitwidth of memory access in one cycle can be significantly reduced. The number of PEs is the same as the number of search positions in the horizontal direction. Each PE not only computes the absolute pixel difference but also accumulates the SAD of a search position. In order to achieve 100% utilization during the row change of search positions, two reference frame pixels are simultaneously fetched, and a multiplexer is required for each PE to select the proper data. Moreover, the chips can be cascaded for larger search range. A chip-pair design is also derived to obtain fractional MVs.

In [76], Hsieh and Lin focused on the decrease of pin counts (bitwidth of memory) and the flexibility of search range. The proposed structure is a 2-D systolic array and a shift register array. The number of PEs is N^2, and the N^2 pixels of current block are stayed in the corresponding PEs. Each row of PEs, except the bottom row, is followed by $(2p-2)$ shift registers (SRs), and the total number of SRs is $(2p-2)\times(N-1)$. The main purpose of SRs is to get the profit of serial data inputs. The length of SRs is programmable to support flexible search range. A search area pixel propagates from the PEs and SRs on the first row, and then to the second row of PEs and SRs, and so on. The PEs sends the partial results of SAD in the upward direction. Only one search area pixel is inputted in each cycle. It takes $(N+2p-1)^2$ to scan the whole search area, and the number of valid cycles for generating SAD values is $4p^2$. Lower PE utilization and the area of SRs are the tradeoff for bitwidth of memory access.

In [77], Jehng, Chen, and Chiueh proposed an efficient and simple tree architecture, which gave a completely different ME structure. The tree structure supports not only FSBMA but also fast algorithms by decimating search positions, such as three step search. Concepts of memory interleaving and pipeline interleaving were proposed to enhance the supported memory bandwidth and to prevent the hardware from idling, respectively. However, when the block size is enlarged, the required bitwidth of full tree will become too large. Therefore, subtree of $1/2^m$-cut was described to trade off between processing speed and memory bitwidth. Another solution of the bitwidth problem is to employ a search area pixel cache with snake scan of search positions. The inputs of the cache are 17 pixels and the outputs are 256 pixels. Related materials can be found in [78].

In [79], Chang et al. contributed an approach that employs transformation on a 4-D DG (represented by multiple 3-D sub-DGs), called slice and tile, to produce different forms of DGs. A 3-D sub-DG is partitioned into slices, and the slices of all sub-DGs are tiled in a direction. They categorized 2-D systolic arrays of FSBMA into six types. Type A and B are obtained by tiling slices in the diagonal direction with diagonal and horizontal projection, respectively. The numbers of PEs for type A and type B are $4\mathrm{p}^2$ and N^2, respectively, and the numbers of cycles required for processing a macroblock (MB) are N^2 and $4p^2$, respectively, assuming the PE utilization is 100%. Usually, type A is called inter-level parallelism in which every PE computes SAD for a candidate block, and type B is referred as intra-level parallelism in which every PE computes an absolute pixel difference in candidate blocks. Type C and D are obtained by tiling slices in the horizontal direction with diagonal and horizontal projection, respectively. Type E and F are achieved through the projection of the direct form DG along the horizontal direction of current block with different scheduling. The numbers of PEs and the ideal numbers of cycles for type C–F are $2p\times N$.

In [80], Yeo and Hu transformed the 1-D linear array of broadcasting reference frame data with $2p$ PEs in [75] to a $2p\times 2p$ 2-D basic mesh array. Current block pixels are propagated among PEs where the last PE of a row is connected to the first PE of the next row. The reference frame data are broadcasted not only in the horizontal direction but also in the vertical direction. A basic mesh array requires $2p$ to be equal to N. For large search range applications, multiple mesh arrays can be applied. The search range is partitioned into multiples of $N\times N$ search positions, and each partition is processed by one mesh array. Under typical specifications, this architecture results in the fastest speed and the fewest amount of memory access when compared with prior FSBMA hardware. However, the number of total PEs in the multiple mesh arrays is relatively high, and thus this architecture is more suitable for high-end applications.

In [81], Lai and Chen proposed a 1-D PE array and two data-interlacing SR arrays that utilize 2-D data-reuse. In fact, it can be regarded as an extension of the 1-D linear array of broadcasting current block data in [75]. The main difference from [80], which is the extension of broadcasting reference frame data in [75], is that search range partition is not always necessary. The PEs can be cascaded when the number of PEs reaches the number of search positions. In this situation, every reference frame pixel is fully reused and is only fetched from memory to PEs for one time. Nevertheless, the MB latency of generating a motion vector will become the number of pixels in the search area, which is quite large. With search range partition, the PE number is still the same, but the reference frame pixel reuse becomes incomplete, which increases the memory bandwidth, as a tradeoff for shorter MB latency.

In [82], Yeh and Lee found that overlapped data flow can increase the PE utilization for 2-D arrays where current block stays (intra-type). Search area data from two different rows are needed as a boundary candidate is detected, and thus two inputs are requested. A stream memory similar to the SRs in [76] is adopted to reduce the memory bandwidth. However, unlike [76], two search area pixels are propagated simultaneously. During the row change of search positions, no bubble cycles exist, so the utilization is much higher. Based on the overlapped data flow and the stream memory, a systolic 2-D array (SA architecture) and a semi-systolic 2-D array (SSA architecture) were devised to collaborate with N-parallel adder tree for FSBMA.

In [83, 84], four levels of search area data reuse were discussed. Let $N \times N$, $[-p, p-1$, $W \times H$, fr be the block size, search range, image size, and frame rate, respectively. Level A is the reuse of the $N \times (N-1)$ overlapped pels between two horizontally adjacent candidate blocks. Level B is the reuse of the $(N+2p-1) \times (N-1)$ overlapped pels between two vertically adjacent candidate block strip. Level C is the reuse of the $(2p-1) \times (N+2p-1)$ overlapped pels between the search areas of horizontally adjacent MBs. Level D is the reuse of the $(W+2p-1) \times (2p-1)$ overlapped pels between vertically adjacent search area strips. In today's VLSI technology, on-chip SRAMs can be as large as several tens of Kbytes. Therefore, level C is the most popular scheme. The search area pels are buffered in the on-chip memory. For the left most MBs in the image, the whole search area ($(N+2p-1) \times (N+2p-1)$ pels) is reloaded from external DRAM to on-chip SRAM. For the rest MBs, only part of the search area ($N \times (N+2p-1)$ pels) is loaded. The bus bandwidth can thus be reduced from $((N+2p-1) \times (N+2p-1)) \cdot (W/N) \cdot (H/N) \cdot fr$ to $(N \times (N+2p-1)) \cdot (W/N) \cdot (H/N) \cdot fr$, that is, $(2p-1)/(N+2p-1)$ of the bus bandwidth is saved. Currently, level D is impractical due to the limited SRAM size. If level D becomes feasible in the future, the bus bandwidth for loading search area pels will be reduced by a factor of five. Level D scheme is also beneficial to power-limited systems because external DRAM access consumes more than tens times higher of power than on-chip SRAM.

There still exist many other FSBMA implementations. For example, reference [85] depicted a type D architecture and implemented the FSBMA with discrete TTL components. Reference [86] combined the architecture in [76] with the MSEA to avoid the unnecessary SAD computations for low power consumption. Reference [87] designed a powerful FSBMA chip with 1024 PEs to provide computational capability of 165 GOPS. Reference [88] modified the 1-D linear array in [75] to support half-pixel precision, AP mode, PB mode, and RRU mode for H.263+. Reference [89] improved the architecture in [74] by changing the snake scan of search positions to regular column scan (top to bottom and then left to right) and using a circular shift scheme for PEs with half-reduced SRs. Reference [90] worked on an architecture design of variable block sizes ME for H.264/AVC.

According to our survey, PE array is the trend of FSBMA architectures. Tradeoffs can be made between area (number of PEs) and throughput (processing capability), SAD latency (total cycles to compute a SAD) and memory bitwidth/bandwidth (serial/parallel loading), PE utilization and data alignment circuits (shift registers/memory with circular addressing), bus bandwidth and on-chip SRAM size. The designers have to carefully select what can be sacrificed and what must be insisted on for target applications.

Usually, MB latency (cycles/MB) is defined as the duration from the start of processing an MB to the end of finding its motion vector. Throughput (MBs/sec) is denoted as the reciprocal of the timing interval between generating two MVs. For intra-type FSBMA architectures, the timing interval is the same as MB latency. For inter-type designs, the PE array starts processing the next MB before finishing one MB. The MB latency will be much larger than the timing interval of generating two MVs if the search range is much larger than an MB. That is, the schedule of an inter-type design is often arranged to process multiple current blocks in parallel. However, such schedule is not very suitable for MB pipelining, which is often adopted in a hardware accelerated video coding system. When an inter-type design is integrated into MB pipelining systems, the claimed throughput and utilization will be dropped. For example, two-stage MB pipelining handles two MBs at the same time. When one new MB is processed at ME stage, its previous MB is processed at block engine $(\text{BE} = \text{DCT} + \text{Q} + \text{IQ} + \text{IDCT} + \text{VLC})$ stage. Before finishing all the operations for one MB, every stage cannot start processing the next MB. Hence, system designers should pay more attention on the MB latency of inter-type architectures than on the claimed throughput.

We derive hexagonal plots to evaluate representative FSBMA architectures including Komarek and Pirsch's AB2 [73], Vos and Stegherr's [74], Yang, Sun, and Wu's broadcasting reference frame [75], Hsieh and Lin's [76], Yeo and Hu's [95], and Lai and Chen's [81]. While [75, 80, 81] are inter-type architectures, AB2 of [73, 74, 76] are intra-type. The results are shown in

Fig. 1. Gate count, required frequency, hardware utilization, memory bandwidth, memory bitwidth, and SAD latency are the six dimensions of the hexagonal plots. The search range is H[−64, +63]/V[−32, +31]. The required frequency is estimated for CIF 30 Hz video. Both the required frequency and hardware utilization are derived with integration of ME into MB pipelining systems. The "latency" in the plot is SAD latency instead of MB latency. The inverse of MB latency is proportional to the required frequency, so we do not have to plot it again. The SAD latency is the timing interval from start processing an MB to finish accumulating the first SAD value. For each dimension of the hexagonal plots, the performance is better (or worse) when the curve approaches vertex (or center). For example, Yeo and Hu's inter-type 2-D broadcasting reference frame architecture is good in required frequency, memory bandwidth, and latency. However, its gate count, memory bitwidth, and utilization are the tradeoffs. These hexagonal plots provide system designers with a quick evaluation of architectures. The detailed data of these designs are shown in Table 3.

3.2. *Fast BMA Architectures*

Fast ME algorithms can reduce the heavy computation burden of FSBMA with acceptable video quality. The challenges of architecture design for fast ME algorithms include unpredictable data flow, irregular memory access, difficult mapping to systolic arrays, low hardware utilization, and sequential procedures with data dependence that cannot be parallelized. Besides, the silicon area of fast ME architectures must be significantly smaller than that of FSBMA architectures for cost efficiency considerations. In this subsection, we will survey the previous arts.

In [91], Jong et al. developed a fully pipelined parallel architecture for three step search BMA. Basically, nine PEs compute the SAD of the nine candidates in each step, and 256 cycles are required for each of the three steps. Intelligent data arrangement and memory arrangement are used to completely utilize the advantage of three step search. The proposed architecture can be extended to 27 PEs with MB pipelining to triple the throughput of blocks. The latency of 27-PE design is still the same as 9-PE design. The proposed architecture can be also reduced to 3 PEs with one third of the throughput and three times of the latency. The 3-PE, 9-PE, and 27-PE designs all have computation efficiency close to 100%. Compared with a 256-PE FSBMA array, the gate count of 9-PE architecture is significantly smaller (36.6 K vs. 192.2 K). When 27-PE design is used, the throughput is about the same as a 256-PE FSBMA array, and the area (110 K) is still much smaller.

In [92], Dutta and Wolf modified the data flow of the 1-D linear array in [75] to support FSBMA, three step search, and conjugate direction search on the same architecture. Multiple memory banks are organized in communication with PEs via a multistage interconnection network. When three step search is selected as the target BMA, the throughput is eight times of FS-BMA. The programmability makes the 1-D linear array suitable for more applications with different timing or power constraints.

In [93], Lin et al. proposed a joint algorithm-architecture design of a programmable motion estimator chip. Various algorithms are implemented through a search strategy with macrocommands that can be executed efficiently on the chip. Two types of programmable mechanism are supported. One is subsampled search positions and/or block pixels, and the other is the cluster search. Because the programmable ME requires executing macrocommands interactively as opposed to executing fixed search patterns in batches, the latency in computing the SAD values must be kept low. Therefore, serial array architecture is not suitable, and parallel 2-D array is chosen. Conventionally, multiple banks of SRAMs with bank selection and window shift are required as an interface between SRAMs and the parallel 2-D array to perform data alignment. In this paper, a synchronous self-align array composed of pixel-rotating PEs, together with a dual-addressing single-port memory, achieves data alignment without the complex interface. A prototype chip was implemented. The maximum computational power is 14 GOPS, and it is enable to deliver full search quality at CIF 30 fps, near full search quality at NTSC level, and a wide range of other video quality and resolution rasolution tradeoffs.

In [94], Cheng and Hang used a universal systolic arrays structure to realize many BMAs. In summary, they found that the relative performance in chip area and I/O bandwidth between various algorithms is strongly picture size- and search range- dependent. For small pictures and slow motion, all the BMAs under consideration are on a par. For larger picture sizes and fast motion, certain fast algorithms have significantly

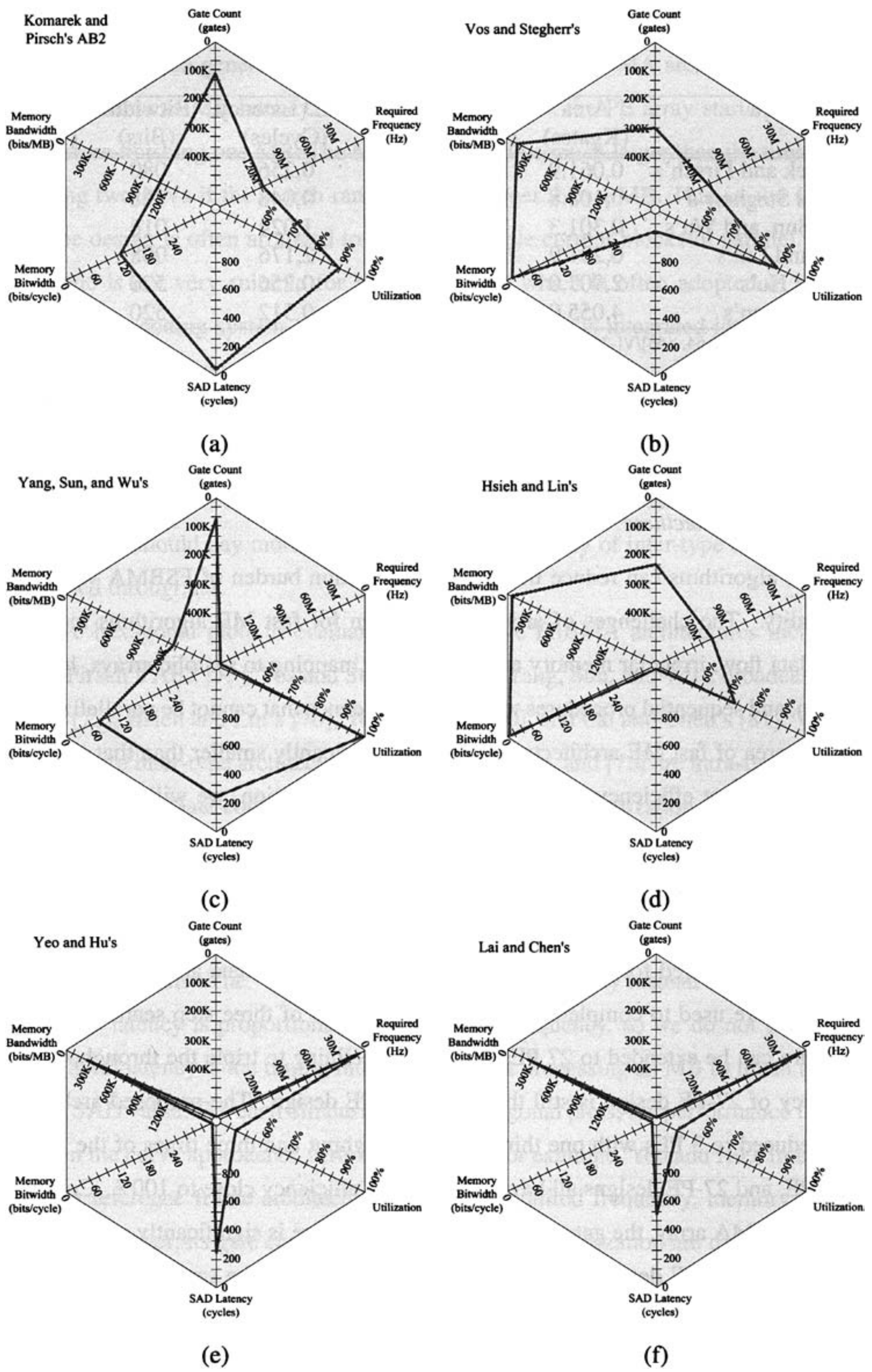


Figure 1. Hexagonal plots to evaluate different architectures for CIF 30 Hz video with search range as H[−4, +63]/V[−2, +31]; (a) Komarek and Pirsch's AB2; (b) Vos and Stegherr's; (c) Yang, Sun, and Wu's; (d) Hsieh and Lin's; (e) Yeo and Hu's; (f) Lai and Chen's.

Table 3. Comparison of FSBMA architectures.

Architecture	Area (Kgates)	Freq. (MHz)	Util. (%)	SAD Latency (Cycles)	Bitwidth (Bits)	Bandwidth (Kbits/MB)
Komarek and Pirsch's	0,061.9	196.0	99.2	0.256	080	1.290
Vos and Stegherr's	0,108.8	108.9	89.3	0.048	128	1.146
Yang, Sun and Wu's	0,301.3	112.3	88.9	1.024	016	0.146
Hsieh and Lin's	0,231.9	109.4	72.5	2.176	008	0.090
Yeo and Hu's	2,907.0	006.1	50.0	0.256	520	0.260
Lai and Chen's	4,055.0	006.1	50.0	0.512	520	0.260

CIF 30 Hz video, H[−64, +63]/V[−32, +31].

smaller chip areas. In fact, if a more efficient dedicated hardware is designed for a certain fast algorithm instead of using systolic arrays, fast algorithms may even result in much smaller chip area for low-end applications, like the three step search architecture developed in [91]. Their analysis provides useful guidelines to system designers in choosing a suitable high-level algorithm for VLSI implementation.

In [95], Minzuno et al. pointed out that conventional ME designs with pipeline/parallel processing must decide a search window size in advance, since the designs are optimized in accordance with the search window size. The efficiency needed to achieve the performance cannot be maintained when the search window size varies. This lack of flexibility is a significant drawback. Hence, they developed a motion estimation algorithm and a hardware implementation method with which the search window size can be varied without degrading pipeline/parallel processing efficiency. Two-step hierarchical search scheme is employed. The first phase determines coarse motion vectors with a precision of two pel in the horizontal direction and one pel in the vertical direction. The second phase is performed at 5×3 points in the center of the area determined by the first phase. The final MVs are obtained within a precision of half pel. The quality loss for edge-sharpness sequences is 0.13 dB and for typical sequences is 0.03 dB. The search range of ME is −48/+47 horizontal and −16/+15.5 vertical. The pseudo search range, which is the size when the location of the search window is adaptively shifted, is −96/+95 horizontal and −32/+31.5 vertical and improves 0.4/0.8 dB of coding performance. Other hierarchical ME architectures can be found in [45, 96]. Three-level hierarchy was adopted instead. In [95], tasks of different levels are executed in separate processing units, so MB pipelining can be utilized for tasks of different levels. On the contrary, the architecture in [45] iterates a basic searching unit for different levels of block matching to save the chip area.

In [97], Moshnyaga proposed a fast algorithm from observing that the size of the search window can be gradually reduced during processing based on the level of picture distortion between the current block and the candidate block. After i rows of SAD are accumulated for a candidate, if the partial SAD value exceeds a threshold, the candidate can be eliminated at row $i + 1$, $i + 2, \ldots$, with all the corresponding operations. The 1-D linear array [75] and the AB2 [73] are taken as examples to fit the adaptive scheme. However, only the power consumption can be reduced, but the chip area and the latency for an MB is still the same due to the already established data flow. The threshold modification algorithm was described and claimed to save up to 75% of the operations whilst preserving the high quality.

In [98], Hsia adopted temporal prediction of MV with refinement in a significantly smaller search window. Only eight PEs with 8 K gates are designed to achieve throughput of 53 K MBs/s. The produced MVs may cover the range of [−127, +127]. An adaptive −7/+7 full search centered at temporal MV predictor is adopted. The claim that the proposed algorithm is better than FSBMA in video quality is misleading because it was only compared with −7/+7 full search centered at the origin. In fact, compared with −127/+127 FSBMA centered at the origin, the quality loss of this design will become very large. For example, in Football sequence, the camera pans fast toward left at frame 1–70 and then pans fast toward right. In this case, the predictive search method suffers more than 2 dB PSNR degradation.

In [99], Kawahito et al. combined ME with CMOS sensors. In the future, CMOS sensor will become more and more popular than CCD sensors due to the easier integration with other CMOS components to achieve system on chip (SOC). By utilizing high-speed intermediate pictures from the sensor, an adaptive iterative-search BMA was proposed. The concept is very simple. Since the frame rate of intermediate pictures is much higher, the motion between consecutive intermediate pictures is also much smaller. ME can be performed on the intermediate pictures with much smaller search range, and these MVs can be used to composite the MVs of pictures with normal frame rate. For instance, let the original frame rate and search range be 30 and $[-64, +63]$, respectively. If the frame rate is increased to 480, the search range can be reduced to $[-4, +3]$. Assume the same BMA is applied for both frame rates, the ratio of computational complexities is $128^2 \times 30{:}8^2 \times 480 = 16{:}1$. In addition to the reduced complexity, memory access is also reduced due to the shrink of search range.

In [100], Vleeschouwer et al. proposed a directional squared-search BMA. It is similar in performance to other up-to-date fast BMAs belonging to decimation of search positions and predictive search, which is not a surprise as it exploits the same under-lying principles including spatial correlation of MVs, center-biased distribution of MVs, faster convergence toward an optimum, and early termination when the distortion is small enough. However, the algorithm is designed with architectural considerations such that the data reuse of adjacent candidate blocks can be utilized. Three PEs are used for three horizontally/vertically adjacent candidates. According to their experimental data, the video quality is worse than diamond search and the complexity is higher, but their algorithm has more efficient memory access for hardware. Unfortunately, no report of implementation or simulation result is available in the paper, so comparison with other papers cannot be made.

In [102], Chao et al. contributed a novel hybrid motion estimator supporting advanced zonal diamond search [19] and fast full search (SEA) [47]. The design target was CIF 30 fps with search range of $[-16, +15]$. The search area pixels are buffered in the on-chip SRAM for level C data reuse. Search area pixels on different columns are interleaved in eight banks of SRAMs so that any 8×1 pixels in the search area can be accessed in one cycle. An eight-PE SAD tree [77] is adopted to compute a SAD in 32 cycles. When diamond search mode is selected, some candidates will be searched for more than one time without checking. To avoid the computation redundancy, 1024 one-bit flags are conventionally used to record if search positions have been searched. The average saved computation is 24.43%. In this paper, the authors used a ROM-based method to determine the search positions of the next step, and the 1024 flags are omitted. Although it is still possible to search a candidate twice, the saved computation is 24.23% statistically, which almost reaches the conventional way. PDE is also applied in the design to shorten the processing cycles with a comparator to compare the accumulated partial SAD with SAD_{min}. When the design is configured as SEA mode, extra circuits to decide if the SAD calculation of a search position is necessary will be activated. The decision is made for every search position in one cycle. However, if skipping condition does not hold, the number of cycles to compute an SAD is 32, and the SAD decision circuit must be stalled. Therefore, a FIFO is inserted between the SAD decision circuit and 8-PE SAD tree to increase the throughput. This architecture has been successfully integrated into an MPEG-4 simple profile level 3 VLSI encoder and the ME gate count is only 9 K. The extension of this architecture to a computation-controllable version with half/quarter pel accuracy can be found in [102].

Based on our survey, the trend of fast BMA architectures is algorithmic and architectural co-development. The benefits from the algorithmic level are usually larger than those from the architectural level. Not only the traditional algorithmic issues, such as convergence speed and avoidance of being trapped in local minima, but also the architectural issues should be taken into consideration. For example, searching eight random candidates requires memory access of $16 \times 16 \times 8 = 2048$ pixels, but searching eight successive candidates requires to access only $16 \times (16 + 7) = 368$ pixels, which suggests the line search pattern is more efficient in memory access. Serial systolic array architectures, though may have high throughput by many cascaded PEs, requires a long latency to load pixels, which will lead to low throughput and low utilization for fast algorithms whose next-step-candidates cannot be known in prior. Thus, architectures with parallel loading of reference frame pixels, though require a wider memory bitwidth, are more suitable for fast BMAs. Besides, algorithms utilizing sequential processing candidates by candidates, which may cause an upper bound of throughput that cannot be improved by parallelism, should be avoided.

4. Proposed New Design

In [103], we proposed a global elimination algorithm (GEA) to remove the branches of SEA/MSEA. The data flow becomes more regular for hardware. The video quality is almost the same as that of FSBMA. The corresponding GEA architecture was also developed. Compared with FSBMA architectures, we concluded that our GEA design is much more area-speed efficient. The processing capability of ours is about the same as 256-PE intra-type 2-D array while our gate count is five times smaller. However, the drawback is the longer critical path. It is difficult to meet the real-time requirement for high specifications. Recently, we have developed a new parallel GEA and its corresponding architecture to solve the encountered problem, which will be described in detail as follows.

4.1. Global Elimination Algorithm

The original GEA is described as Eqs. (3)–(9).

$$-p \le (m,n) \le p-1 \tag{3}$$

$$0 \le (i,j) \le 2^L - 1 \tag{4}$$

$$CS_{L,j,i} = \sum_{yc=ya}^{yb} \sum_{xc=xa}^{xb} C(xc, yc)$$
$$0 \le (xc, yc) \le N-1 \tag{5}$$
$$y_a = j \cdot N/2^L, \quad y_b = y_a + N/2^L - 1$$
$$x_a = i \cdot N/2^L, \quad x_b = x_a + N/2^L - 1$$

$$\mathrm{SS}_{L,j,i,n,m} = \sum_{ys=y_c}^{yd} \sum_{xs=x_c}^{xd} S(xs, ys)$$
$$0 \le (xs, ys) \le 2p + N - 2 \tag{6}$$
$$y_c = n + p + j \cdot N/2^L, \quad y_d = y_c + N/2^L - 1$$
$$x_c = m + p + i \cdot N/2^L, \quad x_d = x_c + N/2^L - 1$$

$$\mathrm{SSAD}(m,n) = \sum_{j=0}^{2^L-1} \sum_{i=0}^{2^L-1} |CS_{L,j,i} - SS_{L,j,i,n,m}| \tag{7}$$

$$(m_i, n_i) \in \{(m,n) \,|\, \mathrm{SSAD}(m,n) \le \mathrm{SSAD}_M\} \tag{8}$$

$$\mathrm{MV} = \{(u,v) \,|\, \mathrm{SAD}(u,v) \le \mathrm{SAD}(m_i, n_i)\}$$
$$\mathrm{SAD}(m,n) = \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} |C(x,y) - S(x+m+p, y+n+p)| \tag{9}$$

The search range is $[-p, p-]$, (m,n) denotes a search position, and (i,j) is the subblock index. Level is indicated by L, and a block of size $N \times N$ is divided into $2^L \times 2^L$ subblocks of size $(N/2^L) \times (N/2^L)$. The current block data and the search area data are denoted as C and S, respectively. CS is the sum of all pixels within a subblock in current block, and SS is the sum of all pixels within a subblock in a candidate block. Originally, the matching criterion is sum of absolute differences (SAD) for all pixels in the block. Here, we define subsampled-SAD (SSAD) as sum of absolute differences between CS and SS. After all the SSAD(m, n) values are calculated, we will find the most probable M motion vectors (m_i, n_i) whose SSAD values are the smallest. The M-th smallest SSAD among all candidate blocks is denoted as SSAD_M. Finally, we compute the SAD at the M search positions to find the final motion vector (MV). In [103], we found that $L = 2$ and $M = 7$ are suitable parameters for CIF and QCIF under $N = 16$ and $p = 16$ or $p = 32$.

4.2. Problem Statement

The previously proposed architecture is composed of a systolic module and a 16-pel SAD tree to efficiently calculate SSAD and SAD, and a comparator tree to record the M most probable motion vectors. The comparator tree is designed to match the throughput of generating SSAD values, so the critical path of the comparator tree is roughly proportional to $\log_2^{M+1}$. However, for high-end applications with larger frame size, the search range and the M parameter should be enlarged (e.g. $p = 64$, $M = 15$ or 31) to obtain FSBMA quality. Moreover, our previous architecture computes SSAD sequentially (one SSAD per cycle), so the operating frequency must be significantly increased with search range and frame size to an unacceptable degree. Consequently, parallel algorithm and architecture with short critical path are demanded.

4.3. Proposed Parallel GEA

In order to compute the SSADs of several candidate blocks in parallel, we divide them into P groups.

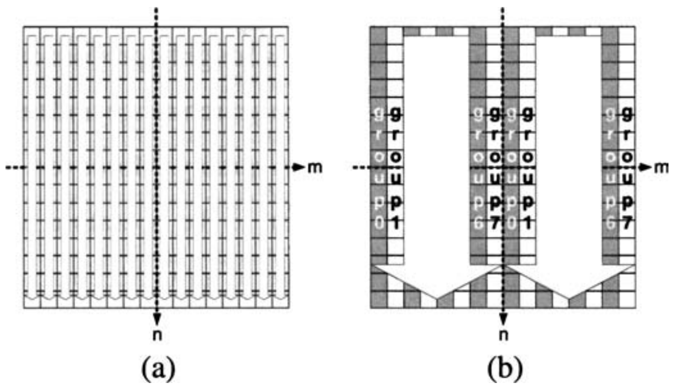


Figure 2. Scanning order of search positions for SSAD calculation: (a) sequential GEA; (b) parallel GEA with $P = 8$.

Candidate blocks with the same value of $m\%P$ are grouped together, and the the most probable K motion vectors with the smallest SSADs are found separately for each group. Hence, after all the SSAD values are estimated, SAD values of the $K \cdot P$ search positions are further computed to get the final motion vector. Although the $K \cdot P$ most probable candidates do not correspond to the $K \cdot P$ smallest SSAD values within the whole search range, the parallel GEA does not suffer noticeable quality degradation because the $K \cdot P$ globally smallest SSAD values usually belong to different groups. The collection of K candidates in each of the P groups should be similar to the $K \cdot P$ candidates with globally smallest SSAD values. In this way, P duplications of the original GEA architecture can be configured as an array of GEA-PEs to support parallel scanning of search positions and parallel calculation of SSAD values. Figure 2 illustrates the scanning order. Besides, K is much smaller than $K \cdot P$, which indicates that the critical path of comparator tree in each GEA-PE can be reduced at the algorithmic level.

Many conditions have been tested to verify the quality of our parallel GEA. In our experiments, we embed parallel GEA with $P = 8$ and $K = 3$ into an MPEG-4 simple profile encoder. The resolution of CS and SS is truncated from 12-bit to eight-bit in order to save more area and to reduce the critical path for hardware. The other parameter sets are {CIF 30 fps [-32, $+31.5$] 384/2048 Kbps} and {D1 30 fps H[-4, $+63.5$] V[-32, $+31.5$] 1536/8192 Kbps}. CIF sequences are Foreman, Hall Monitor, Mobile Calendar, Stefan, and Table Tennis. D1 sequences are two clips from the movie, Crouching Tiger Hidden Dragon. One clip is the scene with two actresses fighting against each other in the courtyard, and the other clip is the leading actor chasing the leading actress on bamboos. Compared with FSBMA, the average PSNR losses for the seven sequences are only 0.16, 0.13, 0.05, 0.00, 0.14, $-.02$, 0.05 dB, respectively. Note that Lagrangian mode decision [104] is applied for both BMAs.

4.4. *Proposed Parallel GEA Architecture*

In the following, $N = 16$, $L = 2$, $P = 8$, $K = 3$, and H[-64, $+63.5$] V[-32, $+31.5$] are used as an example to explain the parallel GEA design. The specification is D1 30 fps.

The purpose of the systolic module is to generate 16 subblock sums of 4×4 pixels in parallel. As shown in Fig. 3, the input is a row of 16×1 pixels. After consecutive 16 rows of pixels are inputted, the 16 subblock sums at search position $(m, -p)$ are produced. The systolic module utilizes vertical data reuse, and thus the subblock sums at the search positions $(m, -p+1)$ $(m, p-1)$ can be obtained in the following $(2p - 1)$ cycles. Compared with the original systolic module in [103], the improved systolic module not only removes the redundant computation of subblock sums but also reduces the resolution of subblock sums from 12-bit to eight-bit. The gate count of this part is reduced from 6.0 to 4.7 K.

The SAD tree is illustrated in Fig. 4, and the goal is to compute SSAD/SAD values. An AD unit computes the absolute difference of two eight-bit samples. When SAD tree is used to generate SSAD values, the inputs are 16 subblock sums of current block and 16 subblock sums of a candidate block. The throughput is the same as the systolic module, i.e. one candidate block per cycle (except the first candidate block at each column

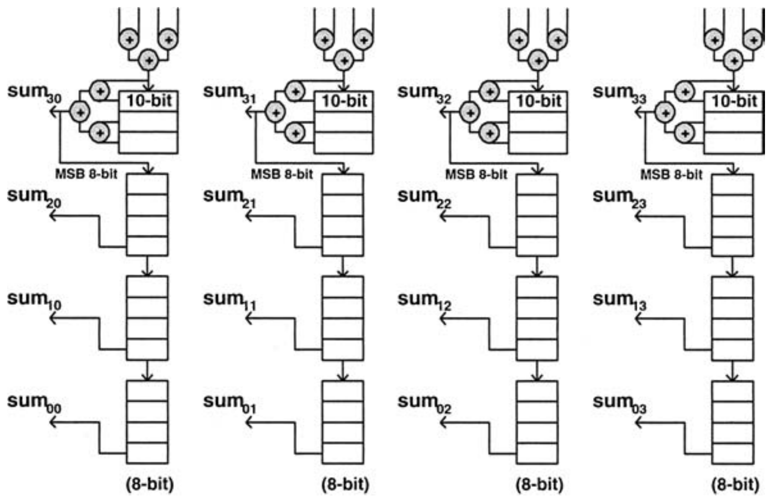


Figure 3. Systolic module to generate 16 subblock sums of 4×4 pixels.

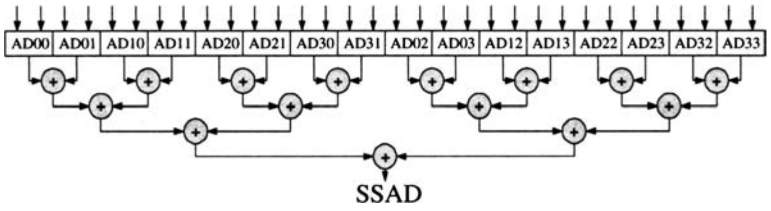


Figure 4. SAD Tree to compute SSAD/SAD.

of search positions). When SAD tree is used to compute SAD values, the inputs are rows of current block data and search area data, and its output is fetched to a 16-bit accumulator. It takes 16 cycles for one candidate block to compute SAD. Due to the bitwidth reduction of CS and SS, the gate count of this part is reduced from 4.6 to 2.8 K.

The purpose of the comparator tree is to find the three smallest SSAD values among one group of candidate blocks. The throughput is also matched with the systolic module and the SAD tree. The concept is to keep the up-to-date three smallest SSAD values and their corresponding MV's in the registers, compare the new coming SSAD value with the three stored values, and replace maximum stored values by the new SSAD if it is larger than the new one. Figure 5 illustrates the comparator tree. The MAX unit outputs the larger value of its two inputs, and the EQU unit tells if the two given inputs are the same. The previous architecture shown in [103] finds the maximum SSAD value and feed it back to compare with three stored values to see if a stored value should be replaced. Then, a CHECK unit is to ensure that only one stored value will be replaced if more than one stored values are equal to the maximum. Stall signal should be activated when the invalid SSAD value is generated from SAD tree for the first 15 cycles of a column of search positions. We shorten the critical path in three aspects. First, at the algorithmic level, search positions are divided into eight groups. Originally, if $M = 24$, we will have to find the 24 smallest values, but now only three smallest values in each group are required. Second, the bitwidth of SSAD is reduced from 16-bit to 12-bit. Third, as shown in Fig. 5, instead of feeding the maximum SSAD value back to compare for replacement, we give each SSAD value an unique 2-bit tag and feed the tag with the maximum SSAD back for comparison. The gate count of the comparator tree is reduced from 1.5 to 1.1 K, compared with the previous version [103] for $M = 3$.

Figure 6(a) illustrates a GEA-PE. The systolic module, SAD tree, MV cost generator, and comparator tree are configured in cascoding. The MV cost generator, which requires only 0.6 K gates, adds a bias of motion information to the distortion function (known as Lagragian method [104]) and provides additional coding gain of 0.2–1.0 dB in PSNR for the MPEG-4 simple profile encoder. The gate count of a GEA-PE is 11.3 K.

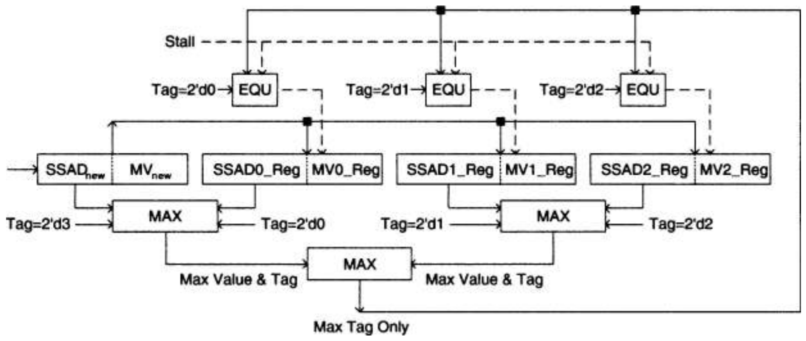


Figure 5. Comparator tree to find the three smallest SSAD values.

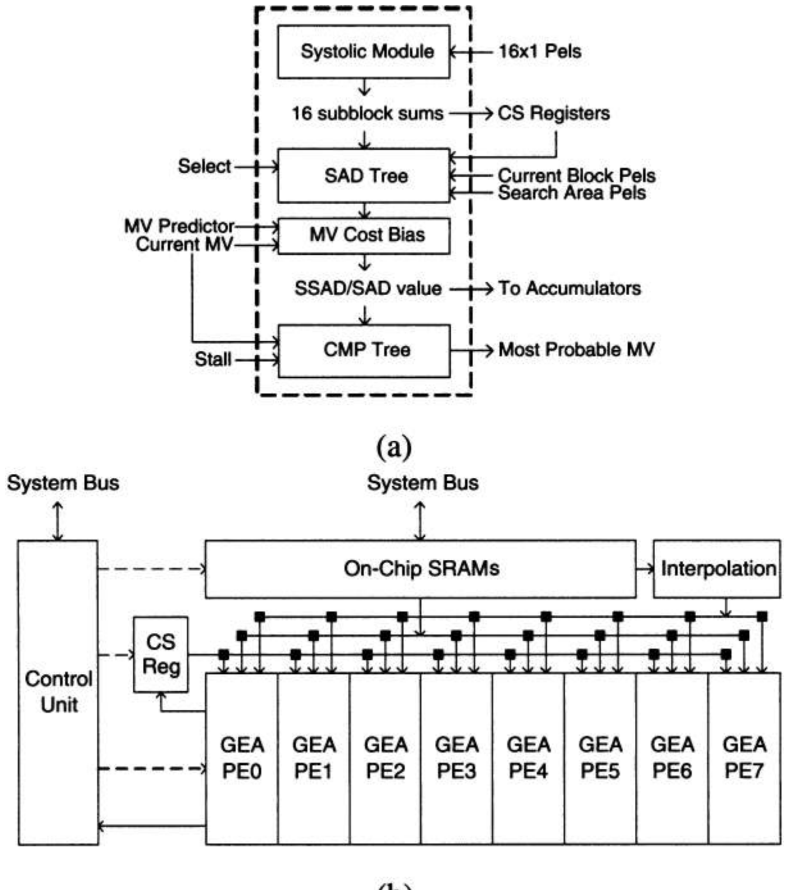


Figure 6. Illustration of the motion engine: (a) GEA processing element; (b) system block diagram.

Figure 6(b) illustrates the entire ME accelerator. Current block data and search area data are loaded from external SDRAM to on-chip SRAMs. The SRAMs behave as a cache of the ME processor, and the bus bandwidth is reduced from more than 7 Gbytes/sec to 477 Mbytes/sec. We adopt (level C) data reuse of overlapped search area between two horizontally adjacent MBs to further reduce the bus bandwidth from 477 Mbytes/sec to 71 Mbytes/sec. The interpolation circuit is used to generate half-pixels. Besides, thanks to the versatility of SAD tree, advanced prediction (AP) mode (four 8×8-MVs for an MB) is also supported. Inter mode selection between 16×16 and 8×8 configurations of an MB is done after half-pixel ME, and intra/inter mode decision is also included in the accelerator. In general, an MPEG-4 simple profile encoder with our ME accelerator provides 0.5 dB better coding performance than the reference software using FSBMA.

4.5. *Discussion and Comparison*

The sequential GEA only utilizes the data reuse in the vertical direction by systolic module to compute the SSAD values. For one column of 64 search positions, 79 rows of 16×1 pixels are fetched, and 1264 bytes of memory access are required. The parallel GEA utilizes not only the vertical but also the horizontal data reuse. As mentioned before, SSAD values of eight columns of search positions are generated in parallel. In order to achieve parallel SSAD calculation, 79 rows of 23×1 pixels (1817 bytes) are fetched. Let us denote the fetched 23×1 pixels from left to right as $p0$–$p22$. The first 16 pixels, $p0$–$p15$, are sent to PEO, $p1$–$p16$ are sent to PE1,..., and $p7$–$p22$ are sent to PE7. In this way, compared with the sequential GEA (1264 bytes for one column), parallel GEA is much more efficient in on-chip SRAM access (1817 bytes for eight columns, i.e. 227 bytes for one column on average). The bandwidth of on-chip SRAM for SSAD computation is reduced from 6.55 Gbytes/sec to 1.18 Gbytes/sec.

The numbers of cycles to compute CS values, SSAD values, integer-pel SAD values, and half-pel SAD values are 16, $(64 \times 15) \times 128/8 = 1264$, $16 \times 24 = 384$, and $(16 + 2) + (8 + 2) \times 4 = 58$, respectively. Therefore, for processing an MB, about 1722 cycles are required (including pipelines and mode decision). For D1 30 fps, there are 40,500 MBs in a second, so the required frequency is 69.741 MHz. We use one 16×128 (words $\times$ bits) single-port SRAM to store current block data (16×16 pels) and eight 400×32 single-port SRAMs to store the search area data (160×80 pels). If the bus bitwidth is 32, loading current block data and search area data from external DRAM to on-chip SRAM requires about 500–700 cycles in average, depending on the bus traffic and protocol. Therefore, the frequency must be increased to 100 MHz. If dual-port SRAMs can be used, the loading operations of the next MB and the ME operations of the current MB can be executed at the same time, and no extra cycle is needed. Nevertheless, dual-port SRAMs are larger than single-port SRAMs, which is the tradeoff for speed.

In Fig. 6(b), there are eight GEA-PEs in parallel. Since the utilization rates of different modules vary a lot, the inputs of the idling circuits should be masked to zero for power saving. When the ME engine is configured to compute CS values, only one “Systolic Module” should be activated, and the rest of circuits can be shut down. When the ME engine is configured to compute SSAD values, all the logic circuits except interpolation will be activated. When the ME engine is configured to compute integer-pel SAD values of the 24 potential candidates, five ‘SAD Tree’ and ‘MV Cost Bias’ circuits (one for 16×16-block, four for 8×8-blocks) will be on. When the ME engine is configured to compute half-pel SAD values of one 16×16-block and four 8×8-blocks, only the interpolation and one ‘SAD Tree” and ‘MV Cost Bias” circuits will be used.

For the sake of clarity, GEA-PEs are used to explain the parallel processing for multiple columns of search positions. However, there exist redundant computations of SS values. For example, when pixels of 8×1 candidate blocks are inputted row by row (23 pels by 23 pels), the second column 4×4-block sum of the first candidate block is exactly the same as the first column 4×4-block sum of the fourth candidate block. To avoid the redundant SS computation of the 8×1 candidate blocks, the ME engine can be modified as Fig. 7. First, “Systolic Module” is removed from a GEA-PE, as shown in Fig. 7(a). Second, a systolic column PE is used to compute four 4×4-block sum with one inputted 4×1-pel sum at every cycle, as shown in Fig. 7(b). Third, for replacing the original eight “Systolic Module” circuits in GEA-PEs, we integrate 24 systolic column PEs and additional adders to compute 4×1-pel sums, as shown in Fig. 7(c). Finally, the modified system block diagram is shown in Fig. 7(d). In this way, about 15 K gates can be saved, and the overall logic gate count in Fig. 7(d) is reduced from 113 to 98 K (synthesized by Synopsys Design Compiler with Artisan 0.18 μm cell library at 70 MHz).

In order to compare our implementation with the hexagonal plots in Fig. 1, the specification is lowered from D1 30 fps to CIF 30 fps to get the scaled hexagonal plot of our eight-parallel GEA design, as shown in Fig. 8. With the moderate performance in memory bitwidth and utilization, our design performs

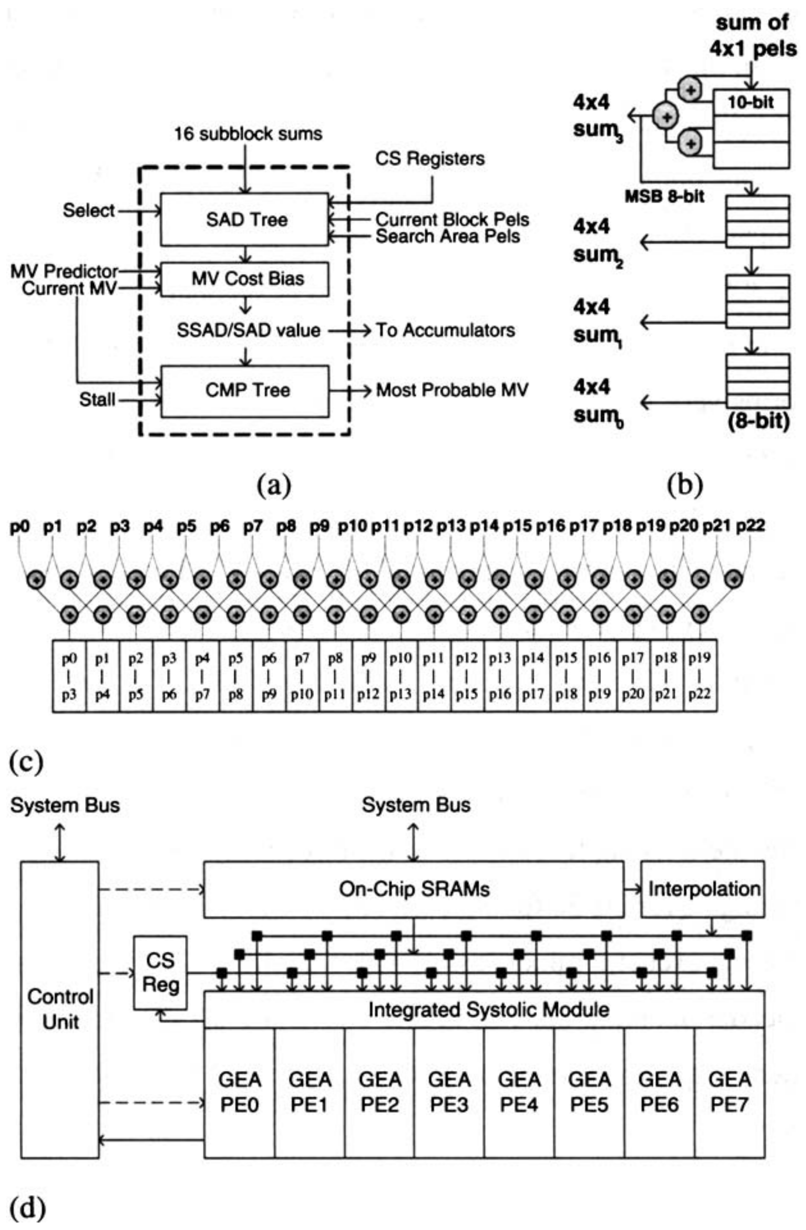


Figure 7 Illustration of the modified motion engine: (a) modified GEA processing element; (b) systolic column processing element (c) integrated systolic module with systolic column processing elements (d) modified system block diagram.

very well in the other four dimensions. Besides, the functionality of our ME engine is quite rich. Not only integer-pel ME, but also half-pel ME, AP mode, MV cost bias, intra/inter MB mode decision are included. It is a good silicon intellectual property (IP) and has been integrated into several commercial MPEG-4 VLSI coding systems.

Next, we compare our implementation with 1-D semi-systolic FSBMA array architecture [75] due to its high flexibility of search range and scalability of PEs. The results are shown in Table. 4. Search range partition is adopted for [75] to achieve the scalable PEs. The gate count of our design is 4.57, 9.14, and 18.29 times smaller than 512-PE, 1024-PE, and 2048-PE array, respectively, while the working frequencies of the three array configurations are 2.37, 1.19, and 0.6 times of our design's for D1 30 fps H[−4, +63] V[−2, +31]. Apparently, our design is more efficient

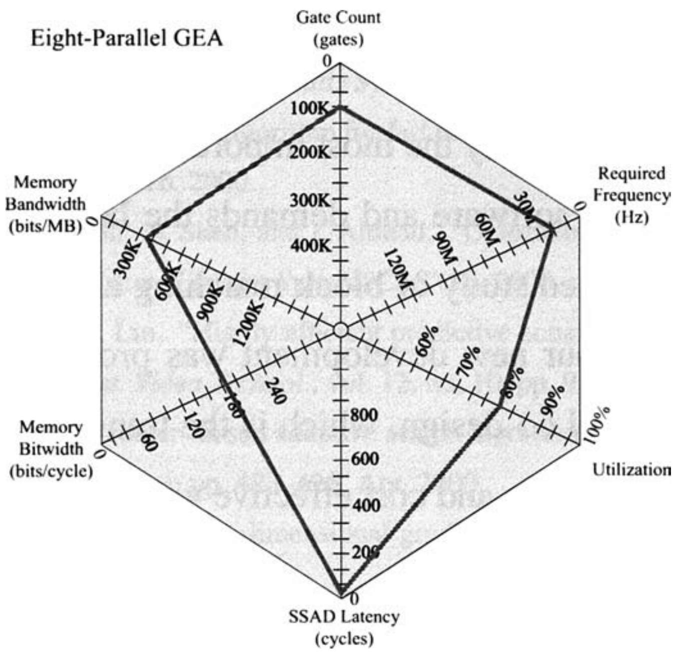


Figure 8. Hexagonal plots of our eight-parallel GEA design for CIF 30 Hz video with search range as H[−4, +63]/V[−2, +31].

in all aspects except a minor loss of video quality. The video quality of an MPEG-4 simple profile encoder adopting our ME accelerator is 0.1–0.2 dB worse than that of adopting FSBMA and Lagrangian mode decision, but is significantly better than that produced by the reference software.

4.6. Summary

We presented a new parallel global elimination algorithm and architecture for fast block matching. By rejecting less possible candidates with simplified distortion estimation, only a few potential candidates are required to be refined with fine distortion estimation. The software complexity of our algorithm is about 8.8% of the full search. To achieve parallel processing, candidate blocks are divided into independent groups so that the coarse distortion estimation of several search positions can be executed simultaneously. Many design techniques, such as systolic flow, 2-D data reuse, reuse of overlapped search area, and resource sharing, are adopted to maximize the overall system performance. Our accelerator is ten times more area-speed efficient than FSBMA architectures and provides better coding performance than the MPEG-4 reference software.

Table 4. Comparison of ME architectures under the specification of D1 30 fps H[−4, +63] V[−2, +31].

Architecture	8-Parallel GEA	512-PE 1-D array	1024-PE 1-D array	2048-PE 1-D array
Gate count	98 K	448 K	896 K	1792 K
Working frequency	70 MHz	166 MHz	83 MHz	42 MHz
SRAM size	13,312 Kbytes	13,312 Kbytes	13,312 Kbytes	13,312 Kbytes
SRAM bandwidth	1,842 Mbytes/sec	6,142 Mbytes/sec	6,059 Mbytes/sec	6,090 Mbytes/sec
Bus bandwidth	71 Mbytes/sec	71 Mbytes/sec	71 Mbytes/sec	71 Mbytes/sec
Functionality	Integer ME, Half ME, AP Mode, Lagrangian MB mode decision	Integer ME (FSBMA)	Integer ME (FSBMA)	Integer ME (FSBMA)

5. Conclusion

Motion estimation engine is usually the most important module in a typical video encoder. It spends the longest run-time in software and demands the largest area/bandwidth in hardware. In this paper, we made a detailed study of block matching algorithms and architectures during the past two decades. Also, our new development was proposed to illustrate the advantages of algorithmic and architectural co-design, which is the trend for VLSI implementation of ME engine to provide high performance and cost effective video coding systems.

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. *Video Codec for Audiovisual Services at p × 64 Kbit/s*, ITU-T Recommendation H.261, Mar. 1993.
5. *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263, Feb. 1998.
6. 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.
7. J. Jain and A. Jain, "Displacement Measurement and its Application in Internal Image Coding," *IEEE Trans. Commun.*, vol. COM-29, no. 12, 1981, pp. 1799–1808.
8. T. Koga, K. linuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, 1981, pp. C9.6.1–C9.6.5.
9. R. Srinivasan and K.R. Rao, "Predictive Coding based on Efficient Motion Estimation," *IEEE Trans. Commun.*, vol. COM-33, no. 8, 1985, pp. 888–896.
10. S. Kappagantula and K.R. Rao, "Motion Compensated Interframe Image Prediction," *IEEE Trans. Commun.*, vol. COM-33, no. 9, 1985, pp. 1011–1015.
11. M. Ghanbari, "The Cross Search Algorithm for Motion Estimation," *IEEE Trans. Commun.*, vol. 38, no. 7, 1990, pp. 950–953.
12. L.G. Chen, W.T. Chen, Y.S. Jehng, and T.D. Chiueh, "An Efficient Parallel Motion Estimation Algorithm for Digital Image Processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 4, 1991, pp. 378–385.
13. 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, 1994, pp. 504–509.
14. 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.
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, 1996, pp. 313–317.
16. 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, 1996, pp. 419–422.
17. J.Y. Tham, S. Ranganath, M. Ranganath, and A.A. Kassim, "A Novel Unrestricted Center-biased Diamond Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, 1998, pp. 369–377.
18. S. Zhu and K.K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation," *IEEE Trans. Image Processing*, vol. 9, no. 2, 2000, pp. 287–290.
19. A.M. Tourapis, O.C. Au, M.L. Liou, G. Shen, and I. Ahmad, "Optimizing the mpeg-4 Encoder - advanced Diamond Zonal search," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'00)*, 2000, pp. 674–677.
20. A.M. Tourapis, O.C. Au, and M.L. Liu, "Highly Efficient Predictive Zonal Algorithms for Fast Block-matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 10, 2002, pp. 934–947.
21. V. Christopoulos and J. Cornelis, "A Center-biased Adaptive Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, 2000, pp. 423–426.
22. O.T.C. Chen, "Motion Estimation using a One-Dimensional Gradient Descent Search," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 4, 2000, pp. 608–616.
23. C.H. Cheung and L.M. Po, "A Novel Cross Diamond Search Algorithm for Fast Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, 2002, pp. 1168–1177.
24. Y.W. Huang, S.Y. Ma, C.F. Shen, and L.G. Chen, "Predictive Line Search: An Efficient Motion Estimation Algorithm for mpeg-4 Encoding Systems on Multimedia Processors," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 13, no. 1, 2003, pp. 111–117.
25. C.W. Lam, L.M. Po, and C.H. Cheung, "A Novel Kite-Cross-diamond Search Algorithm for Fast Video Coding and Videoconferencing Applications," in *Proc. of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP'04)*, 2004, pp. 365–368.
26. S. Zhu and K.K. Ma, "A New Diamond Search Algorithm for Fast Block Matching Motion Estimation," in *Proc. of IEEE Int. Conf. Image Processing (ICIP'97)*, 1997, pp. 292–296.
27. M. Bierling, "Displacement Estimation by Hierarchical Block Matching," in *Proc. of SPIE Visual Commun. Image Processing (VCIP'88)*, 1988, pp. 942–951.
28. A. Zaccarin and B. Liu, "Fast Algorithms for Block Motion Estimation," in *Proc. of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP'92)*, 1992, pp. 449–452.
29. 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, 1993, pp. 148–157.
30. Y. Wang, Y. Wang, and H. Kuroda, "A Globally Adaptive Pixel-decimation Algorithm for Block-motion Estimation," *IEEE*

Trans. Circuits Syst. Video Technol., vol. 10, no. 6, 2000, pp. 1006–1011.

31. H. Gharavi and M. Mills, "Block Matching Motion Estimation Algorithms - New Results," *IEEE Trans. Circuits Syst.*, vol. 37, no. 5, 1990, pp. 649–651.
32. M.J. Chen, L.G. Chen, T.D. Chiueh, and Y.P. Lee, "A New Block-matching Criterion for Motion Estimation and its Implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 3, 1995, pp. 231–236.
33. M.J. Chen, "Predictive Motion Estimation Algorithms for Video Compression," *J. of St. John's St. Mary Institute of Technol.*, vol. 15, 1997, pp. 197–214.
34. J.S. Kim and R.H. Park, "A Fast Feature-based Block Matching Algorithm using Integral Projections," *IEEE J. Select. Areas Commun.*, vol. 10, no. 5, 1992, pp. 968–979.
35. K. Sauer and B. Schwartz, "Efficient Block Motion Estimation using Integral Projections," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, 1996, pp. 513–518.
36. B. Natarajan and V. Bhaskaran, "Low-complexity Block-based Motion Estimation via one-bit Transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 4, 1997, pp. 702–706.
37. J.H. Luo, C.N. Wang, and T. Chiang, "A Novel All-binary Motion Estimation (ABME) with Optimized Hardware Architectures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, 2002, pp. 700–712.
38. Z.L. He, C.Y. Tsui, K.K. Chan, and M.L. Liou, "Low-power VLSI Design for Motion Estimation using Adaptive Pixel truncation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, 2000, pp. 669–678.
39. C.H. Hsieh, P.C. Lu, J.S. Shyn, and E.H. Lu, "Motion Estimation Algorithm using Interblock Correlation," *IEE Electron. Lett.*, vol. 26, no. 5, 1990, pp. 276–277.
40. S. Zafar, Y.Q. Zhang, and J.S. Baras, "Predictive Block Matching Motion Estimation for TV Coding—Part I: Inter-block Prediction," *IEEE Trans. Broadcast.*, vol. 37, no. 3, 1991, pp. 97–101.
41 Y.Q. Zhang and S. Zafar, "Predictive Block-matching Motion Estimation for TV Coding—Part II: Inter-frame Prediction," *IEEE Trans. Broadcast.*, vol. 37, no. 3, 1991, pp. 102–105.
42. M.C. Chen and A.N. Willson J., "A Logarithmic-time Adaptive Block Matching Algorithm in Estimating Large Displacement Motion Vectors," in *Proc. of IEEE Multimedia Commun. Video Coding Symp.*, 1995.
43. J. Chalidabhongse and C.C.J. Kuo, "Fast Motion Vector Estimation using Multiresolution-Spatio-Temporal Correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, 1997, pp. 477–488.
44. D. Tzovaras, M.G. Strintzis, and H. Sahinolou, "Evaluation of Multiresolution Block Matching Techniques for Motion and Disparity Estimation," *Signal Processing: Image Commun.*, vol. 6, 1994, pp. 56–67.
45. J.H. Lee, K.W. Lim, B.C. Song, and J.B. Ra, "A Fast Multiresolution Block Matching Algorithm and its VLSI Architecture for Low Bit-rate Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 12, 2001, pp. 1289–1301.
46. J.H. Lee and N.S. Lee, "Variable Block Size Motion Estimation Algorithm and its Hardware Architecture for H.264," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, 2004, pp. 740–743.
47. W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," *IEEE Trans. Image Processing*, vol. 4, no. 1, 1995, pp. 105–107.
48. X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation," *IEEE Trans. Image Processing*, vol. 9, no. 3, 2000, pp. 501–504.
49. M. Brunig and W. Niehsen, "Fast Full-search Block Matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, 2001, pp. 241–247.
50. C. Zhu, W.S. Qi, and W. Ser, "A New Successive Elimination Algorithm for Fast Block Matching in Motion Estimation," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, 2004, pp. 733–736.
51. C.J. Duanmu, M.O. Ahmad, and M.N.S. Swamy, "8-bit Partial Sum of 16 Luminance Values for Fast Block Motion Estimation," in *Proc. of IEEE Int. Conf. Multimedia Expo (ICME'03)*, 2003, pp. 689–692.
52. Digital Video Coding Group, *ITU-T recommendation H.263 software implementation*, Telenor R'D, 1995.
53. 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, 2000, pp. 417–422.
54. J.N. Kim and T.S. Choi, "A Fast Full-search Motion-estimation Algorithm using Representative Pixels and Adaptive Matching Scan," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, 2000, pp. 1040–1048.
55. K. Lengwehasatit and A. Ortega, "Probabilistic Partial-distance Fast Matching Algorithms for Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, 2001, pp. 139–152.
56. A. Hatabu, T. Miyazaki, and I. Kuroda, "Optimization of Decision-timing for Early Termination of SSDA-based Block Matching," in *Proc. of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP'03)*, 2003, pp. 533–536.
57. Y.S. Chen, Y.P. Huang, and C.S. Fuh, "Fast Block Matching Algorithm based on the Winner-update Strategy," *IEEE Trans. Image Processing*, vol. 10, no. 8, 2001, pp. 1212–1222.
58. I.M. Pao and M.T. Sun, "Modeling Dot Coefficients for Fast Video Encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, 1999, pp. 608–616.
59. D.S. Turaga and T. Chen, "Estimation and Mode Decision for Spatially Correlated Motion Sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 10, 2001, pp. 1098–1107.
60. J. Xin, M.T. Sun, and V. Hsu, "Diversity-based fast block motion estimation," in *Proc. of IEEE Int. Conf. Multimedia Expo (ICME'03)*, 2003, pp. 525/528.
61. P.L. Tsai, S.Y. Huang, C.T. Liu, and J.S. Wang, "Computation-aware Scheme for Software-based Block Motion Estimation," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 13, no. 9, 2003, pp. 901–913.
62. Y.W. Huang, B.Y. Hsieh, T.C. Wang, S.Y. Chien, S.Y. Ma, C.F. Shen, and L.G. Chen, "Analysis and Reduction of Reference Frames for Motion Estimation in MPEG-4 AVC/JVT/H.264," in *Proc. of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP'03)*, 2003, pp. 145–148.
63. H.Y.C. Tourapis and A.M. Tourapis, "Fast Motion Estimation Within the H.264 Codec," in *Proc. of IEEE Int. Conf. Multimedia Expo (ICME'03)*, 2003, pp. 517–520.

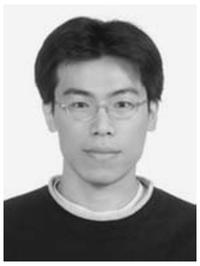
64. Y.K. Tu, J.F. Yang, Y.N. Shen, and M.T. Sun, "Fast Variable Block Motion Estimation using Merging Procedure with an Adaptive Threshold," in *Proc. of IEEE Int. Conf. Multimedia Expo (ICME'03)*, 2003, pp. 789–792.
65. W.I. Choi, B. Jeon, and J. Jeong, "Fast Motion Estimation with Modified Diamond Search for Variable Motion Block Sizes," in *Proc. of IEEE Int. Conf. Image Processing (ICIP'03)*, 2003, pp. 371–374.
66. X. Li, E.Q. Li, and Y.K. Chen, "Fast Multi-frame Motion Estimation Algorithm with Adaptive Search Strategies in H.264," in *Proc. of IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP'04)*, 2004, pp. 369–372.
67. Z. Zhou and M.T. Sun, "Fast Vaiable Block-size Motion Estimation Algorithms based on Merge and Split Procedures for H.264/MPEG-4 AVC," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, 2004, pp. 725–728.
68. M.J. Chen, Y.Y. Chiang, H.J. Li, and M.C. Chi, "Efficient Multi-frame Motion Estimation Algorithms for MPEG-4 AVC/JVT/H.264," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'04)*, 2004, pp. 737–740.
69. C.H. Kuo, M. Shen, and C.C.J. Kuo, "Fast inter-prediction mode decision and motion search for H.264," in *Proc. of IEEE International Conference on Multimedia and Expo*, 2004.
70. P. Yang, Y.W. He, and S.Q. Yang, "An Unisymmetrical-cross Multi-Resolution Motion Search Algorithm for MPEG-4 AVC/H.264 coding," in *Proc. of IEEE International Conference on Multimedia and Expo*, 2004.
71. Y. Su and M.T. Sun, "Fast Multiple Reference Frame Motion Estimation for H.264," in *Proc. of IEEE International Conference on Multimedia and Expo*, 2004.
72. S.Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice Hall, 1988.
73. T. Komarek and P. Pirsch, "Array Architectures for Block Matching Algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, no. 2, 1989, pp. 1301–1308.
74. L.D. Vos and M. Stegherr, "Parameterizable VLSI Architectures for the Full-search Block-matching Algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 2, 1989, pp. 1309–1316.
75. K.M. Yang, M.T. Sun, and L. Wu, "A Family of VLSI Designs for the Motion Compensation Block-matching Algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 2, 1989, pp. 1317–1325.
76. C.H. Hsieh and T.P. Lin, "VLSI Architecture for Block-matching Motion Estimation Algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, 1992, pp. 169–175.
77. Y.S. Jehng, L.G. Chen, and T.D. Chiueh, "An Efficient and Simple VLSI Tree Architecture for Motion Estimation Algorithms," *IEEE Trans. Signal Processing*, vol. 41, no. 2, 1993, pp. 889–900.
78. C.Y. Chen, Y.W. Huang, T.C. Shen, and L.G. Chen, "Analysis and Architecture Design of Variable Block Size Motion Estimation for Video Coding Systems," *IEEE Trans. Circuits and Syst. I*, 2004 (submitted).
79. S.F. Chang, J.H. Hwang, and C.W. Jen, "Scalable Array Architecture Design for Full Search Block Matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 4, 1995, pp. 332–343.
80. H. Yeo and Y.H. Hu, "A Novel Modular Systolic Array Architecture for Full-search Block Matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, 1995, pp. 407–416.
81. Y.K. Lai and L.G. Chen, "A Data-interlacing Architecture with two-Dimensional Data-reuse for Full-search Block-matching Algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 2, 1998, pp. 124–127.
82. Y.H. Yeh and C.Y. Lee, "Cost-effective VLSI Architectures and Buffer Size Optimization for Full-search Block Matching Algorithms," *IEEE Trans. VLSI Syst.*, vol. 7, no. 3, 1999, pp. 345–358.
83. 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, 2002, pp. 61–72.
84. Mei-Yun Hsu, *Scalable module-based architecture for MPEG-4 BMA motion estimation*, Master Thesis, National Taiwan Univ., 2000.
85. C.H. Chou and Y.C. Chen, "A VLSI Architecture for Real-time and Flexible Image Template Matching," *IEEE Trans. Circuits Syst.*, vol. 36, no. 2, 1989, pp. 1336–1342.
86. 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, 1998, pp. 393–398.
87. A. Hanami, S. Scotzniovsky, K. Ishihara, T. Matsumura, S. I. Takeuchi, H. Ohkuma, K. Nishigaki, H. Suzuki, M. Kazayama, T. Yoshida, and K. Tsuchihashi, "A 165-GOPS Motion Estimation Processor with Adaptive Dual-array Architecture for High Quality Video-encoding Applications," in *Proc. of IEEE Custom Integrated Circuits Conf. (CICC'98)*, 1998, pp. 169–172.
88. J.F. Shen, T.C. Wang, 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, 2001, pp. 890–897.
89. N. Roma and L. Sousa, "Efficient and Configurable Full-search Block-matching Processors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1160/1167, Dec. 2002.
90. Y.W. Huang, T.C. Wang, B.Y. Hsieh, and L.G. Chen, "Hardware Architecture Design for Variable Block Size Motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'03)*, 2003, pp. 796–799.
91. H.M. Jong, L.G. Chen, and T.D. Chiueh, "Parallel Architectures for 3-step Hierarchical Search Block-matching Algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, 1994, pp. 407–416.
92. S. Dutta and W. Wolf, "A Flexible Parallel Architecture Adopted to Block-matching Motion Estimation Algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, 1996, pp. 74–86.
93. H.D. Lin, A. Anesko, and B. Petryna, "A 14-GOPS Programmable Motion Estimator for H.26 × Video Coding," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, 1996, pp. 1742–1750.
94. S.C. Cheng and H.M. Hang, "A Comparison of Block-matching Algorithms Mapped to Systolic-array Implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, 1997, pp. 741–757.
95. M. Mizuno, Y. Ooi, N. Hayashi, J. Goto, M. Hozumi, K. Furuta, A. Shibayama, Y. Nakazawa, O. Ohnishi, S. Y. Zhu, Y. Yokoyama, Y. Katayama, H. Takano, N. Miki, and Y. Senda,

"A 1.5-W Single-chip MPEG-2 MP@ML Video Encoder with Low Power Motion Estimation and Clocking," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, 1997, pp. 1807–1816.

96. M. Takahashi, M. Hamada, T. Nishikawa, H. Arakida, T. Fujita, F. Hatori, S. Mita, K. Suzuki, A. Chiba, T. Terazawa, F. Sano, Y. Watanabe, K. Usami, M. Igarashi, T. Ishikawa, M. Kanazawa, T. Kuroda, and T. Furuyama, "A 60-mW MPEG-4 Video Codec using Clustered Voltage Scaling with Variable Supply-voltage Scheme," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, 1998, pp. 1772–1780.
97. V.G. Moshnyaga, "A New Computationally Adaptive Formulation of Block-matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 1, 2001, pp. 118–124.
98. S.C. Hsia, "VLSI Implementation for Low-complexity Full-search Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, 2002, pp. 613–619.
99. S. Kawahito, D. Handoko, Y. Tadokoro, and A. Matsuzawa, "Low Power Motion Vector Estimation using Iterative Search Block-matching Methods and a High-speed Non-destructive CMOS Sensor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, 2002, pp. 1084–1092.
100. C.D. Vleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and Architectural Co-design of a Motion-estimation Engine for Low-power Video Devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, 2002, pp. 1093–1105.
101. W.M. Chao, C.W. Hsu, Y.C. Chang, and L.G. Chen, "A Novel Motion Estimator Supporting Diamond Search and Fast full Search," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'02)*, 2002, pp. 492–495.
102. W.M. Chao, T.C. Chen, Y.C. Chang, C.W. Hsu, and L.G. Chen, "Computationally Controllable Integer, Half, and Quarter-pel Motion Estimator for MPEG-4 Advanced Simple Profile," in *Proc. of IEEE Int. Symp. Circuits Syst. (ISCAS'03)*, 2003, pp. 788–791.
103. Y.W. Huang, S.Y. Chien, B.Y. Hsieh, and L.G. Chen, "Global Elimination Algorithm and Architecture Design for Fast Block Matching Motion Estimation," *IEEE Trans. Circuits and Syst. Video Technol.*, vol. 14, no. 6, 2004, pp. 898–907.
104. A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G.J. Sullivan, "Performance Comparison of Video Coding Standards using Lagragian Coder Control," in *Proc. of IEEE International Conference on Image Processing*, 2002.

Yu-Wen Huang was born in Kaohsiung, Taiwan, in 1978. He received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from National Taiwan University, Taipei, in June 2000 and December 2004, respectively. He joined MediaTek, Inc., Hsinchu, Taiwan, in 2004, where he develops integrated circuits related to video coding systems. His research interests include video segmentation, moving object detection and tracking, intelligent video coding technology, motion estimation, face detection and recognition, H.264/AVC video coding, and associated VLSI architectures.
yuwen@video.ee.ntu.edu.tw

Ching-Yeh Chen was born in Taipei, Taiwan, in 1980. He received the B.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 2002. He currently is pursuing the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University. His research interests include intelligent video signal processing, global/local motion estimation, scalable video coding, and associated VLSI architectures.
cychen@video.ee.ntu.edu.tw



Chen-Han Tsai received the B.S. degree in electrical engineering from National Taiwan University in 2002. Now he is working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University. His major research interests include face detection and recognition, motion estimation, H.264/AVC video coding, digital TV systems, and related VLSI architectures.
phenom@video.ee.ntu.edu.tw

Chun-Fu Shen received the B.S. and M.S. degrees in electrical engineering from National Taiwan University in 1996 and 1998, respectively. After two years of military service, he joined VIVOTEK, Inc., Taipei County, Taiwan, in 2000. He developed many video coding systems and IP camera products on DSP platforms and ASICs.

His major research interests include JPEG, H.263, MPEG-4, and H.264/AVC coding systems, network camera SOC, and embedded systems.
sor@vivotek.com

Liang-Gee Chen was born in Yun-Lin, Taiwan, in 1956. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, in 1979, 1981, and 1986, respectively. He was an instructor (1981–1986), and an associate professor (1986–1988) in the Department of Electrical Engineering, National Cheng Kung University. In the military service during 1987 and 1988, he was an associate professor in the Institute of Resource Management, Defense Management College. From 1988, he joined the Department of Electrical Engineering, National Taiwan University. During 1993 to 1994 he was a visiting consultant of DSP Research Department, AT&T Bell Lab, Murray Hill. In 1997, he was the visiting scholar of the Department of Electrical Engineering, University, of Washington, Seattle. Currently, he is a professor of National Taiwan University. From 2004, he is also the executive vice president and the general director of Electronics Research and Service Organization (ERSO) in the Industrial Technology Research Institute (ITRI). His current research interests are DSP architecture design, video processor design, and video coding systems.
Dr. Chen is a Fellow of IEEE. He is also a member of the honor society Phi Tau Phi. He was the general chairman of the 7th VLSI Design CAD Symposium. He was also the general chairman of the 1999 IEEE Workshop on Signal Processing Systems: Design and Implementation. He has served as the associate editor of *IEEE Transactions on Circuits and Systems for Video Technology* since 1996, the associate editor of *IEEE Transactions on VLSI Systems* since 1999, the associate editor of *Journal of Circuits, Systems, and Signal Processing* since 1999, and the guest editor of *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology* since 2001. Now he is also the associate editor of *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* and the associate editor of *Proceedings of the IEEE*.
Dr. Chen received the Best Paper Awards from ROC Computer Society in 1990 and 1994. From 1991 to 2005, he received Long-Term (Acer) Paper Awards annually. In 1992, he received the Best Paper Award of the 1992 Asia-Pacific Conference on Circuits and Systems in VLSI design track. In 1993, he received the Annual Paper Award of Chinese Engineer Society. In 1996, he received the Outstanding Research Award from National Science Council (NSC) and the Dragon Excellence Award from Acer. He was elected as the IEEE Circuits and Systems Distinguished Lecturer from 2001–2002.
lgchen@cc.ee.ntu.edu.tw