Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 890410, 8 pages
doi:10.1155/2008/890410

Research Article

Fully Pipelined Parallel Architecture for Candidate Block and Pixel-Subsampling-Based Motion Estimation

Reeba Korah and J.Raja Paul Perinbam

School of Electronics and Communication Engineering (ECE), Anna University, Chennai-600 025, Tamil Nadu, India

Correspondence should be addressed to Reeba Korah, reeba26in@yahoo.co.in

Received 7 May 2007; Revised 26 September 2007; Accepted 2 January 2008

Recommended by Mohab Anis

This paper presents a low power and high speed architecture for motion estimation with Candidate Block and Pixel Subsampling (CBPS) Algorithm. Coarse-to-fine search approach is employed to find the motion vector so that the local minima problem is totally eliminated. Pixel subsampling is performed in the selected candidate blocks which significantly reduces computational cost with low quality degradation. The architecture developed is a fully pipelined parallel design with 9 processing elements. Two different methods are deployed to reduce the power consumption, parallel and pipelined implementation and parallel accessing to memory. For processing 30 CIF frames per second our architecture requires a clock frequency of 4.5 MHz.

1. INTRODUCTION

The coding of video sequences has been the focus of a great deal of researches in recent years. Video phone, video conferencing, CD-ROM archiving, and HDTV are some of the present-day applications. Data compression techniques must be used before transmission due to a large amount of image data to be transmitted, whatever be the application.Video compression can be achieved by reducing spatial and temporal redundancies within video streams. Motion estimation and compensation (MEC) is the key technique for the exploitation of temporal redundancy. Since MEC operations take up to 80% of the computational burden of a complete video compression system, it is the most important component in real-time video applications. Many VLSI implementable algorithms aim at either high-performance or low-power design. Most of the architectures targeting the abovementioned applications do not seem to be suitable for mobile and low-power applications. Due to the rapid advances in VLSI technology, the attributes of parallelism, pipeline ability, concurrency, modularity, and regularity have become a new set of criteria in designing the hardware for digital video processing.

Systolic arrays are good candidates for such design. High-speed systolic array architectures able to process a large number of calculations needed for FSBMA have been widely proposed. Artieri and Jutand [1] presented a systolic array architecture where a processor is associated with each possible match. A 2D SIMD systolic architecture was proposed by Wu and Yeh [2] for an FSBMA-based motion estimator with a major advantage of reduced pin count of 68 pins. Yang et al. [3] proposed a 1D semisystolic architecture to perform FSBMA with the help of parallel processing. They further optimized the processing elements using pipeline architecture in order to reduce the cycle time. Roma and Sousa [4] implemented a configurable block matching processor based on an improved 2D multiple array architecture with both pipelining and parallel processing which resulted in minimum latency, maximum throughput, and full utilization of hardware.

A motion estimation chip for block-based MPEG-4 video applications, using predictive diamond search, was presented by Abbas et al. [5] with reconfigurable search window and pixel block size. A parallel pipelined architecture for FSBMA was proposed by Sayed and Badawy [6]. Vos and Schobinger [7] implemented a programmable block matching processor using a quadratic systolic array architecture with meander-like data flow. A data interlacing VLSI architecture with 2D data reuse to implement FSBMA was proposed by Lai and Chen [8], which efficiently reuses data to decrease external

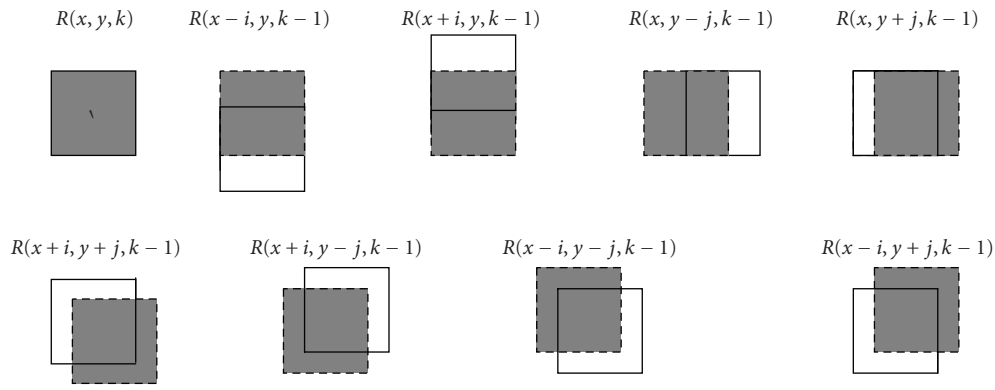


Figure 1: Overlapped Candidate blocks in different orientations.

memory accesses and saves pin counts. Xu et al. [9] proposed a 1D tree-based architecture which featured high-data utilization by using parallel pipeliningand the low-clock rate by introducing the dual register/buffer technique which reduces idle clock cycles. Instead of a systolic array, Wang et al. [10] used a 2D mapping unit and a binary adder tree to compute the block matching metric in one cycle.

The CBPS algorithm is a proper blend of FSBMA and FBMME approaches. With this, the subjective quality is at par with that of FSBMA whereas computational complexity and hence area consumption and power consumption are least compared to any fast BMME reported so far. In our earlier work [11], we implemented a single processing element architecture which requires a clock frequency of 65 MHz to process 352×288 resolution CIF frames at the rate of 30 f/s. Here, we present an efficient fully pipelined parallel architecture for CBPS algorithm.

The rest of the paper is organized as follows. Section 2 presents the mathematical modeling of CBPS-based motion estimation. CBPS algorithm is dealt with in Section 3 in detail. Section 4 deals with the proposed architecture design. Section 5 presents the architecture prototyping and evaluation of results.The performance of the architecture is analyzed in terms of power dissipation, gate count, and clock frequency required. Finally, the conclusions are drawn in Section 6.

2. MATHEMATICAL MODELING OF CBPS-BASED MOTION ESTIMATION

Even though many fast motion vector estimation techniques have been proposed as reviewed before, the spatial and temporal correlations of motion vectors have not yet been fully exploited in reducing the search time while maintaining a reasonable rate-distortion tradeoff. The use of an AR model to characterize spatiotemporal correlations of the motion field could provide an elegant theoretical result. However, its derivation requires a certain amount of computational complexity and its practical value decreases. The goal of this research is to develop a fast motion vector estimation algorithm, which exploits the spatiotemporal correlations of motion vectors in a computationally simple way and yet works effectively in the sense of producing small residual errors.

The following framework is adopted in our discussion. Each image frame is divided into nonoverlapping square macro blocks of 16×16 pixels as specified by H.264/AVC. Let $R(i, j, k)$ represent a block of the kth frame, where i and j are block indices along the row and column directions, respectively. For example, an image of size 288×352 pixels has block indices $i = 0, 1, 2, \ldots, 17$ and $j = 0, 1, 2, \ldots, 21$. $R = (0, j, k)$ and $R = (i, 0, k)$ represent the blocks in the first row and first column, respectively. Motion vector for each block between kth frame and one or more consecutive frames with a certain fast motion vector estimation algorithm needs to be determined.

A simple way to incorporate the temporal information with the spatial information is to include the motion vector of the block at the same location from the previous frame $R(x, y, k-1)$. Furthermore, information about the motion vector block can be obtained by searching the blocks surrounding $R(x, y, k-1)$. Thus, for a block $R(x, y, k)$ its temporally correlated blocks in the previous frame as shown in Figure 1 will be $R(x, y, k-1)$, $R(x-i, y, k-1)$, $R(x+i, y, k-1)$, $R(x, y+j, k-1)$, $R(x, y-j, k-1)$, $R(x-i, y-j, k-1)$, $R(x-i, y+j, k-1)$, $R(x+i, y-j, k-1)$, and $R(x+i, y+j, k-1)$, where i, j are pixel indices: $i = 0, 1, \ldots, 15$ and $j = 0, 1, 2, \ldots, 15$.

Typical patterns with $i = 0/1$ and/or $j = 0/1$ are shown in Figure 1.

For $i = 0$ to 15, $j = 0$ to 15 and for a search range $-p$ to $p = -8$ to 8; $(2p+1)^2$ overlapped blocks can be identified with a one pixel distance horizontally or/and vertically between adjacent overlapped blocks in the $(k-1)$th frame. Out of these $(2p+1)^2$-overlapped blocks, $(p^2 + 3p/2 + 1)$ blocks are chosen for coarse search. The selection is done in such a way that the mean horizontal or vertical pixel distance between the selected overlapped blocks is 1 as shown in Figure 2.

Pixel distance of 1 refers to the first-order neighborhood sites as per the Markov model. Markovianity emphasizes the spatial interactions of adjacent sites, and hence this is employed as a basis of fine search in this work. Fine search is performed with those unselected overlapped blocks which are

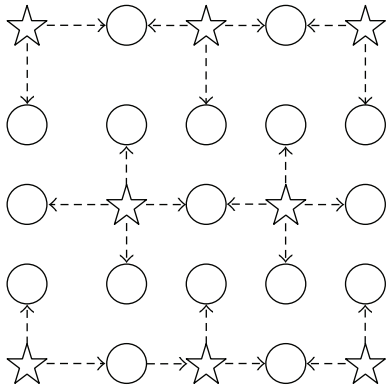


Figure 2: Distribution lattice of overlapped search points.

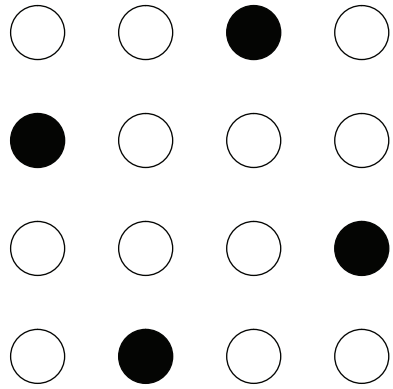


Figure 3: 4-Queen pixel lattice.

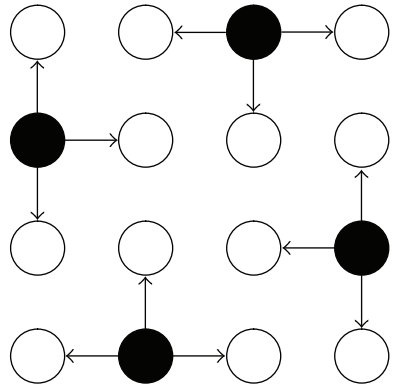


Figure 4: Spatial homogeneity in the subsampling lattice.

first-order neighbor sites for the block with minimum block difference (MBD) among all the $(p^2 + 3p/2 + 1)$ overlapped blocks. Thus, fine search is performed always on 8 blocks, in spite of changes in the search range or macroblock size. MBD is calculated with the most popular sum of absolute difference (SAD) criteria. Pixel decimation is used to reduce the computation for measuring the distortion for each block during the search. High activity in the luminance signals such as edges and texture mainly contributes to the matching criterion. The most representative sampling lattice is selected based on how much the texture and edge information are retained with minimal number of pixels. The sampling lattice is analyzed with spatial homogeneity and directional coverage. The spatial homogeneity is based on the presence of significant intraframe correlations within an image. The intraframe autocorrelation function for a less detailed image is higher when the horizontal and vertical spacing (in pixels) are close to 1. The same holds good for highly detailed images. Correlation coefficient between neighboring pixels increases as their spacing decreases. The spatial homogeneity is measured by the average and variance of spatial distances from each skipped pixel to the nearest selected pixel:

$$\mu_d = \frac{1}{N^2 - K} \sum_{x=1}^{N} \sum_{y=1}^{N} ||(x, y) - S(x, y)||,$$
$$\sigma_d^2 = \frac{1}{N^2 - K} \sum_{x=1}^{N} \sum_{y=1}^{N} (||(x, y) - S(x, y)|| - \mu_d)^2, \quad (1)$$

where N is the dimension of the block and $S(x, y)$ indicates the coordinates of the selected pixel nearest to the pixel at the position (x, y). K is the number of the selected pixels. Smaller μ_d and σ_d^2 indicate a more spatially homogeneous sampling lattice.

An edge is defined as a line passing through the sampling grids in any of 0°, 45°, 90°, and 135° directions. The directional coverage is measured as the percentage of edges that at least one of the selected pixels exists on an edge.

To fully represent the spatial information of an $N \times N$ block, it is required that at least one pixel should be selected for each row, column, and diagonal. To satisfy such a condition, the solution is identical to the problem of placing N-queens on a chess board. This is referred to as N-queen pattern. For an $N \times N$ block, every pixel of the N-queen pattern occupies a dominant position, which is located at the center. All the other pixels located on the four lines in the vertical, horizontal, and diagonal directions are removed from the list of the selected pixels. With such elimination process, there is exactly one pixel selected for each row, column, and diagonal (not necessarily the main) of the block.

To reduce the computational burden, SAD is performed on 4-queen lattices within 16×16 blocks. This is a 4 : 1 pixel subsampling technique as shown in Figure 3.

The above pattern is chosen based on the spatial homogeneity as shown in Figure 4 and directional coverage of pixels as depicted in Figure 5. Accordingly, for a block of size 16×16, only the values at 64 pixels are used to compute the SAD. Similar technique presented by Liu and Zaccarin [12] shows that a reasonably good motion vector estimate can be obtained by using a pixel subsampling technique. The 4-queen method gives a computational reduction by a factor of 4 in comparison with a straightforward implementation of the SAD computation. In order to further reduce the computational complexity, the error criterion is modified as reduced bit sum of absolute difference (RBSAD). As the operation of the BMA is block-by-block comparison, and the comparison is performed pixel by pixel, the basic operation of the BMA is pixel comparison. Since the pixel comparison

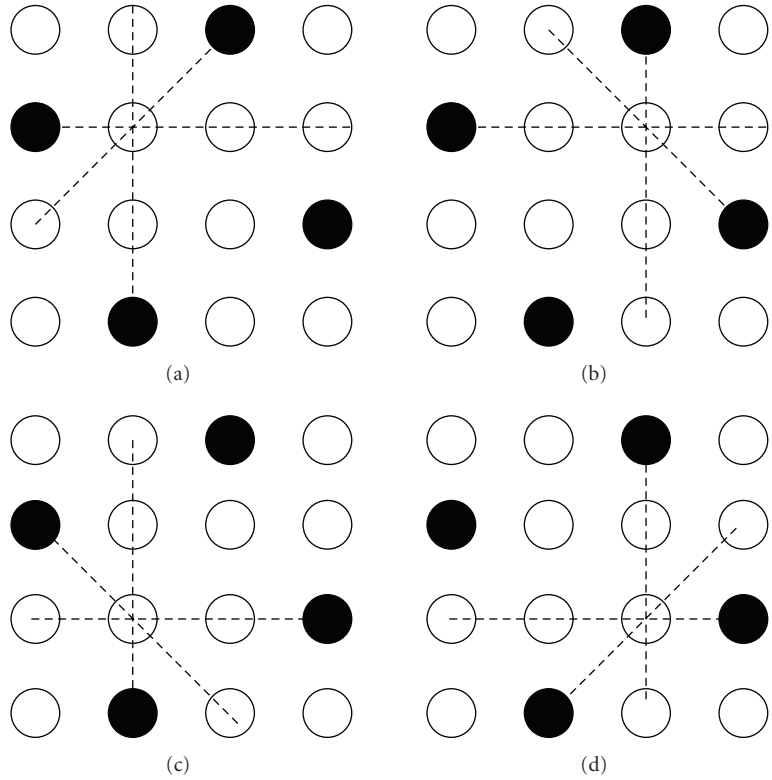


Figure 5: Directional coverage in the subsampling lattice.

is a bitwise operation, if the number of bits is reduced then the corresponding hardware can be reduced. Taking hardware realization into consideration, RBSSAD is proposed.

For any pixel value A, let $Ap : q$, be $p - q + 1$ bits $Ap, \ldots, Aq$, in the binary representation of $A = A_{n-1} \times 2^{n-1}, \ldots, A_2 \times 2^2 + A_1 \times 2 + A_0$, where $n \geq p \geq q \geq 0$. (Let n be the number of bits in a pixel.)

For example, let $A = (10110101)_2$. Then $A_{7:5} = (10110)_2$.

The proposed RBSAD criterion can be described as follows:

$$\text{RBSAD}\,(u, v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| S(i+u, j+v)_{n-1:n-k} - R(i, j)_{n-1:n-k} \right|, \tag{2}$$

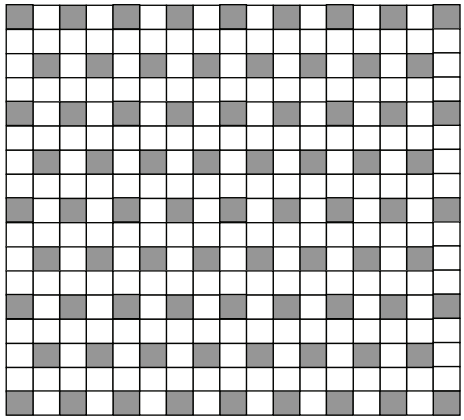
where k is a given factor which should be $n \geq k \geq 0$, $R(i, j)$ is the reference block of size $N \times N$ at coordinate (i, j), $S(i+u, j+v)$ is the candidate block within a search area in the previous frame, and (u, v) represents the candidate motion vector. The motion vector is determined by the least RBSAD (u, v) for all possible displacements (u, v) within a search area. Since RBSAD uses only upper k bits of pixels, if $k = n$, RBSAD is equivalent to SAD.

A normally encountered problem in fast search methods is the chances of misinterpreting local minima as global minima. This is mainly due to the reduced number of search points spread in a particular small region out of the entire search area. So in the proposed technique, the search points are spread over the entire search area. This goes in line with the probabilistic global search procedure of "Multistart." The inherent drawback of MultiStart is the possibility of determination of same minimum several times. To avoid this to an extent, the search points are chosen based on Markov model.

Suppose that the maximum motion in the vertical and horizontal directions is $\pm p$, there are $(2p + 1)^2$candidates in total to be checked if the full search method is used, each corresponding to a point (called candidate block) in the search window. The SAD values resulted from these points form an error surface

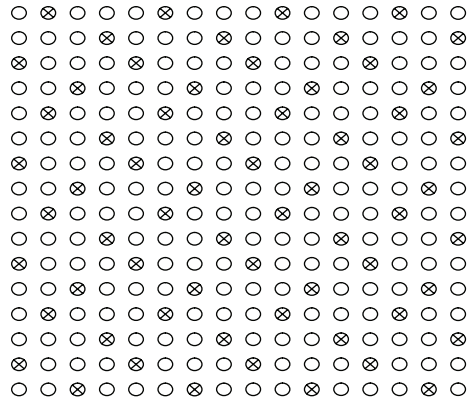
$$E(u, v) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |\mathrm{R}(i+u, j+v, k-1) - \mathrm{R}(i, j, k)|; \quad -p \leq u, v \leq p. \tag{3}$$

The complexity of this error surface has a significant impact on the performance of the algorithm. Almost all conventional fast algorithms have explicitly or implicitly made the assumption that the SAD increases monotonically as the checking point moves away from the global minimum or the error surface is unimodal over the search window. Unfortunately, this assumption is usually not true due to many reasons such as the luminance change between frames. As a consequence, the search would easily be trapped at a local minimum. Despite that the error surface defined above exhibits uncertainties in large spatial scale, we can reasonably assume that it is monotonic in a small neighborhood around the global minimum. In the existence of local minima, one



Selected

Figure 6: Selection of candidate blocks.



Selected pixel

Figure 7: New 4-queen pattern.

simple but perhaps the most efficient and reliable strategy is to put (at least) one checking point as close as possible to the global minimum point (representing the true motion vector). This is equivalent to reducing the distance from the true motion vector to the closest checking point as much as possible. If this distance is small enough, it will be very likely to find the global minimum through a local search.

3. CBPS ALGORITHM

Candidate block and pixel subsampling algorithm is based on a coarse-to-fine search approach. It is a proper blend of full search block matching algorithm and fast search block matching approach.

Here, we have constructed a new pattern of candidate blocks, as shown in Figure 6, to characterize spatial information in all directions. Initially, with coarse search, $(p^2 + 3p/2 + 1)$ blocks are searched and during fine search 8 more blocks are searched. This arrangement reduces the number of candidate blocks to be searched to $(p^2 + 3p/2 + 9)$ against $(2p + 1)^2$ used by FSBMA. Since the candidate blocks are uniformly chosen throughout the search area, during the coarse search, we get the direction of candidate block with global minima. During fine search, error is calculated for 8 blocks surrounding the chosen candidate block. Here, the block with minimum error will be the global minimum point. Thus, in this method, absolutely there is no chance of misinterpreting local minima as global minima. Pixel subsampling technique proposed here consists of four alternating subsampling patterns selected for each step as shown in Figure 7, so that all the pixels in the current block are visited. The total computational complexity is found to be reduced to 7.35% compared to FSBMA. The loss in PSNR is very negligible and at the worst case comes to an average of less than 0.23 dB in both low-motion and medium-motion video sequences. Thus, the subjective quality is at par with that of FSBMA whereas computational complexity, area consumption, and power consumption are the least compared to any fast BMME reported so far. The error criterion used is the subsampled sum of absolute difference (SSAD). The computational complexity could be further reduced by truncating the least significant two bits. This is termed as reduced bit subsampled sum of absolute difference (RBSSAD6). The total computational complexity is found to be reduced to 5.9 % compared to FSBMA. The loss in PSNR is very negligible and at the worst case comes to an average of less than 0.23 dB in both low-motion and medium-motion video sequences. Thus, the subjective quality is at par with that of FSBMA whereas computational complexity, area consumption, and power consumption are competitive compared to any fast BMME reported so far.

CBPS algorithm can be expressed with the following equations:

$$\begin{aligned}
&\text{RBSSAD}\,(m,n)\\
&= \text{RBSSAD}\,(m,n) + \text{SM}(i,j)^{*} R(x^{*}N + i, y^{*}N + j)_{n-1:\,n-k}\\
&\quad \times S(x^{*}N + i + u,\ y^{*}N + j + v)_{n-1:\,n-k},\\
&\text{MV} = \{(u,v \mid \text{RBSSAD}\,(\text{u,v}) \le \text{RBSSAD}\,(\text{m,n})\},
\end{aligned} \tag{4}$$

where R = current frame macroblock pixel, S = previous or candidate macroblock pixel, and SM = subsampling matrix for macroblock $= \begin{bmatrix} 0&0&1&0\\ 1&0&0&0\\ 0&0&0&1\\ 0&1&0&0 \end{bmatrix}$.

4. PROPOSED FULLY PIPELINED PARALLEL ARCHITECTURE

In digital CMOS systems, there are three major sources of power consumption:

$$P_{\text{total}} = P_t(C_L V_{\text{dd}}^2 f_{\text{clk}}) + I_{\text{sc}} V_{\text{dd}} + I_{\text{leakage}} \cdot V_{\text{dd}}. \tag{5}$$

The first term represents the switching power component, where P_t is the transition probability, C_L is the loading capacitance, V_{dd} is the supply voltage, and f_{clk} is the clock

Table 1: Comparison of performance parameters of propsed architecture with other architectures.

	[1]	[2]	[3]	[4]	[5]	[6]	[11]	Proposed
Process (μm)	1.2	0.8	0.8	0.25	0.18	0.18	0.18	0.18
Supply voltage (V)	—	—	—	3.3	1.8	1.6	1.6	1.6
Core size (mm^2)	64	23	—	16.07	2.1	0.795	0.287	0.826
Clock rate (MHz)	36	23	150	36.5	67	100	65	4.5
Power (mW)	1200	—	—	—	452	312.07	234.4	147.6
Number of transistors required	2,49,000	85,736	3,20,000	—	61,603	—	39,488	64,736

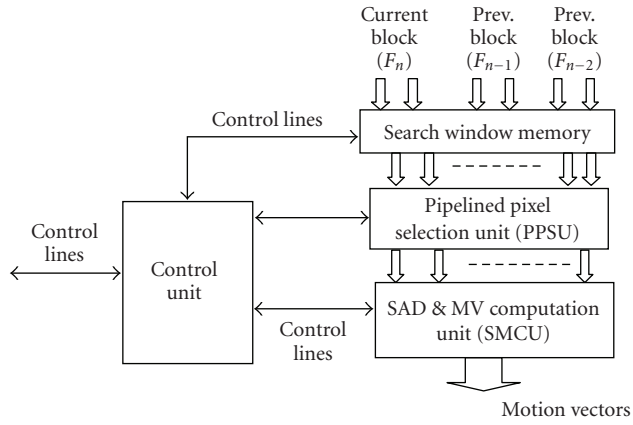


Figure 8: Fully pipelined parallel architechture for CBPSA.

frequency. The second term ($I_{sc} \cdot V_{dd}$) is the short circuit power component, and the third term ($I_{leakage} \cdot V_{dd}$) is due to the leakage current $I_{leakage}$. Switching power is the dominant source of power dissipation in digital CMOS systems. Reducing the supply voltage is the most effective way to reduce power consumption as shown in (5). Quadratic improvement in power consumption can be achieved in this way, although this happens at the cost of increasing the delay. One of the techniques to increase the throughput is a parallel implementation, which in turn reduces the supply voltage. Thus reduction in power consumption can be achieved at the cost of increasing the area instead of increasing the delay.

The proposed architecture consists of two main parts, namely, the pipelined pixel selection unit (PPSU) and the SAD&MV computation unit (SMCU) with pipelining registers between them and a control unit to control their operation as shown in Figure 8.

A search window of 32×32 pixels and a macroblock of 16×16 pixels are used for prototyping. For candidate block and pixel subsampling algorithm (CBPSA), we have 289 (17×17) candidate blocks out of which only 77 blocks are chosen for coarse search and 8 candidate blocks for fine search. Seventy seven blocks are arranged in groups of 9 blocks and 8 blocks in alternate odd rows referred to as *A rows* and *B rows*. Block diagram of the parallel architecture of the pipelined pixel selection unit (PPSU) is shown in Figure 9. It consists of a memory for the search window data, a queue for the reference macroblock pixels, a pipelined structure for the candidate block pixels, and 9 processing elements, one for each selected column of candidate blocks. The search window pixels are selected as follows. A pipeline structure made of 17 multiplexers, in the first layer, is used in order to access 17 columns of pixels. Eight multiplexers and one buffer are used to select between *A rows* block pixels and *B rows* block pixels. During fine search, the outputs ($M_0 - M_{16}$) of the first layer multiplexers are mapped into 3 outputs ($O_0 - O_2$). The reference block is stored in an internal memory of 64 bytes and the reference block pixels are broadcasted to all the processing elements at the same time.

Internal structure of the search window memory consists of a horizontal address decoder and a vertical address decoder as shown in Figure 10. Seventeen memory columns are multiplexed and connected to 9 processing elements.

Processing element consists of one subtractor and one accumulator. Processing element takes two inputs: one coming from the reference block queue and the other coming from one of the 17 columns consisting the candidate block in the search window memory. The accumulator adds or subtracts the subtractor output according to its sign bit to accumulate the absolute value needed in the SSAD computation as in (2). SSAD computed by PE is compared to the existing SSAD and the result is given to a final coarse comparator which is a parallel structure. Nine parallel PEs comparator and the final coarse comparator form the SSAD and motion vector computation unit (SMCU) as shown in Figure 11.

Structure of final coarse comparator, as shown in Figure 12, consists of comparators arranged in a parallel tree architecture. The structure is simple and regular, so that coarse and final motion vectors are obtained very easily.

5. ARCHITECTURE PROTOTYPING AND RESULTS

The proposed architecture has been prototyped with Xilinx Virtex-II Pro XUPV2 as the target device, simulated using Xilinx 8.1 *i* and synthesized using *SynplifyPro 8v0* for a typical real-time video application with $p = 16$, $f = 30$, $N_h = 352$, and $N_v = 288$ (CIF). Table 1 shows the comparison between the proposed architecture and six different motion estimation architectures. Our architecture is 60.66% more area efficient and 67.35% more power efficient compared to [5]. The major achievement is in the clock frequency. Required frame rate of 30 f/s is achieved with 4.5 MHz clock which allows the usage of a reduced power supply which in turn promises less power consumption. With this hardware, time required to encode a frame is 3.7 milliseconds against 33.3 milliseconds as per the standard. This shows that the architecture can be

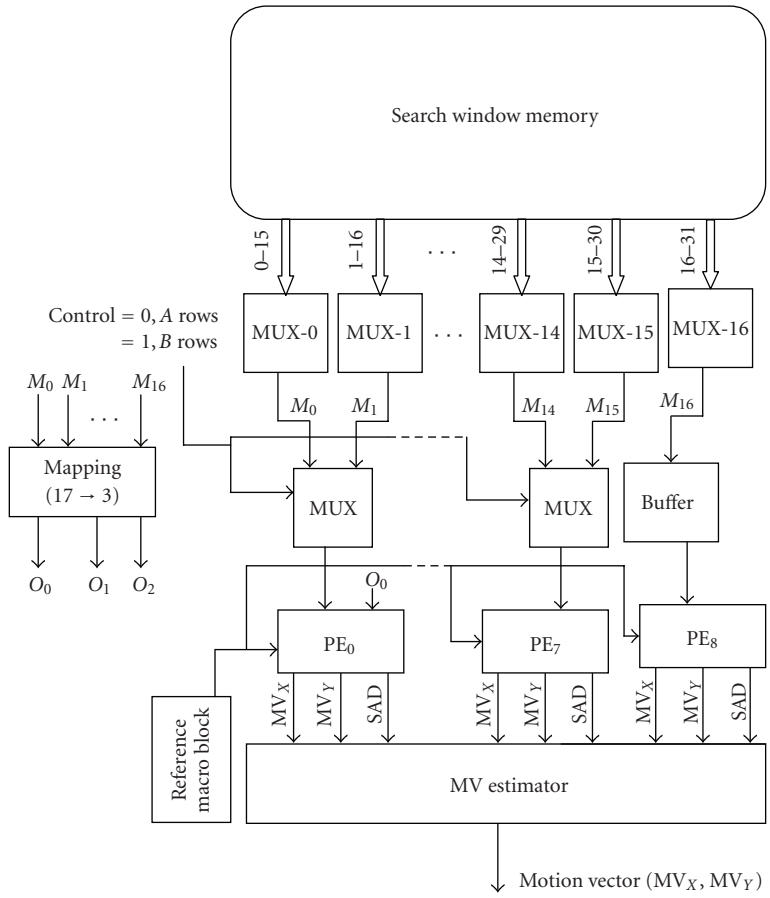


Figure 9: Pipelined pixel selection unit(PPSU).

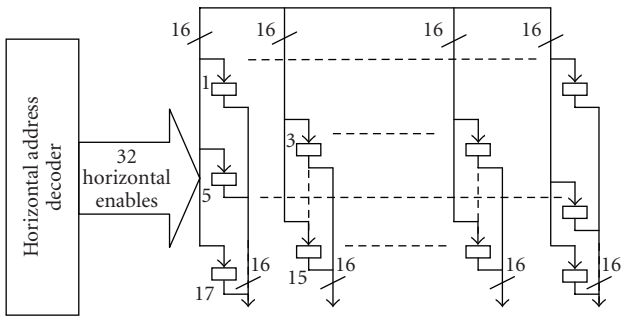


Figure 10: Internal structure of search window memory.

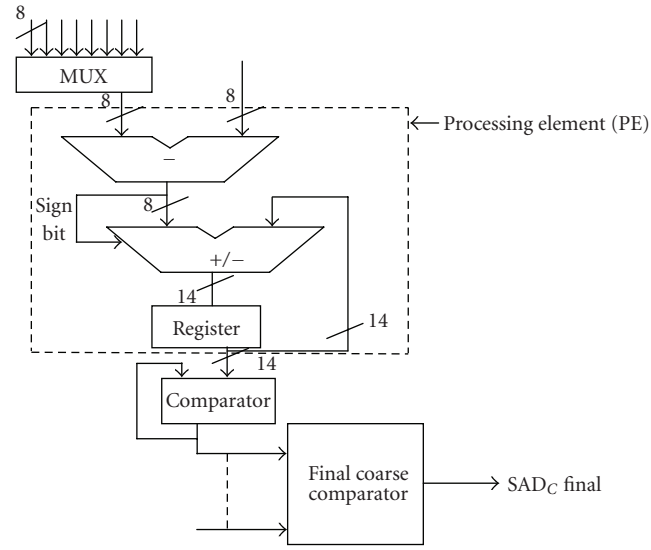


Figure 11: Block diagram of the SAD and MV Computation Unit (SMCU).

very much useful in high throughput applications. However, gate count is 5.08 % more than that in [5] and the core size is 3.9 % more than that in [6]. Results of frequency scaling and voltage scaling are shown in Tables 2 and 3, respectively.

6. CONCLUSION

This paper presented a fully pipelined parallel implementation of novel coarse-to-fine search CBPS algorithm for motion estimation which successfully reduces the computational complexity to a large extend without affecting the subjective quality. This architecture gives very good speed performance and very low-power dissipation compared to

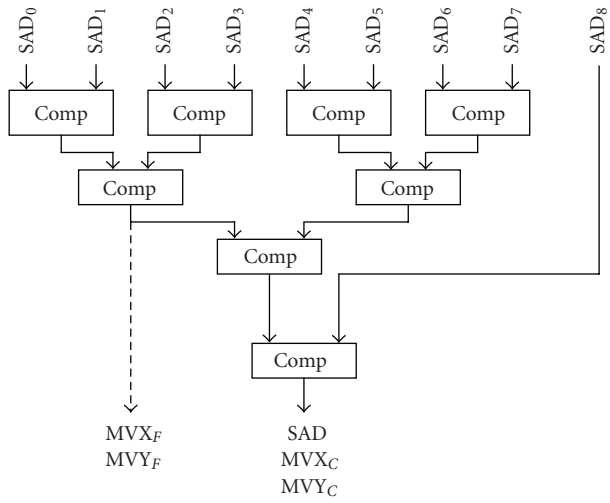


Figure 12: Block diagram of the final coarse comparator.

Table 2: Performance of proposed architecture with frequency scaling.

Video application	Frame resolution	Frequency (MHz)
CIF	352 x 288	4.5
QCIF	176 x 144	1.1
NTSC	480 x 483	20.74
PAL	576 x 576	24.88
CCIR	720 x 480	25.92
HDTV	1280 x 720	165.9
HDTV2	1920 x 1080	188

Table 3: Effect of voltage scaling for the proposed architecture (for 0.18 μ technology, VDD =1.8 V).

Architecture	[5]	Proposed
Power (mW)	452	168.7

different architectures for block motion estimation. We have tested its performance with different CIF sequences for videophone applications choosing Xilinx FPGA Virtex-II Pro as the target device. We have also performed an ASIC design simulation with Microwind and DSCH version 3.0. Thus with algorithmic as well as architectural optimization, great performances in terms of speed and power dissipation are achieved. However, to accommodate different block and sub-block sizes prescribed in H.264/AVC or to provide flexible search ranges to suit any complex application, certain modifications need to be incorporated in the present architecture. For any block size, some of the channels of the MUXes can be disabled. For flexible ranges, more MUXes can be appended to the existing MUX banks in the PPSU. Further work involves the application of leakage power reduction techniques to the proposed architecture.

ACKNOWLEDGMENT

The authors wish to express their sincere gratitude to all the anonymous reviewers whose valuable suggestions helped to improve the paper.

REFERENCES

[1] A. Artieri and F. Jutand, "A versatile and powerful chip for real time motion estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '89)*, vol. 4, pp. 2453–2456, Glasgow, UK, May 1989.

[2] C.-M. Wu and D.-K. Yeh, "A VLSI motion estimator for video image compression," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 4, pp. 837–846, 1993.

[3] F.-M. Yang, S. Wolter, and R. Laur, "VLSI architecture for HDTV motion estimation based on block-matching algorithm," in *Proceedings of the 7th International Conference on VLSI Design*, pp. 287–290, Calcutta, India, January 1994.

[4] N. Roma and L. Sousa, "Efficient and configurable full-search block matching processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1160–1167, 2002.

[5] M. Abbas, B. Talha, S. Khan, and A. Abbas, "A motion estimation chip for block based MPEG-4 video applications," in *Proceedinga of the 7th International Multi Topic Conference (INMIC '03)*, pp. 253–257, Islamabad, Pakistan, December 2003.

[6] M. Sayed and W. Badawy, "A fully parallel-pipelined architecture for full-search block-based motion estimation," in *Proceedings of the 14th International Conference on Microelectronics (ICM '02)*, pp. 24–27, Beirut, Lebanon, December 2002.

[7] L. De Vos and M. Schobinger, "Efficient architecture of a programmable block matching algorithm," in *Proceedings of the International Conference on Application-Specific Array Processors*, pp. 560–571, Venice, Italy, October 1993.

[8] Y.-K. Lai and L.-G. Chen, "A data-interlacing architecture with two-dimensional data-reuse for full-search block matching algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, pp. 124–127, 1998.

[9] D. Xu, R. Gao, and H. Batatia, "An improved parallel architecture for MPEG-4 motion estimation in 3G mobile applications," in *roceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 2, pp. 689–692, Hong kong, April 2003.

[10] S.-H. Wang, W.-L Tao, C.-N. Wang, W.-H. Pong, and H. Chiang, "Platform based design of all binary motion estimation (ABME) with bus interleaved architecture," in *Proceedings of International Symposium on VLSI Design, Automation and Test (VLSI-TSA-DAT '05)*, pp. 241–244, Hsinchu, Taiwan, April 2005.

[11] R. Korah and J. R. P. Perinbam, "A novel coarse-to-fine search motion estimator," *Information Technology Journal*, vol. 5, no. 6, pp. 1073–1077, 2006.

[12] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148–157, 1993.