Efficient Block-based Motion Estimation Architecture using Particle Swarm Optimization

Vani Rajamanickam[1] and Sangeetha Marikkannan[2]
[1]Meenakshi College of Engineering, Anna University, India
[2]Karpaga Vinayaga College of Engineering and Technology, Anna University, India

Abstract: *High speed video transmission is the key to achieve high quality live or through offline streaming. Block Matching Motion Estimation (ME) is adopted in video coding standards to improve the performance in terms of speed and at the same time, the power consumption should be minimal. The paper proposes an efficient block-based ME architecture, in which the motion vectors are obtained by searching for the best match in the previous frame. A resizable smart snake order is utilized for scanning the frames of different block sizes which improves the data reuse efficiency. The architecture is based on applying the global search ability of Particle Swarm Optimization (PSO) that reduces the number of logic elements. The parallel execution involved in the processing of sub-regions in the search window enables the architecture to achieve high speed. The proposed work coded in Verilog hardware description language, and implemented with Altera cyclone II FPGA, operates at a maximum frequency of 265.01MHz. It is observed that the total thermal power dissipation is 74.27 mw, making it suitably efficient for low power implementation of ME.*

Keywords: *Resizable smart snake scan, swarm optimization, ME, block matching.*

Received October 18, 2013; accepted June 9, 2014

1. Introduction

The H.264 advanced video codec is an ITU standard, for encoding and decoding videos with a target coding efficiency twice that of H.263, and with quality comparable to that of the H.262. It addresses ongoing applications ranging from high definition digital video disc or BluRay for living room entertainment with large screens, to digital video broadcasting for handheld terminals with small screens [4]. The compression efficiency is not the result of a single feature, but rather a combination of a number of encoding tools. Figure 1 depicts the block diagram of the H.264 encoder [25]. The encoder contains three steps: Prediction, transformation/quantization and entropy encoding. Macro block mode decision and Motion Estimation (ME) are the most computationally expensive processes. Mode decision is a process in which for each block-size, the bit-rate and distortion are calculated, by encoding and decoding the video.

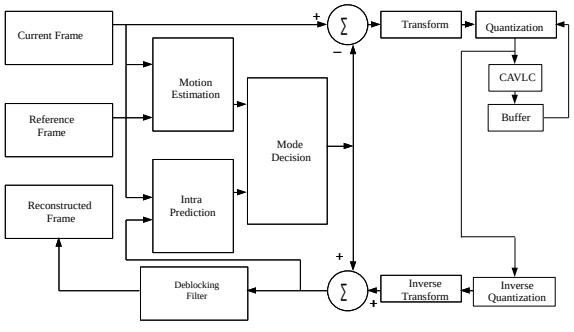

Figure 1. H.264 Encoder.

ME is an inter-coding technique. Inter coding refers to a mechanism of finding the co-relation between two frames, which are not far away from each other (one called the reference frame and the other called the current frame), and then encoding the information which is a function of this co-relation instead of the frame itself. In the H.264 encoder system, ME is part of the prediction step. It contains mainly two parts: The Integer Motion Estimation (IME) and the fractional ME. According to the runtime profile of the H.264 Joint Model (JM) encoder, the IME consumes close to 60% of the total encoder time, and up to 90% when the fractional ME is included. Therefore, efficient ME algorithms and hardware architectures for IME are needed. This paper is about the IME algorithm.

In the IME, the current frame is divided into non-overlapping NxN Macroblocks (MB). In the reference frame, for each MB a search window is defined around a point. Each point in the search window corresponds to a candidate MB to predict the current MB. A distortion measure is defined to measure the similarity between the candidate MB and the current MB. A search is performed for the best matched candidate MB with maximum similarity within the search window. The displacement of the best matched MB from the current MB is the Motion Vector (MV) [6, 13, 27]. There are many mismatch measures such as the Sum of Absolute Difference (SAD), the Sum of Squared Difference (SSD) and the Sum of Absolute Transformed Difference (SATD). Our method uses the SSD as the performance evaluation criterion.

The block matching approach is the most widely used ME module, and it is also adopted in all the existing video coding standards due to its simplicity and good performance. Among all the Block Matching Algorithms (BMAs), the Full Search Block Matching Algorithms (FSBMA) is the most popular [7]. An important coding tool of H.264 is the variable block size matching algorithm for ME. Variable Block Size Motion Estimation (VBSME) provides more accurate predictions, when compared to the traditional Fixed Block Size Motion Estimation (FBSME). In the FBSME, if an MB consists of two objects with different motion directions, the coding performance is poor [10].

On the other hand, in the VBSME, the MB can be divided into smaller blocks, in order to fit the different motion directions [3]. And hence, the coding performance is improved. In the H.264/AVC, an MB can be divided into seven kinds of blocks of 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, and 16x16, as shown in Figure 2. Though, the VBSME can achieve a higher compression ratio, it involves huge computational complexity and also increases the difficulty of ME hardware implementation.

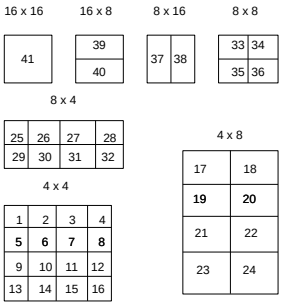

Figure 2. Variable block size.

Recent researchers apply the probabilistic searching method, such as Particle Swarm Optimization (PSO) to address ME problems. The PSO algorithm utilizes the global optimization characteristics of the swarm intelligent, to obtain better global estimation, especially for video sequences with violent motion [20]. Since the global optimal solution is found only after finishing several iterations, the computational complexity of the PSO algorithm becomes very high. This could be overcome by using hybrid algorithms [28]. This paper utilizes the benefits of the PSO in block based ME, by applying the resizable snake scan concept.

The rest of this paper is organized as follows: Section 2 presents a literature review of existing optimization algorithms. Section 3 discusses the existing scanning methods. Section 4 gives an overview of PSO. In section 5, an efficient ME architecture based on PSO is proposed to which the resizable smart snake scan is applied. The performance of ME architecture is analyzed in terms of the SSD. The implemented results using Altera cyclone II FPGA, are presented in section 6. In section 7, the comparison of the existing and proposed architecture is discussed. Finally, a brief conclusion is given in section 8.

2. Review of Optimization Algorithms

The existing block matching algorithms are the three step search proposed by Koga *et al.* [14], the New Three Step Search (NTSS) proposed by Li *et al.* [15], the Four Step Search (4SS) proposed by Po and Ma [21], the Diamond Search (DS) [18] and the Hexagon Search (HS) [1]. The drawback is that these algorithms are susceptible to being trapped in local optima on the error surface. In order to avoid the problem of local minima, several approaches are presented in the literature to solve the problem of ME. One such approach is the Genetic Algorithm (GA) which has been considered for block matching motion estimation by Ouaazizi *et al.* [5]. However, the drawback is that the algorithm becomes complex and suffers from a high computationality burden. The concept of Simulated Annealing (SA) is used by Shi *et al.* [24] to control the searching process and to adaptively choose the intensive search region. In addition to the GA and SA, PSO is used to solve the problem of ME in the literature [2, 8, 9, 16, 19, 20, 22, 23].

PSO is a population based stochastic optimization technique developed by Kennedy and Eberhart [12]. Block matching is an optimization problem, and it is necessary to reduce the complexity and to improve its performance. The PSO method is used to solve the problem, by selecting the initial individuals based on random points and this improves the speed of convergence. The PSO iterations can also achieve faster convergence when the temporal correlation with the collocated block in the adjacent frame is exploited. The algorithm proposed by Jacob and Pandian[8] using the PSO, maintains high estimation accuracy in block-based motion estimation. The good-point set theory is used to choose better points than random selection to accelerate the convergence of the algorithm. The PSO, based on the good-point set theory, reduces the deviation between random numbers and improves the estimation accuracy [16].

The correlation between local statistical characteristics, scene duration and scene change is analysed, and based on this analysis, a scene change algorithm for the H.264 codec using the PSO is used. It reduces the complexity involved in the scene changes between frames [22]. In the research work proposed by Jalloul [9], a ME scheme based on PSO strategies is applied to find the optimal MVs for all the macroblocks of a given frame in parallel, which yields a tremendous speed up for encoding and decoding applications. The number of search points can be reduced using a pattern based PSO in ME. The PSO is modified to get accurate solutions in addressing ME

problems [19]. Our research focuses on applying the concept of PSO with the scanning algorithm in ME, to achieve better speed and to make it suitable for video coding applications.

3. Existing Scanning Methods

Full Search (FS) is the simplest, and the most computation expensive search algorithm, which exhaustively searches all the candidate points in the search range. Several fast search algorithms have been developed to reduce the computational complexity of the FS. They can be roughly classified into lossy and lossless algorithms. Existing ME implementations are done using the FS, since it has hardware friendly features, such as a regular data flow and low control overhead [14].

There are different methods of scanning. Zigzag scanning is used to convert a 2D image matrix into a 1-D vector at the encoder. The horizontal zigzag scan mode is used, in order to reduce the partitions and improve the IO utilization of the search window memory. Two approaches are applied in [9]. First, the reference pixels are rearranged in the search window memory. And second, instead of reading the pixels in the same row, the design reads one column pixels from the rearranged search window and pushes them into the reference pixel buffer. The raster scan is effective in reusing data horizontally, with a relatively high data re-use ratio, but with redundant loading. The scan order introduced in [10], for reading and writing the reference data, improves the efficiency of memory accessing and also obtains high data re-use of the search area.

Data re-usability is improved slightly by another scanning order called the snake scan. In both the raster and snake scans, the data re-use ratio and search window size are fixed. The smart snake scan in [8, 14] achieves variable data re-use ratios and minimum redundant data loading. Based on the smart snake scan, the resizable smart snake scan is used, whose range of scanning depends on the variations involved between successive frames based on the PSO.

4. Overview of Particle Swarm Optimization

PSO is an optimization method, based on Swarm intelligence. The system is initialized by a set of random values, and then looks for the optimal solution by updating these candidates. In the PSO, potential solutions or particles move in the solution, while tracking the optimal particles. In the PSO, each particle periodically updates its own velocity and position, as given in Equations 1 and 2:

$$V_{in}(t+1)=W\times V_{i1}(t)+C_1\times R_1(.)\times(P_{in}-X_{in})+C_2\times R_2(.)\times(P_{gn}-X_{in}) \quad (1)$$

$$X_{in}(t+1)=X_{in}(t)+V_{in}(t+1),\ 1\le i\le N,\ 1\le n\le D \quad (2)$$

Where N is the number of particles and D is the dimensionality; $V_i=(V_{i1}, V_{i2}, \ldots, V_{in})$, $V_{in} \in [-V_{max}, V_{max}]$ is the velocity vector of particle i, which decides the particle's displacement in each iteration. Similarly, $X_i=(X_{i1}, X_{i2}, ..., X_{in})$, $X_{in} \in [-X_{max}, X_{max}]$ is the position vector of particle i, which is a potential solution in solution space. W is the inertial weight which decreases linearly; C_1 and C_2 are both positive constants, called the acceleration factors, which are generally set to 2.0; $R_1(.)$ and $R_2(.)$ are two independent random numbers distributed uniformly over the range [0, 1]; and P_g, P_i are the best solutions developed by the group and itself respectively [12]. In the t+1 time iteration, particle i uses P_g and P_i as the heuristic information, to update its own velocity and position. The first terms in Equation 1 represent diversification, while the second and third, represent intensification. A balance between diversification and intensification is to be achieved, based on which the optimization progress is possible.

5. Proposed Method

This section proposes the architecture for the efficient block-based ME. It utilizes the resizable smart snake scan order, and finds the sum of squared difference. The best match is obtained by using PSO. Data flow is discussed in section 4.4, which explains the processing of the current frame and reference frame, in and out of the estimator.

5.1. Concept of PSO

The PSO is initialized with a group of random particles and in every iteration, each particle is updated by the two best values, P_i and P_g which indicate the Pbest and Gbest respectively, as shown in Equation 1. Here the population chosen is N=5 particles. These particles are placed in the sub-region of the search window. Each particle traverses in the corresponding sub-regions to find the minimum Pbest value, using the SSD calculation. The particle's traversing order is in the form of a snake scan order. After all the particles are searched in the corresponding sub-regions, the minimum Pbest value of the five particles are used, for finding the Gbest value of the current MB. These Gbest values give the motion vector of the current MB. The iterations of each particle are limited only by the size of the sub-regions, where the particles traverse.

5.2. Motion Estimation Architecture

The input from the external memory is stored in the registers, as shown in the block diagram, Figure 3. From the Register Array (RA), the input is then sent to the process control unit. The reference frame is stored in RAM. The reconfigurable search RA R_0, R_1, R_2, R_3, R_4 utilizes the array of registers for the data reusability of each particle in its sub-region. The block based algorithm is implemented using the PSO. All these are

implemented in the blocks called the pattern analyser and Process Element (PE).

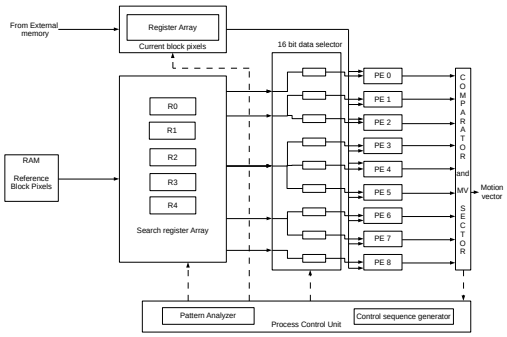

Figure 3. Block diagram of the ME architecture.

The image/frame is analyzed pixel by pixel, in the pattern analyzer which is present in the process control unit. The result is either stored in the registers or given to the data selector, which is connected to the process element. In this unit, the particles are processed after the application of the clock, and the SSD calculation is scheduled and then performed. The PE consists of the XOR array, adder and accumulator, which performs the processing of the current and reference frame. Figure 4 shows the partial schematic of PE during implementation.

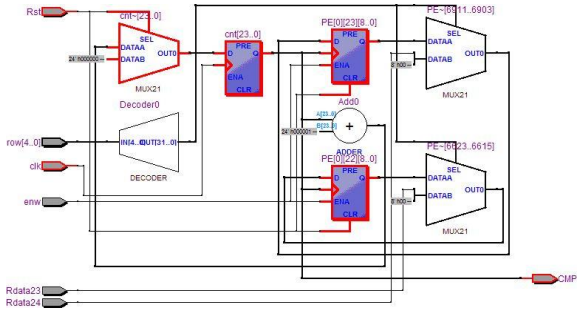

Figure 4. Partial schematic of the PE as implemented.

The data selector is used to select the block to be processed. When one frame is processed, the next frame to be processed is stored in the search RA. The output of the PSO from the pattern analyzer is sent to the control sequence generator, which generates the sequences. Then, the output of the process elements is sent as MV in the last stage. The processing also takes advantage of the resizable smart snake scanning concept. The iteration is continued for the rest of the reference frames.

This architecture has 2 benefits: First, the proposed work reduces the number of logic elements and is suitable to implement the FS. This is possible as the search size (N) is decided using the concept of the PSO. The other advantage is that in the programming, the parallel execution of the coding is done, for which the concept of iteration is use. And it can be effectively applied for images/frames which have varying backgrounds.

5.3. Resizable Snake Scan

In the resizable smart snake scan order, the search window of size MxN is split into several sub-regions. Figure 5 shows the search window for six sub-regions with M=N=32, and the snake scan is performed in each sub-region. The PSO chooses five particles, and here we place the particles in each sub-region.

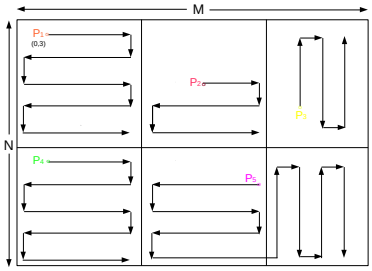

Figure 5. Resizable snake scan.

Based on the positioning of the particle, the particle traverses from top to bottom or bottom to top, or left to right or right to left, along the width of the sub-regions. Let us consider the particle P_1 in the first sub-region, whose position is (0, 3), as indicated in Figure 6. In step I_1, the reference pixels of size 16x16 are loaded into the PE array, and the SSD is calculated. In step I_2, the remaining row pixels of width 'W' are loaded by successive clock cycles, and the corresponding SSD values are calculated for the particles. While the particles traverse in the row, the reference pixels that are re-used, are moved to the RA.

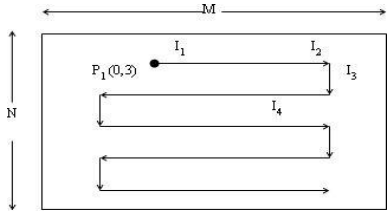

Figure 6. Resizable snake scan in a sub-region.

After processing the first row, step I_3 is used to move to the next row. In step I_4, the particle traverses in the reverse direction. In each step, the reused reference data is moved into the RA. For each SSD calculation, the values evaluated are compared with the Pbest value. If the new SSD value is smaller than the Pbest value, the Pbest value is replaced by the SSD value; otherwise, the older Pbest value is used for comparison. Since the sixth sub-region does not have any particle, the particle P_5 continues the scanning in the next sub-region. Thus, the resizable smart scan is applied to the all five particles in different positions at the same time, which improves the speed of the algorithm.

5.4. Data flow for the ME Architecture

Let us consider M=2 and N=4. The RA contains (M-1) columns. The column indicates the data reusability of

the pixels stored in the RA, as shown in Figure 7. In step A, the current pixels and the reference pixels are propagated into the PE array.

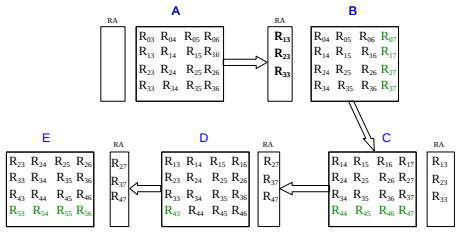

Figure 7. Data flow for the ME architecture.

After calculating the SSD for the first search point at (0, 3), the three (N-1) reference pixels are propagated into the column of the RA in step B. A new column of reference pixels R_{07}, R_{17}, R_{27}, and R_{37} is loaded into the PE array. The SSD of a new search point (0, 4) is calculated in step B. In step C, a new row of reference pixels R_{44}, R_{45}, R_{46}, and R_{47} is loaded into the PE array and the RA is not changed. In step D, only one reference pixel R_{43} is loaded, and data reusability is achieved. In step E, a new row of reference pixels is loaded and the iteration continues. Table 1 summarizes the frame data flow, in and out of the estimator. For each clock cycle, the data is read from the memory. The RA shows the reusability of the data. For each traversal, a search point is obtained.

Table 1. Data flow for the ME architecture.

Cycle	Step	Read from Memory	Data in RA	Search Point
0	initial A_1	R_{03}, R_{13}, R_{23}, R_{33}	_	
1	A_2	R_{04}, R_{14}, R_{24}, R_{34}	_	
2	A_3	R_{05}, R_{15}, R_{25}, R_{35}	_	
3	A_4	R_{06}, R_{16}, R_{26}, R_{36}	_	(0, 3)
4	B	R_{07}, R_{17}, R_{27}, R_{37}	R_{13}, R_{23}, R_{03}	(0, 4)
5	C	R_{44}, R_{45}, R_{46}, R_{47}	R_{13}, R_{23}, R_{03}	(1, 4)
6	D	R_{43}	R_{27}, R_{37}, R_{47}	(1, 3)
7	E	R_{53}, R_{54}, R_{55}, R_{56}	R_{27}, R_{37}, R_{47}	(2, 3)

5.5. Sum of Squared Difference

The performance is analysed in terms of the SSD, which is a widely used simple algorithm for measuring the similarity between the image blocks. The difference between each pixel in the block is used for comparison. These differences are squared to create a block similarity, and a summation is performed on the obtained output. The SSD is preferred in our work, as it magnifies even the small differences, and can be used for object recognition, the generation of disparity maps for stereo images, and ME for video compression.

$$SSD = \Sigma_{i,j \epsilon w} (I_1(i, j) - I_2(x + i, y + j))^2 \quad (3)$$

Where, *i*, *j*-> row and column of current frame I_1, x, y> row and column of reference frame I_2, w-> window size.

Initially, the search size is chosen for the minimum value 4x4. This could be extended for different block sizes depending on the variations between successive frames. The partial 4x4 current frame and reference frame of a Foreman sequence, applied as inputs to the motion estimator, are shown in Table 2.

Table 2. Current frame and reference frame.

Current Frame				Reference Frame			
127	98	98	97	104	103	97	100
98	96	101	100	99	97	95	100
99	98	98	97	98	97	96	96
98	96	101	100	101	102	102	98

As indicated in Equation 3, the difference is estimated and the sum of the squared output is obtained as 659. The simulation waveform is shown in Figure 8 below. The process is repeated for the next four pixels in the reference frame and so on. This process continues until the best match is found. Then the process stops searching the rest of the pixels in the reference frame.

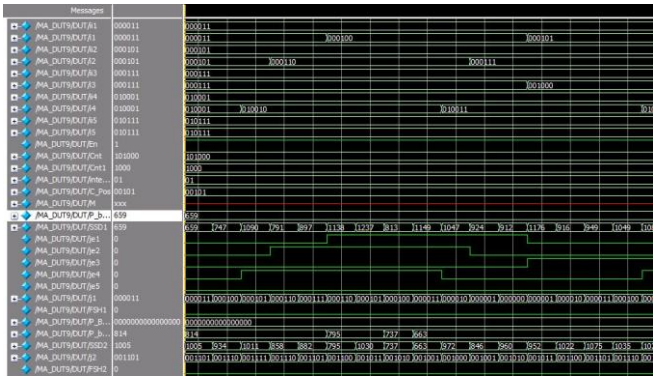

Figure 8. Simulation result of SSD calculation.

6. Implementation Results

The performance analysis of the ME architecture is being coded in Verilog HDL. The implementation results are shown in Tables 3 and 4. The architecture is implemented, using EP2C20F256C6 cyclone II Altera FPGA. The performance of the architecture is done in terms of the number of multipliers, the number of adders, the number of registers, the number of logic elements and the operating frequency.

Table 3. Synthesis report of ME architecture.

FPGA	Altera Cyclone II EP2C20F256C6
Total logic elements	12,152
Total Combinational functions	12,120
Total registers	1,165
Total pins	63
Maximum Operating frequency(MHz)	265.01 MHz

Table 4. Power analysis report of ME architecture.

Total Thermal Power Dissipation (mw)	Core Static Thermal Power Dissipation (mw)	I/O Thermal Power Dissipation (mw)
74.27	47.40	26.87

From the implementation result, it is clear that 265.01 MHz of frequency of operation is achieved. The total power dissipation of the proposed architectures is 74.27 mW, which makes it suitable for low power applications.

7. Comparison of ME Architectures

The proposed work is compared with the other ME architectures, which includes both the traditional FBSME and VBSME. The variable block size ME architecture in [3] utilizes the modified snake scan order and operates at a frequency of 108 MHz, whereas the architecture in [11] applies raster scanning for processing different blocks and operates at 130 MHz . The architecture in [26] uses the snake scan for processing variable block size frames at the operating frequency of 250 MHz, with a power consumption of 118mw. The FBSME architecture in [10] operates at 200 MHz. It is based on the Zig-Zag scan method and utilizes 40K logic elements. Zuo *et al*. [29] proposes an architecture in, which uses an efficient sub pixel search algorithm that will simplify the process of variable block size ME. It is capable of processing frames at an operating frequency of 200 MHz, with 530k gate count. In [17], the VBSME architecture operates at a maximum frequency of 200 MHz with a power consumption of 729 mw. It is based on the raster scan method. From the graph shown in Figure 9, it is clear that the proposed design method based on PSO achieves a high speed of 265.01 MHz, when compared to other existing methods suitable for real time video applications.

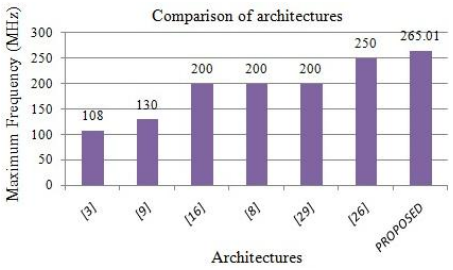

Figure 9. Comparison of fixed block size and variable block size ME architectures.

8. Conclusions

A high speed and efficient ME VLSI architecture is proposed in this paper. Based on applying the global search ability of the PSO, the number of logic elements is reduced. The Resizable smart scan order is used to improve the data reuse efficiency. The proposed architecture is coded in hardware description language, and implemented with Altera cyclone II FPGA. The parallel processing of the input coefficients is done to achieve a high speed of 265.01MHz, suitable for handling high speed, low power multimedia applications. Hybrid PSO techniques can be used to further implement parallel processing, so as to reduce the computational complexity.

References

[1] Belloulataa K., Zhub S., Tianb J., and Shen X., "A Novel Cross-Hexagon Search Algorithm for Fast Block Motion Estimation," *in proceedings of 7th International Workshop on Systems, Signal Processing and their Applications*, Tipaza, pp. 1-4, 2011.

[2] Cai J. and Pan W., "On Fast And Accurate Block-Based Motion Estimation Algorithms Using Particle Swarm Optimization," *International Journal of Information Sciences*, vol. 197, pp. 53-64, 2012.

[3] Chen C., Chien S., Huang Y., Chen T., Wang T., and Chen L., "Analysis and Architecture Design of Variable Block Size Motion Estimation for H.264/AVC," *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 3, pp. 578-593, 2006.

[4] Chen T., Lian C., and Chen L., "Hardware Architecture Design of an H.264/AVC Video Codec," *in proceeding of Asia and South Pacific Conference on Design Automation*, Yokohama, pp 750-757, 2006.

[5] El Ouaazizi A., Zaim M., and Benslimane R., "A Genetic Algorithm for Motion Estimation," *International Journal of Computer Science and Network Security*, vol. 11, no. 4, pp. 165- 172, 2011.

[6] Gu M., Yu N., Zhu L., and Jia W., "High Throughput and Cost Efficient VLSI Architecture of Integer Motion Estimation for H.264/AVC," *Journal of Computational Information Systems*, vol. 7, no. 4, pp. 1310-1318, 2011.

[7] Huang Y., Chen C., Tsai C., Shen C., and Chen L., "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results," *Journal of VLSI Signal Processing*, vol. 42, no. 3, pp. 297-320, 2006.

[8] Jacob A. and Pandian I., "An Efficient Motion Estimation Algorithm Based on Particle Swarm Optimization," *International Journal of Electronics Signals and Systems*, vol. 3, no. 1, pp. 223-5969, 2013.

[9] Jalloul M., "A Novel parallel Computing Approach for Motion Estimation Based on Particle Swarm Optimization," *in Proceeding of International Conference on Engineering of Reconfigurable Systems and Algorithms*, Las Vegas, 2013.

[10] Jang S., Kim S., Lee J., Choi G., and Ra J., "Hardware Software Co-Implementation of a H.263 Video Codec," *IEEE Transaction on Consumer Electronics*, vol. 46, no. 1, pp. 191-200, 2000.

[11] Kao C. and Lin Y., "A High-Performance and Memory-Efficient Architecture for H.264/AVC Motion Estimation," *in Proceeding of IEEE International Conference on Multimedia and Expo*, Hannover, pp. 141-144, 2008.

[12] Kennedy J. and Eberhart R., "Particle Swarm Optimization," *in Proceeding of IEEE*

International Conference on Neural Networks, Washington, pp. 1942-1948, 1995.

[13] Kim S., Wook B., Burm Q., Wook J., and Sunwoo M., "Data Reusable Search Scan Methods for Low Power Motion Estimation," *in Proceeding of IEEE International Symposium on Circuits and Systems*, Seoul, pp. 1556-1559, 2012.

[14] Koga T., Iinuma K., Hirano A., Iijima Y., and Ishiguro T., "Motion Compensated Interframe Coding for Video Conferencing," *in Proceeding of National Telecommunication Conference*, New Orleans, pp. 1- 5, 1981

[15] Li R., Zeng B., and Liou M., "A New Three-Step Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438-442, 1994.

[16] Liu X., Xuan S., and Liu F., "An Advanced Particle Swarm Optimization Based on Good-Set Point and Application to Motion Estimation," *in Proceeding of the 9th International Conference on Intelligent Computing Theories and Technology*, Nanning, pp. 494-502, 2013.

[17] Liu Z., Song Y., Shao M., Li S., Li L., Goto S., and Ikenaga T., "32-Parallel SAD Tree Hardwired Engine for Variable Block Size Motion Estimation in HDTV 1080P Real-Time Encoding Application," *in Proceeding of IEEE Workshop on Signal processing Systems*, Shanghai, pp. 675-680, 2007.

[18] Ndili O. and Ogunfunmi T., "Algorithm and Architecture Co-Design of Hardware-Oriented, Modified Diamond Search for Fast Motion Estimation in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 9, pp. 1214-1227, 2011.

[19] Pandian S., Bala G., and Anitha J., "A Pattern Based PSO Approach for Block Matching in Motion Estimation," *Journal of Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1811-1817, 2013.

[20] Ping Z., Ping W., and Hongyang Y., "A Novel Block Matching Algorithm Based on Particle Swarm Optimization With Mutation Operator and Simplex Method," *WSEAS Transactions on Systems and Control Journal*, vol. 6, no. 6, pp. 207-216, 2011.

[21] Po L. and Ma W., "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," *IEEE Transactions of Circuits and System for Video Technology*, vol. 6, no. 3, pp. 313-317, 1996.

[22] Rao L. and Rao D., "Dynamic Threshold and Fast Motion Estimation Algorithm Based on PSO for H.264's Scene Change Detection," *International Journal of Computer Science and Network Security*, vol. 11, no. 5, pp. 166-171, 2011.

[23] Ren R., Manokar M., Shi Y., and Zheng B., "A Fast Block Matching Algorithm for Video Motion Based on Particle Swarm Optimization and Motion Prejudgment," *The Computing Research Repository*, pp. 1-10, 2006.

[24] Shi Z., Fernando W., and Kondoz A., *Simulated Annealing for Fast Motion Estimation Algorithm in H.264/AVC, Simulated Annealing-Single and Multiple Objective Problems*, InTech, 2012.

[25] Vani R. and Sangeetha M., "Survey on H.264 Standard," *in Proceeding of International Conference on advances in Computer Science and Information Technology*, Bangalore, pp. 397-410, 2012.

[26] Wen X., Au O., Xu J., Fang L., Cha R., and Li J., "Novel RD-Optimized VBSME with Matching Highly Data Re-Usable Hardware Architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 2, pp. 206-219, 2011.

[27] Werda I., Chaouch H., Samet A., Ayed M., and Masmoudi N., "Optimal DSP Based Integer Motion Estimation Implementation for H.264/AVC Baseline Encoder," *The International Arab Journal of Information Technology*, vol. 7, no. 1, pp. 96-104, 2010.

[28] Zhang P., Wei P., Yu H., and Wang Z., "Simplex Particle Swarm Optimization for Block Matching Algorithm," *in proceeding of International Symposium on Intelligent Signal Processing and Communication Systems*, Chengdu, pp. 1-4, 2010.

[29] Zuo S., Wang M., and Xiao L., "An Efficient VLSI Computation Reduction Scheme in H.264/AVC Motion Estimation," *WSEAS Transactions on Signal Processing*, vol. 10, no. 1, pp. 178-187, 2014.

Vani Rajamanickam received her B.E. degree in Electronics and Communication Engineering from Bharathiar University, Coimbatore and M.E. degree in Communication Systems from Anna University, Chennai. She is currently pursuing Ph.D. in Anna University, Chennai and also working as Assistant Professor in the Electronics and Communication Engineering Department, Meenakshi College of Engineering, Chennai. She is member of the ISTE and IEICE.

Sangeetha Marikkannan born in 1975 in Tamilnadu, India, received her B.E. degree from the Bharathidasan University, Trichy, in 1996 and M.E. degree from the University of Madras in 1999. She is currently with Karpaga Vinayaga College of Engineering and Technology in the Department of Electronics and Communication Engineering and Ph.D. degree with the Anna University, Chennai, India. She is a member of the IEEE and ISTE. Her research interests are Hardware/Software Co-design and High Level Synthesis.