A Novel Low-Power Motion Estimation Design for H.264

Maria G. Koziri, Adonios N. Dadaliaris, Georgios I. Stamoulis
Department of Computer and Communication Engineering,
University of Thessaly, Volos - Greece
{mkoziri, adadalia, georges}@uth.gr

Ioannis X. Katsavounidis
InterVideo, Inc.
46430 Fremont Blvd.,
Fremont, CA 94538, USA
ioannis@intervideo.com

Abstract

The H.264 video coding standard can achieve considerably higher coding efficiency than previous video coding standards. The keys to this high coding efficiency are the two prediction modes (Intra & Inter) provided by H.264 which adopt many new features such as variable block size searching, motion vector prediction etc. However, these result in a considerably higher encoder complexity that adversely affects speed and power, which are both significant for the mobile multimedia applications targeted by the standard. Therefore, it is of high importance to design architectures that minimize the speed and power overhead of the prediction modes. In this paper we present a new algorithm, and the architecture that implements it, that can replace the standard Sum of Absolute Differences (SAD) approach in the two main prediction modes, supports the variable block size motion estimation (VBSME) as it is defined in the standard and provide a power efficient hardware implementation without perceivable degradation in coding efficiency or video quality.

1. Introduction

Video has always been the backbone of multimedia technology. In the last two decades, the field of video coding has been revolutionized by the advent of various standards like MPEG-1 to MPEG-4 and H.261 to H.263, each addressing different aspects of multimedia. H.264[1] is a new standard which adds one more step in the endeavor towards video coding excellence and provides one-stop solution for wide range of applications. The primary goal of H.264 is to achieve higher compression while preserving video quality. In order to accomplish this goal, it uses the spatial and temporal predictions to eliminate the spatial and temporal data dependency.

One computational element that appears is met in both, Inter (Spatial) and Intra (Temporal) Prediction modes, is the Sum of Absolute Differences (SAD). In this paper we present a new algorithm that can replace SAD in the two main prediction modes, and the circuit that implements the proposed algorithm for inter prediction, also supporting variable block size motion estimation (VBSME), which results in better speed and significantly lower power. The rest of this paper is organized as follows: Section 2 reviews the Motion Estimation as it is presented in H.264, as well as the basic concepts of SAD. In Section 3 a description of the proposed algorithm is introduced. Section 4 presents the coding efficiency and video quality results achieved from the software implementation of the new algorithm. In Section 5 the architecture of the proposed algorithm is presented, while in Section 6 the results of the hardware implementation and the performance analysis are shown.

2. Background

2.1 Motion Estimation in H.264

For all the types of macroblocks, except the I (Intra), H.264 uses inter-prediction to compress them. Inter coding uses the temporal model. In this case the predicted frame is created from one or more past or future frames ('reference frames'). The process of finding the best predicted frame is known as Motion Estimation. The accuracy of the prediction can usually be improved by compensating for motion between the reference frame(s) and the current frame (Motion Compensation). A practical and widely-used method of motion compensation is to compensate for movement of rectangular sections or 'blocks' of the current frame. H.264 is a block-based motion-compensated hybrid transform codec, which supports a variety of block sizes (denoted as modes), varying from 16×16, 8×16, 16×8, 8×8, 8×4, 4×8 to 4×4 pixels as shown in fig.1. A separate motion vector is required for each partition or sub-macroblock, so we have a total of 41 MVs for each macroblock [2].

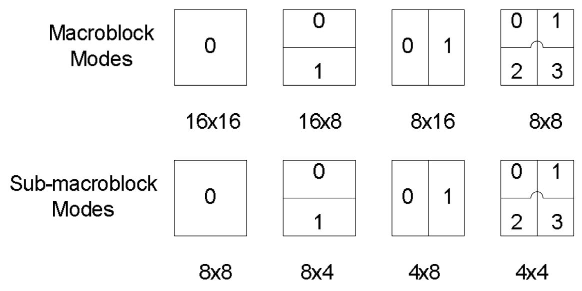


Figure 1. Macroblock and Sub-Macroblock partitions in H.264

Usually the criterion to find the matching block is the energy in the residual formed by subtracting the candidate block from the current M×N block, and the candidate region that minimizes the residual energy is chosen as the best match. However, in order to reduce the computational complexity, most real world application, among them H.264, uses the sum of absolute differences (SAD) [3]. Furthermore the H.264 reference software, when calculating the motion cost, takes also into account the bitrate cost for the motion vector. So in H.264 the criterion to find the matching block is the motion cost, which is given by the following equation

$$J(\vec{m},\lambda) = SAD\,(c, r(\vec{m})) + \lambda \times R(\vec{m} - \vec{p}) \qquad (1)$$

where $\vec{m} = (mv_x, mv_y)$ is the current candidate motion vector, $SAD\;(c, r(\vec{m}))$ is the Sum of Absolute Differences between current block and the candidate reference block, $\vec{p} = (mvp_x, mvp_y)$ is the predicted motion vector, $R(\vec{m} - \vec{p})$ is the number of bits needed to code the motion vector difference, and λ is the Lagrangian multiplier.

2.2 Sum of Absolute Differences (SAD)

The encoder typically selects the prediction mode, both in Intra and Inter Prediction, for each block that minimizes the motion cost given by equation (1). One of the main aspects of motion cost is the SAD of the predicted block P and the block to be encoded C, also known as L_1 distance between the vectorized form of the two blocks, P and C.

L_1-distance is a positive-definite metric defined over vectors in k-dimensional vector spaces by the corresponding L_1 norm of the difference vector, as follows:

Let $\underline{x}, \underline{y} \in R^k$. Then,

$$L_1(\underline{x}, \underline{y}) = \sum_{i=1}^{k} | x_i - y_i | \qquad (2)$$

It is easy to show the following properties of the L_1 norm

1. $L_1(x,y) = L_1(y,x)$ (symmetric)
2. $L_1(x,y) \geq 0$, with $L_1(x,y) = 0 \Leftrightarrow x=y$ (positive-definite)
3. $L_1(x,y) + L_1(y,z) \geq L_1(x,z)$ (Triangle inequality)

In addition to the L1-norm, there are other metrics used in vector spaces. The well-known Euclidean distance is perhaps the most popular, also known as L2-norm, defined as

$$L_2(\underline{x}, \underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^2 \right)^{1/2}$$

It is easy to show that L_2 has the same properties (1)-(3) as L_1.

We can consider the n^th-norm of two vectors as:

$$L_n(\underline{x}, \underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n}$$

and show that it has the same properties (1)-(3) above. The limit case ($n \to \infty$) known as L_∞ can be shown to be

$$L_\infty(\underline{x}, \underline{y}) = \lim_{n\to\infty} L_n(\underline{x}, \underline{y}) = \lim_{n\to\infty} \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n} \Rightarrow$$

$$L_\infty(\underline{x}, \underline{y}) = \max_{i \in \{1,\cdots k\}} | x_i - y_i |$$

Clearly, all L_n-norms try to quantify in a single number the amount of difference between two vectors. There is great debate over which norm is the best to use in order to express error in signal processing, and especially audio, image and video [4]. Traditionally, researchers use the L_2-norm as the minimization criterion for improving signal processing and compression. That is the main reason the peak signal-to-noise ratio (PSNR, a logarithmic representation of the L_2-norm) has been used throughout the signal processing literature to express signal quality. On the other hand, SAD, which is the L_1-norm, has been used as the basic computational block to find block matches in video compression, since it does not require the additional complexity of the multiplier needed for L_2-norms. This is a necessary compromise – one of many one needs to make – in order to have a practical implementation of a video encoder.

The spirit of our work is the same: we offer an alternative compromise to the L_1-norm, an alternative that is similar in spirit (but yet distinct) to the L_∞-norm that yields very good results at a significant reduction in computations and power, especially from a hardware point of view.

3. Proposed algorithm

In this section we introduce a new technique for approaching the problem of both, Intra and Inter mode decision. The base of this technique is to avoid the stage of addition, which increases significantly the power and delay cost at the hardware level.

For a given 4×4 block, according to equation (2), a total of 16 subtractions and 15 additions are needed in order to produce the SAD for one candidate reference block. In this paper we introduce a new algorithm which bypasses the additions needed for SAD and replaces them with pure comparisons among the corresponding absolute differences of two candidate blocks. It must be noted here that the new algorithm would not have achieved better delay and power results if we had used the standard comparator [5], [14], which has inferior power and delay characteristics. It was through a synergy with the novel implementation that we were able to achieve these goals with the proposed algorithm.

In order to implement the new approach, we first calculate the absolute difference between the corresponding pixels for each candidate reference block. This can be written as:

$$M_{ij} = | C_{ij} - P_{ij} | \tag{2}$$

where M, C, P are 4×4 arrays, and i, j are the indices specifying the single pixel within the 4×4 array.

So, for each candidate reference block there is an M array. The proposed algorithm compares the values of the corresponding elements of all the available M arrays and chooses the one with the smallest number of maximum values. The metric of the new algorithm is the number of greater values.

Let's suppose we have two candidate reference blocks. That means we have two M array, let's say M_k and M_{k+1}. The value of the new metric for the first candidate is the number of elements of array M_k which are greater than the corresponding elements of M_{k+1}, whereas the for the second candidate is the number of elements of array M_{k+1} which are greater than the corresponding elements of array M_k. This can be written as:

$$f_{k_i} = \begin{cases} 1, M_{k_i} > M_{(k+1)_i} \\ 0, M_{k_i} \le M_{(k+1)_i} \end{cases} \tag{3}$$

$$F_k = \sum_{i=0}^{S} f_{k_i} \tag{4}$$

where M_k and M_{k+1} are the MxN arrays with the absolute differences of the two candidate reference blocks, S is the total number of the elements of each array, i.e. S= MxN and is always equal to the block size. According to equations (3) and (4) the new metric F is a positive number, smaller or equal to S. For example, in the case of 4x4 blocks we have $0 \le F \le 16$.

We have seen in Section 2.2 that in H.264 the criterion to find the matching block is to minimize the equation (1). With the new algorithm equations (1) changes slightly as we replace SAD with the new metric. So, according to H.264 and in combination with the proposed algorithm the criterion for the matching block is to minimize the equation

$$\begin{aligned} &J(\vec{m}_1, \vec{m}_2, \lambda) = \\ &F(c, r(\vec{m}_1), r(\vec{m}_2)) + \lambda \times R(\vec{m}_1 - \vec{p}) \end{aligned} \tag{5}$$

where $F(c, r(\vec{m}_1), r(\vec{m}_2))$ is the new metric for the candidate reference block with corresponding motion vector $\vec{m}_1$ when compared with another candidate reference block with corresponding motion vector $\vec{m}_2$.

4. Software measurements

The proposed mode decision scheme has been integrated with the H.264 JM11.0 [6] codec for performance evaluation. It is compared with the original codec (which uses SAD) of H.264 in terms of bitrate and PSNR. We used 5 sequences from the Video Quality Experts Group (VQEG) [7], src13_ref__720x480_420, src15_ref__720x480_420, src18_ref__720x480_420, src19_ref__720x480_420 and src22_ref__720x480_420 all in 525 SD video format. The video sequences are 720×480 and have frame rate 30 fps. The total number of frames used in the simulation is 260 for each sequence, with an I-Frame for each 15 frames. The motion estimation scheme used is the "Full search" scheme with the search range set to 16 and number of reference frames set to 1 and the entropy coding method was CAVLC.
The simulation process was done with the RD-optimization parameter turned off and for various values of the quantization parameter. In particular the quantization parameter (QP) was set to QP=14, QP=28, QP=38 and QP=50 for I and P frames, so as to test the five sequences from low to high bitrates. The simulation results are presented in Figure 2, where one can see the rate distortion curves for the five tested sequences.

As it is clear from Fig. 2 the curves for JM 11.0 and the proposed algorithm are nearly identical, which means that there is no substantial difference between

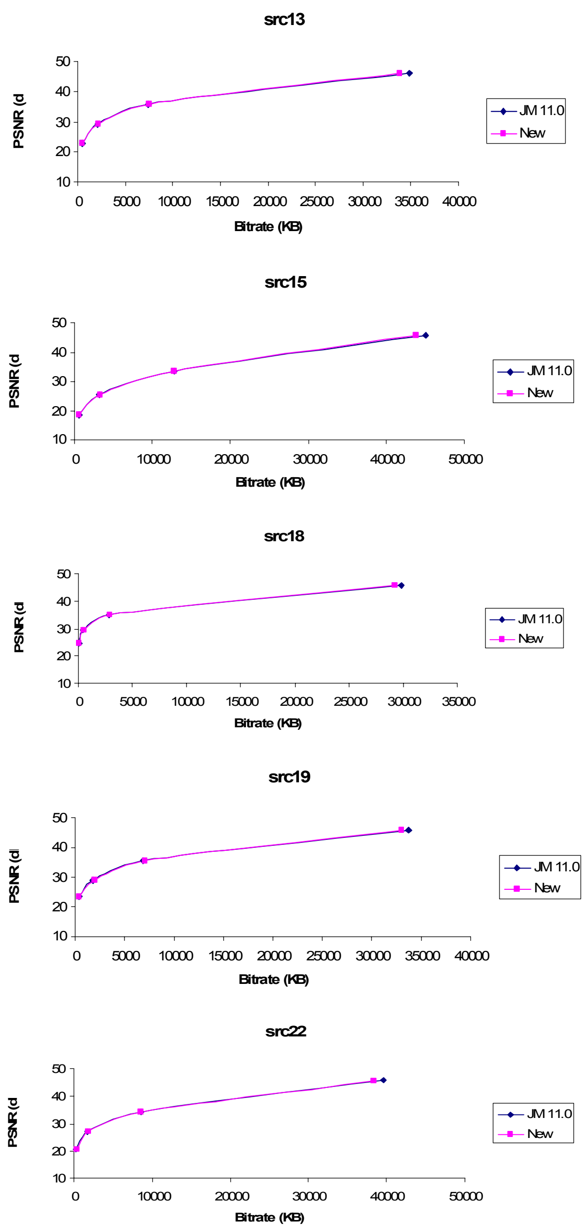
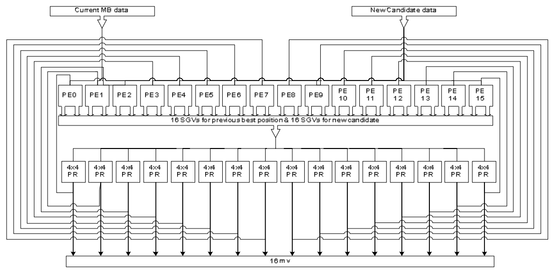


Figure 2. The rate distortion curves of five VQEG test sequences generated by JM 11.0 and the proposed algorithm.

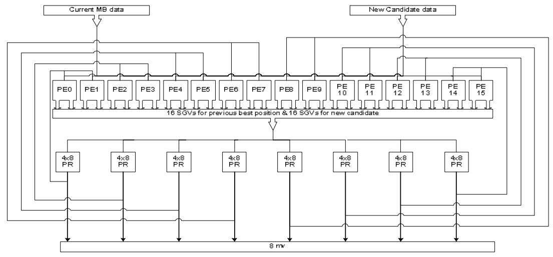
the performance of the two algorithms. In particular, results showed that for small QPs, i.e high bitrates the proposed architecture can achieve the same or even better performance compared to JM. For low-bitrate the performance of the proposed algorithm is reduced but remains competitive to that of JM11. The performance evaluation process also showed that the average PSNR remains practically the same, as the measurements gave us average differences that do not exceed the 0.2%. The aforementioned results confirm that the proposed algorithm achieves the same efficiency as that of SAD, with a significant reduction in the complexity of the encoder.

5. The architecture for the Variable Block Size Motion Estimation

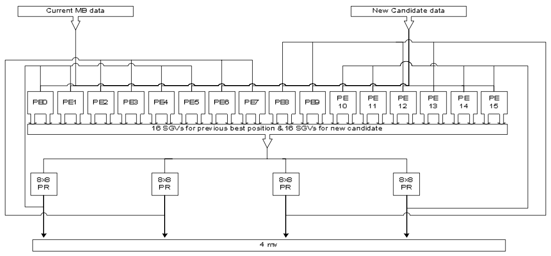
Based on the new algorithm presented in Section 3, we propose an architecture which can support Variable Block Size Motion Estimation (VBSME). VBSME is a new coding technique, presented in H.264, and provides more accurate predictions compared to traditional fixed block size motion estimation used by previous standards. There are many methods to support VBSME in hardware but the one used by most of the presented architectures, as for example in [8], [9], [10], [11], [12] and [13], is that of reusing the SADs of the smallest blocks, which are the blocks partitioned with the smallest block size , to derive the SADs of larger blocks. Nevertheless, the proposed architecture, because of the dependency of data occurring in each mode, cannot follow the same logic. On the contrary, it processes each mode separately, but because of the smaller execution time, and the lower power consumption of the new algorithm, the proposed architecture can process all modes in parallel. Furthermore, the proposed architecture takes into account the bit-rate cost, i.e. chooses the best candidate according to equation (5). In order to achieve the parallelism needed to implement the proposed architecture we use the modified predicted motion vectors, proposed in [12] and [13]. The core of the new algorithm is the comparison of each new candidate with the best found so far. It is clear that each mode has different selections, so it is obligatory to be processed independently. Because of this need for re-use of different data provided by the process of each mode the design consists by 7 similar blocks, one for each mode i.e. 4x4, 4x8, 8x4, 8x8, 8x16, 16x8 and 16x16, which work in parallel. In Fig.3 we present four of these seven blocks. The proposed architecture selects the best among two candidate reference blocks for the 41 different sub-blocks of a MB in two stages. In the first stage 16 PEs are used to calculate the new metric for the two reference blocks for the 16 4x4 blocks. In the second stage the metrics of the previous stage are used to calculate the metrics for the larger blocks and the value of equation (5) for each candidate block for each one of the 41 different sub-blocks of the processing MB.


Current MB data
New Candidate data
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8 PE9 PE 10 PE 11 PE 12 PE 13 PE 14 PE 15
16 SGVs for previous best position & 16 SGVs for new candidate
4x4 PR
16 mv

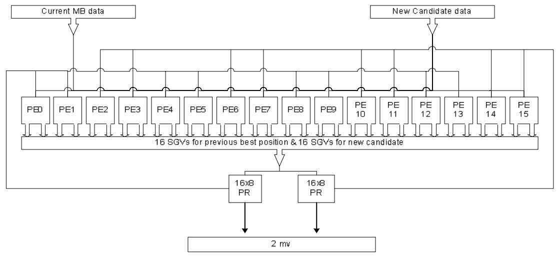
a. 4x4 block size motion estimation


Current MB data
New Candidate data
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8 PE9 PE 10 PE 11 PE 12 PE 13 PE 14 PE 15
16 SGVs for previous best position & 16 SGVs for new candidate
4x8 PR
8 mv

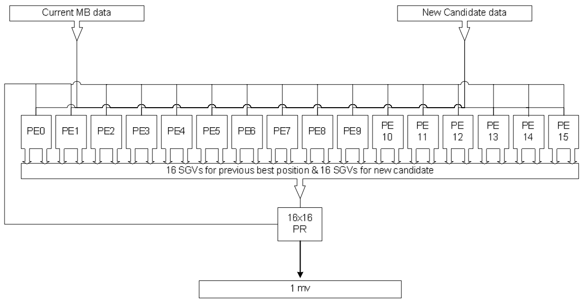
b. 4x8 block size motion estimation


Current MB data
New Candidate data
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8 PE9 PE 10 PE 11 PE 12 PE 13 PE 14 PE 15
16 SGVs for previous best position & 16 SGVs for new candidate
8x8 PR
4 mv

c. 8x8 block size motion estimation


Current MB data
New Candidate data
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8 PE9 PE 10 PE 11 PE 12 PE 13 PE 14 PE 15
16 SGVs for previous best position & 16 SGVs for new candidate
16x8 PR
16x8 PR
2 mv

d. 16x8 block size motion estimation


Current MB data
New Candidate data
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PE8 PE9 PE 10 PE 11 PE 12 PE 13 PE 14 PE 15
16 SGVs for previous best position & 16 SGVs for new candidate
16x16 PR
1 mv

e. 16x16 block size motion estimation

Figure 3. The block diagram of the proposed architecture

Each PE element presented in stage 1 has as inputs 16 pixels of the current block, 16 pixels of the new candidate and a signal which selects which of the two previous candidates was the best, as shown in Fig.4. At the beginning, the absolute differences among the pixels of current block and those of the new candidate are computed. Then each absolute difference (AD) produced by new candidate is compared with the corresponding absolute difference produced of the previous best candidate. If the AD of candidate 1 is greater than that of candidate 2 the Sum of Greater Values (SGV) of the previous best candidate is increased by 1, else if the AD the new candidate is greater than that of candidate 1 the Sum of Greater Values (SGV) of candidate 2 is increased by 1. When all the absolute differences are compared we have the final values of the new metric (SGV) for the two candidates. In the next stage, after all the 16 SGVs for each of the previous and new candidates have been calculated for all the 4x4 blocks, they are used by the different modes' processors in order to find the best of the two candidates for each mode. A block diagram of such a processor is shown in Fig. 5. In all mode processors the inputs are the SGVs of the 4x4 blocks, as computed in stage 1 and the corresponding bit-rate cost for each candidate position. In order to calculate the SGV for the current mode the SGVs of the appropriate 4x4 blocks are added. After the SGV for each candidate for the current mode is calculated, the corresponding bit-rate cost (shown as mv1 and mv2 in Fig.5) is added in order to have the final motion cost for each candidate. The two motion costs are compared and the one with the minimum cost is chosen.

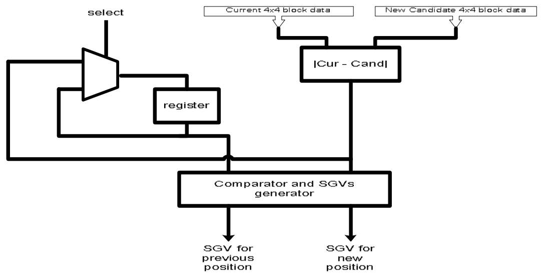

select
Current 4x4 block data
New Candidate 4x4 block data
|Cur - Cand|
register
Comparator and SGVs generator
SGV for previous position
SGV for new position

Figure 4. Block diagram of a PE

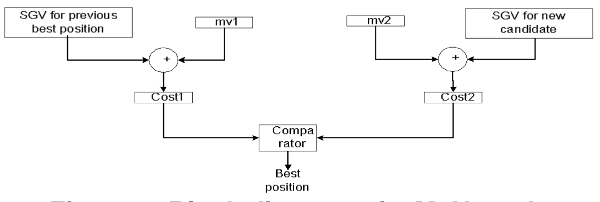


Figure 5. Block diagram of a MxN mode processor

6. Performance analysis

The implementation of the architecture presented in Section 5 has been captured using VHDL and synthesized with Synopsys Design Compiler® using the UMC 130nm CMOS standard cell technology (1.20V). The power estimates was obtained by Synopsys PrimePower®. The performance of the proposed architecture is shown in Table 1.

Table 1. The proposed VBSME chip performance

Algorithm	Variable block size full search motion estimation
Number of PE	112
Block Size	4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16
Technology	UMC 130nm CMOS (1.20V)
Gate Count	370k
Max Frequency	241 MHz
Power Consumption	95 mW @ 233MHz

The proposed architecture fulfills the selection of one search position among two candidates using the VBSME in every clock cycle.

For a frame with size of W x H pixels and search range of P_x x P_y pixels and frame rate of F fps, where W is the frame's width, H is the frame's height, P_x is the search range's width and P_y is the search range's height, the processing requirements (in units of search-positions/s) are given by the equation (6):

$$SP = \frac{W}{16} \times \frac{H}{16} \times ((2 \times Px) + 1) \times ((2 \times Py) + 1) \times F \qquad (6)$$

According to equation (6), the time needed to calculate one search position for an HDTV video sequence (1280x720, 60 fps) and search range 16x16 is 4,3 ns, which is more than the time needed (4.15 ns) for the proposed architecture to choose one search position.

7. Conclusions

In this paper we presented a new mode selection algorithm for both Intra and Inter prediction modes in H.264 along with logic transformations for its hardware implementation. The proposed replacement of the SAD algorithm with a simple comparison was integrated with the H.264 JM11.0 codec for performance evaluation and was implemented using standard circuit synthesis tools.

The experimental results of the hardware implementation of the proposed algorithm indicate that the design achieves the real-time encoding requirements for processing an HDTV video sequence with search range 16x16, with a power consumption of just 95mW. Therefore, the new algorithm, by reducing the complexity of the computation, can achieve significantly faster execution time and significant reduction in power consumption without having any perceivable impact on the file size and the quality of the final video sequence.

8. References

[1] ITU-T Rec. H.264 / ISO/IEC 11496-10, *Advanced Video Coding*, Final Committee Draft, Document JVT-J010d1, December 2003.

[2] Wiegand, T., Sullivan, G., Bjontegaard, G., and Luthra, A. , *Overview of the H.264/AVC video coding standard*, IEEE Transactions on Circuits and Systems For Video Technology, vol. 13, no. 7, pp.560-576, July 2003.

[3] Richardson, I., *H.264 and MPEG-4 Video Compression – Video Coding for Next-generation Multimedia*, John Wiley & Sons, 2003.

[4] Toivonen, T., Heikkila, J., *A new rate – minimizing matching criterion and a fast algorithm for block motion estimation*, International Conference on Image Processing (ICIP 2003).

[5] Mano, M., *Digital Design*, 3rd edition.

[6] Joint Video Team (JVT) Reference Software JM11.0, at http://iphome.hhi.de/suehring/tml/download/

[7] VQEG – Video Quality Experts Group http://www.its.bldrdoc.gov/vqeg/

[8] Deng, L., Gao, W., Hu, Z., and Ji, Z., *An Efficient Hardware Implementation for Motion Estimation of AVC Standard,* IEEE Transactions on Consumer Electronics, vol.51, no.4, pp.1360-1366, November 2005.

[9] Ou, C., Le, C., and Hwang, W., *An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation,* IEEE Transactions on Consumer Electronics, vol.51, no.4, pp.1291-1299, November 2005.

[10] Yap, S., McCanny, J., *A VLSI Architecture for Variable Block Size Video Motion Estimation,* IEEE Transactions on Circuits and Systems – II: Express Briefs, vol. 51, no. 7,pp. 384-389, July 2004.

[11] Sayed, M., Amer, I., and Badawy, W., *Towards an H.264/AVC Full Encoder on Chip: An Efficient Real-Time VBSME ASIC Chip,* International Symposium on Circuits and Systems (ISCAS 2006).

[12] Song, Y., Liu, Z., Goto, S., and Ikenaga, T., *Scalable VLSI Architecture for Variable Block Size Integer Motion Estimation in H.264/AVC,* IEICE Trans. Fundamentals, vol. E89-A, no. 4, pp. 979-988, April 2006.

[13] 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: Regular Papers, vol. 53, no. 2, pp. 578-593, February 2006.

[14] Wang, C., Lee, P., Wu, C., and Wu, H., *High Fan-In Dynamic CMOS Comparators with Low Transistor Count*, IEEE Transactions on Circuits and Systems – I: Fundamental theory and applications, vol. 50, no. 9, pp. 1216-1220, September 2003.