ISSN: 0976-9102(ONLINE)
DOI: 10.21917/ijivp.2014.0120

A PERFORMANCE COMPARISON ON DIFFERENTIAL PREDICTION BASED ALGORITHMS IN H.264/AVC

D. Malarvizhi[1], R.Vanitha[2] and A. Nandha Kumar[3]
Department of Electrical and Electronics Engineering, Dr. Mahalingam College of Engineering and Technology, India
E-mail: [1]malarvizhibe@gmail.com, [2]vanithark@gmail.com, [3]nandhu.udt@gmail.com

Abstract
H.264 is an Advanced Video Coding standard that gives better coding efficiency than the previous standards. In H.264, video compression is carried out in many ways such as Interframe prediction and Intraframe prediction. Intraframe prediction is carried out by the process of motion estimation by calculating motion vectors. Intraframe prediction reduces spatial redundancy (ie) similarity between pixels and Interframe prediction reduces the temporal redundancy (ie) change in video content from one frame to the next frame. In this paper we compare these two compression schemes of H.264 in terms of compression ratio, PSNR and memory bandwidth. Finally we conclude that intra prediction method provides better video quality with average compression ratio 1.034 and the percentage of memory saving is 96.26% but the average PSNR value is less compared to motion estimation.

Keywords:
H.264, Intra Frame Prediction, Inter Frame Prediction, Motion Estimation

1. INTRODUCTION

A video coder encodes video sequence into a compressed form. H.264/AVC is newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Pictures Experts Group. The main goal of this standard is to give enhanced compression performance. The H.264/AVC standard was first introduced in 2003. The standard defines the video bitstream and decoding method, allowing design flexibility for encoding process. Compared to the other standard, H.264/AVC contains a number of new features, which not only offers lower bit rate and more efficient compression, but also provide more flexibility for application to a wide area of network environments [1].

The Fig.1 shows the simplified block diagram of the H.264 encoder. The frames of video sequence are divided into number of macroblocks (MB). For every macroblock, a prediction is created using previously coded data. The encoder may select between intra- and inter-prediction for block-shaped regions of each picture.

There exist two types of redundancy: spatial and temporal. The proposed algorithms are used to reduce these redundancies.

H.264/AVC intra-frame encoding compresses video sequence on the basis of exploiting spatial redundancy. It encodes every macroblock using previously encoded and then decoded neighboring macroblocks in the same frame and encodes every frame individually. An intra (I) macroblock is coded without referring to any data outside the current slice. I macroblocks may occur in any slice type. Every macroblock in an I slice is an I macroblock. I macroblocks are coded using intra prediction, i.e. prediction from previously-coded data in the same slice. For a typical block of samples, there is a relatively high correlation between samples in the block and samples that are immediately adjacent to the block. Intra prediction therefore uses samples from adjacent, previously coded blocks to predict the values in the current block [2].

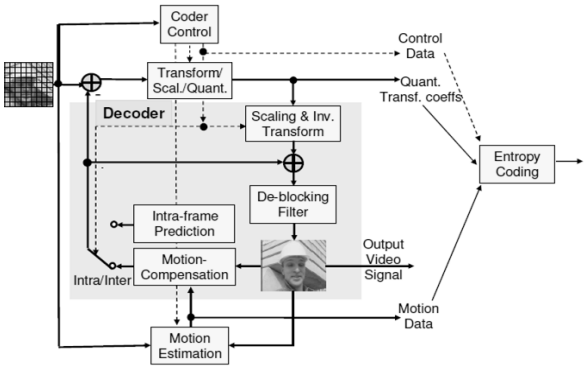

Fig.1. H.264 encoder

Interprediction is carried out by Motion Estimation (ME) process. It identifies temporal redundancy between neighbouring frames. We call the frame currently being processed the currentframe and the neighbouring one the reference frame. We try to find from the reference frame a reference macro block that is very similar to the currentmacro block of the current frame. This process is called motion estimation.

A motionestimator compares the current macro block with candidate macro blocks within a search window in the reference frame. After finding the best-matched candidate macro block, only the displacement and the error need to be encoded and stored/transmitted. The displacement from the location of the current macro block to that of the best candidate block is called motion vector (MV). A MV obtained from motion estimation is adequate for retrieving a block from the reference frame.

This paper proposes the comparative analysis of full search algorithm and intra prediction of H.264. The rest of this paper is organized as follows. Section 2 introduces intra prediction algorithm. Section 3 introduces motion estimation algorithm. Section 4 presents results and discussion of the proposed algorithms for standard video sequence. Finally we drawn some conclusions in section 5.

2. INTRA PREDICTION

In this algorithm prediction of the current macroblock is based on previously coded data from the current frame. Initially one macroblock is processed without prediction. The remaining macroblocks are encoded based on the previously encoded macroblock. Fig.2 shows the block diagram of the proposed algorithm. The process is as follows:

Step 1: The first macroblock of one frame is given to the transform unit.

Step 2: The transformed coefficients are then given to the quantization unit.

Step 3: Finally the quantized coefficients are encoded.

Step 4: The reconstruction process is carried out by inverse quantization, inverse transform. This decoded data is subtracted from the next macroblock and this residual value is given to the transform unit.

Step 5: Repeat the process from step2 until the entire frame is encoded.

2.1 DISCRETE COSINE TRANSFORM(DCT)

A block of residual samples is transformed using a 4 × 4 or 8 × 8 integer transform, an approximate form of the Discrete Cosine Transform (DCT). The transform outputs a set of coefficients, each of which is a weighting value for a standard basis pattern. When combined, the weighted basis patterns re-create the block of residual samples [3].

The equation used to find the DCT is

$$c(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{\pi(2x+1)}{2N}\right] \quad (1)$$

where, u = 0, 1, 2, ...N-1

$f(x)$ – Input coefficients

$c(u)$ – Transformed coefficients

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for\ u = 0 \\ \sqrt{\frac{2}{N}} & for\ u \neq 0 \end{cases} \quad (2)$$

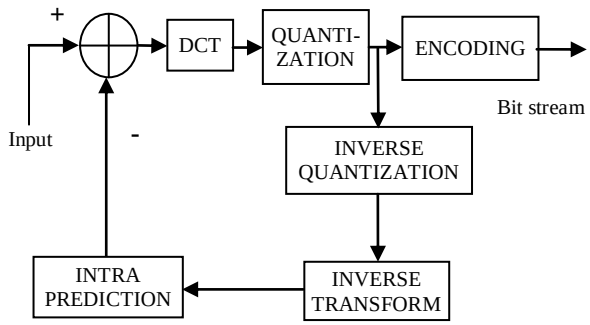

Fig.2. Block diagram of Intra prediction

2.2 QUANTIZATION

The output of the transform, a block of transform coefficients, is quantized, i.e. each coefficient is divided by an integer value. Quantization reduces the precision of the transform coefficients according to a quantization parameter (QP).

Setting QP to a high value resulting in high compression at the expense of poor decoded image quality. Setting QP to a low value resulting in better image quality at the decoder but also in lower compression. A larger quantization step size tends to produce a larger difference between original and reconstructed blocks [7]. The H.264/AVC standard proposes a scalar quantizer. The basic quantization operation is defined in

$$Z_{ij} = round(Y_{ij}/Q_{step}) \quad (3)$$

where, Y_{ij} is the output of the transform unit. Q_{step} is a quantization step derived according to the quantization parameter (QP). There are 52 Q_{step} values as shown in Table.1. Q_{step} is implemented using a multiplication factor and a shift-right operation to avoid division operation. A quantized value is obtained through a multiplication with quantization coefficient (Quant_coef), an addition of constant (Qp_const), and shift as shown in Fig.3.

Table.1. Quantization Step Sizes

QP	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Q_{step}	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375
QP	**8**	**9**	**....**	**18**	**...**	**24**	**...**	**30**
Q_{step}	1.625	1.75	....	5	...	10	...	20

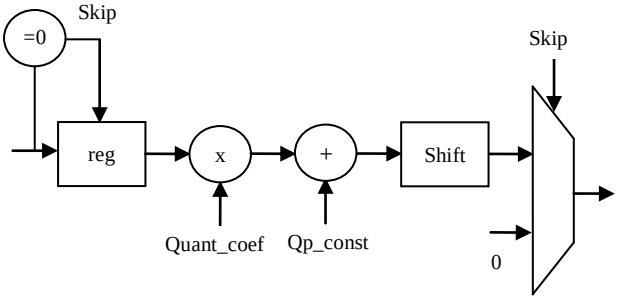

Fig.3. Quantization unit

2.3 ENCODING

The video coding process produces a number of values that must be encoded to form the compressed bitstream. The quantized coefficients are given to the encoder block. In this step binary value is assigned for each quantized coefficients individually.

2.4 INVERSE QUANTIZATION AND INVERSE TRANSFORM

In the process of inverse quantization, the quantized transform coefficients are re-scaled. Each coefficient is multiplied by an integer value to restore its original scale. To recover the data, a multiplication followed by rounding and shift is performed [5]. An inverse transform combines the standard basis patterns, weighted by the re-scaled coefficients, to re-create each block of residual data.

3. FULL SEARCH ALGORITHM FOR MOTION ESTIMATION

There are many kinds of algorithm for block based motion estimation. The most accurate stratergy is the Full search (FS) algorithm. By exhaustively comparing all reference blocks in the search window, FS gives the most accurate motion vector which causes minimum sum of absolute differences (SAD). The advantage of full search is that we can find the absolute optimal solution.

Previous work [8], [9] has proposed several efficient schemes for the fixed block size motion estimation (ME). However, those methods could not perfectly conform to the Variable Block Size Motion Estimation (VBSME) feature proposed in H.264/AVC. The VBSME involves 41 combinations of the MVs for blocks {16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8, and 4 × 4} in each macro

block (MB). Conventional architectures [10]–[12] are no longer fully applicable to H.264/AVC encoders since the VBSME increases both the coding time and the computational burden of hardware design.

Based on this, we will propose our new motion estimation algorithm using Early Termination stratergy. To test its validation and its efficiency, we used standard test video sequence.

3.1 MOTION SEARCH WITH EARLY TERMINATION

In this method [13], the large block SADs are generated by means of adding another 4 × 4 block SAD which has a minimum value and become the candidate position. Based on this observation, we proposed an improved ME algorithm. In this algorithm, the SAD of 4 × 4 blocks is first calculated and each candidate position is recorded when its SAD is smaller than the minimum matching error (MME), i.e., stored into a buffer. The first candidate position of the SAD is always assigned to be the first MME. The same procedure is performed for the other 4 × 4 blocks. The motion search for the 4 × 4 block is stopped when the buffer capacity reaches the value initially assigned. The buffer large block capacity is also set when performing the large block size which is obtained by the combination of the sixteen 4 × 4 blocks. Therefore, it only needs to search the candidate positions that are stored in the buffer for the large block ME.

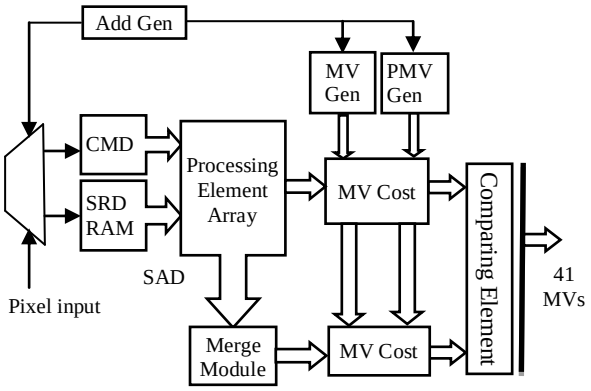

Fig.4. Block diagram of VBSME processor

SAD can be calculated by using the formula given below.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C(i, j) - R(i.j) \right| \qquad (4)$$

where, $C(i,j)$ = $(i,j)^{th}$ location of current frame

$R(i,j)$ = $(i,j)^{th}$ location of reference frame

3.2 VBSME PROCESSOR

The standard function of VBSME[13] includes SAD and MV cost calculation whereas previous works concerned only the SAD calculations without considering the impact of the MV cost function. This kind of simplification may cause some distortion when deciding the best block size. Thus, in this work we have demonstrated the effectiveness of the full search algorithm and integrated it into hardware architecture design for VBSME. Consequently, this architecture is more complete in comparison with previous works [14], [15]. The VBSME consists the following basic elements: current MB data (CMD), search region data (SRD) RAM, address generator (Add Gen), motion vector generator (MV Gen), predicted motion vector generator (PMV Gen), processing element array (PEA) for computing 4 × 4 SAD, comparing element (CE), and a merging tree to sum up the 4 × 4 block for larger block types. Fig.4 shows the functional diagram of the proposed VBSME inter processor. The proposed architecture can compute the optimal MV more efficiently than the conventional methods.

3.3 DATA FLOW OF THE PEA

The basic architecture of a PEA [13] is used to compute the SAD of a 4 × 4 block. A PEA is formed by 16 processing elements (PE) and requires a total of 16 PEAs. Two hundred and fifty-six PEs are needed to perform the entire computation. Fig.5 shows the architecture of a PEA, in which SAD (i, j) represents the SAD value in the (i, j) position of the 4 × 4 block. Each PEA is assigned for a specific block motion search. In the PEA, the CMD is shifted in the horizontal direction (1, 0) and the SRD in the diagonal direction (−1, 1). The ADD module is used to sum up the difference between the CMD and the SRD. SAD are commonly employed in hardware implementation to determine the match between two blocks due to their simplicity.

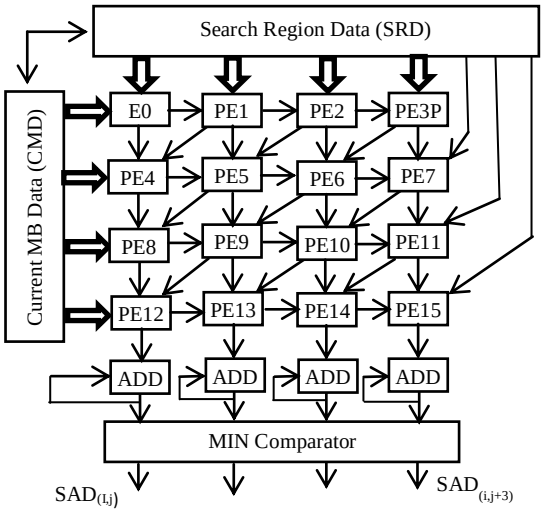

Fig.5. Architecture of PEA unit

3.4 MERGE MODULE

After the outputs of the PEAs are arranged, the 16 SADs of various 4 × 4 blocks will be ready at the input port of the merge module. The merge scheme finds the large block SAD by adding the SAD of 4 × 4 blocks. In order to avoid the long critical path, a carry look-ahead adder is used to replace the 15-bit adder when generating the SAD of a large block.

4. RESULTS AND DISCUSSION

In order to evaluate the performance of video compression coding, it is necessary to calculate the peak signal to noise ratio and compression ratio [4]. Most video compression systems are designed to minimize the mean square error (MSE) between two video sequences Ψ_1 and Ψ_2, which is defined as,

$$MSE = \sigma_e^2 = \frac{1}{N} \sum_t \sum_{x,y} \left[\Psi_1(x, y, t) - \Psi_2(x, y, t) \right]^2 \qquad (5)$$

where, *N* is the total number of frames in either video sequences.

The Peak Signal-to-Noise Ratio metric is defined as:

$$PSNR = 10\log_{10}\frac{\left(2^n - 1\right)^2}{MSE} \quad (6)$$

where, $(2^n–1)^2$ is the square of the highest-possible signal value in the image, n is the number of bits per image sample.

The two proposed algorithms are simulated for Akiyo video sequence and the PSNR value and compression ratio for first five frames are compared in Table.2.

Table.2. Comparison of PSNR value and Compression ratio

Frames	PSNR(dB)		Compression ratio	
	Intra Prediction (IP)	Motion Estimation (ME)	IP	ME
Frame1	41.653	47.867	1.007	1.001
Frame2	40.427	48.960	1.031	1
Frame3	39.824	49.304	1.037	0.999
Frame4	39.248	49.559	1.044	1.002
Frame5	39.034	49.740	1.051	1
Average	40.037	49.086	1.034	1

Table.3. Bandwidth comparison

Algorithm	Input video size (Bytes)	Output video size (Bytes)	% Saving
Intra Prediction	68207616	2545782	96.26 %
Motion Estimation	68207616	2676680	96 %

The algorithms are also compared in terms of bandwidth as shown in Table.3.

5. CONCLUSION

This paper presented the comparative analysis of Full Search algorithm (FS) and Intra prediction method for video compression. Video quality was measured and compared by using both the methods in terms of PSNR, compression ratio and an effective bandwidth. Finally the simulation results shows that intra prediction produces better result than motion estimation with compression ratio of 1.034 and the percentage of memory saving is 96.26% but the PSNR value is less compared to motion estimation.

REFERENCES

[1] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard", *IEEE Transactions on Circuits Systems and Video Technology*, Vol. 17, No. 9, pp. 1103-1120, 2007.

[2] Tung-Chien Chen, et al., "Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 6, pp. 673-688, 2006.

[3] Yu-Kun Lin, Chun-Wei Ku, De-Wei Li and Tian-Sheuan Chang, "A 140-MHz 94 K gates HD1080p 30-Frames/s intra-only profile H.264 encoder", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 19, No. 3, pp. 432-436, 2009.

[4] Huang-Chih Kuo, Li-Cian Wu, Huang Hao Ting, Sheng-Tsung Hsu and Youn-Long Lin, "A low-power high-performance H.264/AVC intra-frame encoder for 1080pHD video", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19, No. 6, pp. 925-938, 2011.

[5] G-L Li, T.-Y Chen, M.-W. Shen, M.-H. Wen and T.-S. Chang, "135-MHz 258-K Gates VLSI Design for All-Intra H.264/AVC Scalable Video Encoder", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 21, No. 4, pp. 636-647, 2013.

[6] Y.W. Huang, B.Y. Hsieh, T.C. Chen and L.G. Chen, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 3, pp. 378-401, 2005.

[7] C.W. Ku, C.C. Cheng, G.S. Yu, M.C. Tsai and T.S. Chang, "A high-definition H.264/AVC intra-frame codec IP for digital video and still camera applications", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 8, pp. 917-928, 2006.

[8] J.C. Tuan, T.S. Chang and C.W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 1, pp. 61-72, 2002.

[9] J.F. Shen, T.C. Wang and L.G. Chen, "A novel low-power full search block-matching motion-estimation design for H.263+", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 7, pp. 890-897, 2001.

[10] S.Y. Yap and J.V. McCanny, "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, 2004.

[11] L. Deng, W. Gao, M.Z. Hu and Z.Z. Ji, "An efficient hardware implementation for motion estimation of AVC standard", *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 4, pp. 1360-1366, 2005.

[12] C.Y. Chen, et al., "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.

[13] An-cho Tsai, K. Bharanitharan, Jhing-FaWangg and Kuan-I Lee, "Effective Search Point Algorithm and its VLSI Design for HDTV H.264/AVC Variable Block Size Motion Estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 22, No. 7, pp. 981-988, 2012.

[14] M. Sayed, W. Badawy and G. Jullien, "Towards an H.264/AVC HW/SW integrated solution: An efficient VBSME architecture", *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 55, No. 9, pp. 912-916, 2008.

[15] C. Wei, H. Hui, T. Jiarong, L. Jinmei and M. Hao, "A high-performance reconfigurable VLSI architecture for VBSME in H.264", *IEEE Transactions on Consumer Electronics*, Vol. 54, No. 3, pp. 1338-1345, 2008.