AN EFFICIENT FRACTIONAL MOTION ESTIMATION ARCHITECTURE FOR AVS REAL-TIME FULL HD VIDEO ENCODER

Fei Wang, Yuan Li, Huizhu Jia, Xiaodong Xie, Wen Gao
National Engineering Laboratory for Video Technology
Peking University
Beijing, 100871, P. R. China

***Abstract* — Fractional motion estimation (FME) as a complement to integer motion estimation (IME) conducts higher compression rate in video coding. Based on multiple reference frames and all modes of variable-block-size (VBS) motion search, the common FME algorithm and architecture generally causes high computational complexity. It consumes plenty of time and hardware resources to process the real-time (30fps) high definition (HD) video compression. In this paper, we propose an efficient FME algorithm and architecture that significantly reduce the number of the candidate sub-pixels by predicting quarter motion vector and cost less hardware by saltatory interpolation. Compared with the conventional FME architecture, the proposed method reduces resources significantly and accelerates the processing speed. 75% circuit area and 68% computational complexity are saved with only 0.1dB drop in performance and similar bit rate saving on average. Experiments of various sequences show that HD video (1920x1080) can be processed in real-time at frequency of 100MHz and it is verified in an AVS HD encoder on a Xilinx Virtex-6 FPGA prototype system.**

Keywords-FME; HD; Real-time; AVS

I. INTRODUCTION

The advanced audio video coding standard (AVS), one of the three international second generation source coding technologies, is independently developed by the Audio Video Coding Standard Work Group of china. AVS produces comparative performance as H.264/AVC with reduced implementation cost. Its block based hybrid framework is similar with the previous MPEG-x and H.26x series of standard. In the video compression coding, Motion Estimation (ME) is crucial to remove the temporal redundancy and achieve a high compression ratio and it is the most time and hardware consuming part. The common ME architecture consists of IME and FME: integer motion search over a large area and sub-pixel search around the best selected integer pixel. FME can improve 1~2dB Peak Signal-to-Noise Ratio (PSNR) by using quarter precision motion vector (MV) after integer motion search.

Although FME reinforce the compression in an efficient way, it still takes considerable computing time and hardware resources. As many algorithms of IME [1][2][3] have been proposed in which only three to five integer pixels are used as candidates. The computation of typical 16-point fractional motion search strategy used in conventional encoder system thus becomes comparatively huge. Also the multiple reference frames and VBS modes need large circuit area demanding parallel processing to meet real-time HD encoding requirements.

The classic FME motion full search (FS) contains two steps: first, interpolate the half pixels around the integer candidate point, find the optimal half pixel; second, execute the same operation to the half pixel candidate to get the best quarter pixel. This can be shown in Figure 1. There are eight half precision sub-pixels (*a*, *b*, *c*, *d*, *e, f*, *g*, *h*) need to be searched and the same for the quarter precision sub-pixels (1, 2, 3, 4, 5, 6, 7, 8). And that need to be done for each reference and every block size partition. So some previous researches focus on optimizing algorithm to reduce the search points, Jeong Jechang's work was at the model using the parameters derived from the neighboring integer pixels' sum of absolute difference (SAD) to estimate the sub-pixels SAD [9]. And in reference [10], a novel algorithm was proposed, which performs a "rough" sub-pixel search before the partition selection, and performs a "precise" sub-search for the best partition.

The inter frame prediction technology of AVS is composed of four modes (forward prediction, backward prediction, direct prediction, bi-prediction) and four

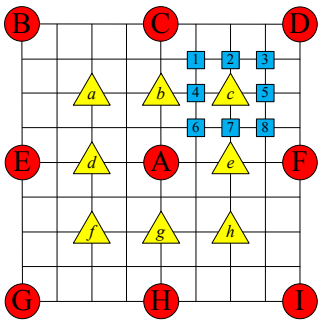


Figure 1. FME Full Search

sub-block partition modes (16x16, 16x8, 8x16, 8x8). And the AVS's bi-prediction has a special technique: Symmetric mode. The particular mode calculates the backward quarter precision MV instead of searching it according to the known forward MV [4]. So both of the direct prediction and the symmetric prediction in AVS can give out the quarter accurate MVs directly, this two modes' quarter pixels can be interpolated directly by using integer pixels in order to save the cost of half-pixel interpolation. Furthermore if we can predict the quarter precision MVs of the other two modes, the integer-to-quarter compute units could be shared efficiently. Based on this optimization algorithm we propose a novel compact framework that can achieve significant improvement both on processing speed and circuit resources.

The following paper is organized as follows. In Section II we discuss the symmetric motion estimate strategy, the equation of integer-to-quarter and the algorithm to get quarter MVs in advance. In Section III the proposed architecture will be introduced. Section IV shows the experiments' results. The final section concludes this paper.

II. ALGORITHM ANALYSIS AND OPTIMIZATION

1. Symmetric Mode

In B frame all the four partition modes of one macro block (16x16, 16x8, 8x16, 8x8) are estimated in three directions: forward, backward and bi-direction. In conventional bi-prediction mode, both of the forward and backward MVs are coded for each block. In AVS, the unique Symmetric mode with single coded forward MV can efficiently save the bits coding MVs as a new bi-prediction mode. As shown in Figure 2, the corresponding backward MV is derived from forward MV depending on the temporal distance between the forward and backward reference frames and the current B frame [4]:

$$\mathrm{MV_B} = \frac{\mathrm{DBf}}{\mathrm{DBb}} \times \mathrm{MV_F} \qquad (1)$$

Where DBf denotes the distance between the current B frame and the forward reference frame, DBb demotes the distance between current B frame and the backward reference frame. If not considering the computation cost, every sub-pixel could be searched to get the best pair of symmetric MVs.

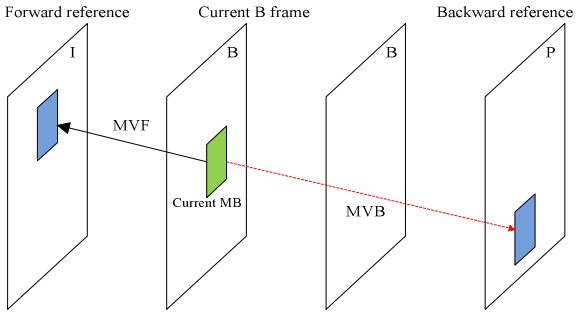


Figure 2. Symmetric Prediction

In the AVS reference software RM52K, a forward MV is derived from the neighboring blocks' MVs, and all the surrounding sub-pixels and their corresponding backwards sub-pixels are estimated. The amount of calculation equals to that of forward prediction combined with backward prediction.

We optimize the Symmetric mode by reusing the forward integer-pixel MVs that the IME provides, so the expected circuit of Symmetric mode's forward FME is saved. And the derived symmetric backward MVs are of the quarter accuracy. The expected comparison computation is saved too. In all, for the proposed Symmetric mode's algorithm we only need to interpolate the desired sub-pixels directly by integer pixels. And we will discuss the integer-to-quarter interpolation in section 3.

2. Optimization of Sub-pixel Estimation

The classic quarter estimation is based on the half pixel search's result. Eight quarter size pixels around the best half pixel (such as point 1, 2, 3, 4, 5, 6, 7, 8) in Figure 1 are interpolated and their costs compared. In inter frame prediction it is often considered that there are three types of half pixels and twelve types of quarter pixels. As shown in Figure 3, point 1, 2, 3 are half pixels and point a, b, c, d, e, f, g, h, i, j, k, l are quarter pixels. Because the quarter candidates are different depending on different position of best half pixels, all the twelve types of quarter pixels' processing units are required. As a result the quarter part of FME is much more complex than the half part.

Now we suppose that the forward prediction's quarter MV and backward prediction's quarter MV can be pre-obtained, the quarter precision search of FME will be significant simplified without 8 interpolation and comparison. The integer-to-quarter interpolation units can be shared with the Symmetric mode which also saves the storage of a large quantity of half pixels used as the original pixels for quarter interpolation.

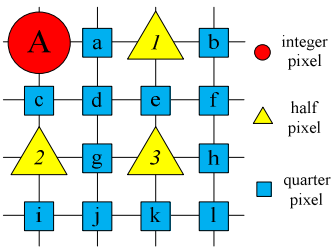


Figure 3. Sub-pixels

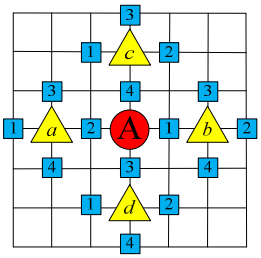


Figure 4. Cross Search Pattern

Now in order to find an effective way to get the quarter MV in advance, we turn to the monotonic error surface. As the sub-pixels are generated by the integer pixels, the correlation of the fractional search window is much higher than the integer search window. So the error surface is monotonic most cases of fractional pixels. Former studies of the error surface come up with three hints for us:

a) The minimum error point can be found along the direction of the block error from the highest to the lowest point. As the error direction is known, only the points on the search path need to be calculated. This method is called Directional Asymmetric Search (DAS) [6].

b) The minimum fractional error point can be assumed to locate between the minimum and the second minimum error point [7]. That leads to a smaller number of candidates too.

c) Based on the monotonic error surface, the statistical data show that the rectangular (the horizontal and vertical) sub-pixels and the search center almost cover 95% of the full search destinations [5]. So at the fractional level we just count the horizontal and vertical sub-pixels as candidates that the candidate points are reduced to 50%.

In the light of above a), b) and c) premises, we propose a novel fast FME algorithm with two steps:

step 1) Interpolate the four rectangular half pixels, compare their cost and find out the maximum and minimum error half points. The cross search pattern is shown in Figure 4.

step 2) Deduce the quarter point using the upper conclusion, and also only consider the four rectangular quarter points. There are four cases of the search pattern (Figure 5-8) and different pattern leads to different quarter candidate.

case 1) The minimum point falls on the center of the cross so the maximum point is at the four ends. We assume the desired quarter point is the further quarter pixel, on the extension line of max and min point, next to the center integer pixel, as the red cube shown in Figure 5.

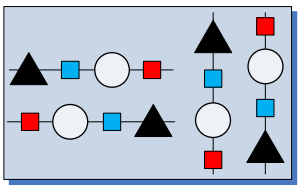


Figure 5. Case 1

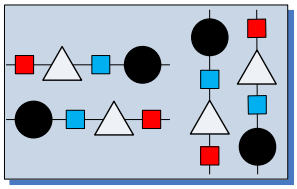
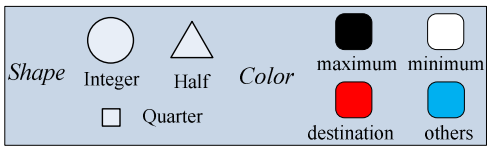


Figure 6. Case 2



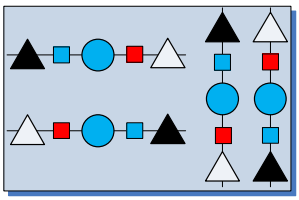


Figure 7. Case 3

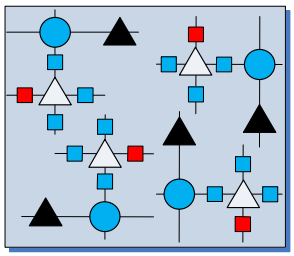


Figure 8. Case 4

case 2) The maximum point falls on the center of the cross so the minimum point is at the four ends. We assume the desired quarter point is the further quarter pixel, on the extension line of max and min point, beside the end half pixel, as the red cube shown in Figure 6.

case 3) The maximum and minimum points are on the four ends and fall on a vertical or horizontal line. So the second minimum point must be one of the other three points on their perpendicular bisector. We assume the desired quarter point is the quarter pixel between the minimum point and the center point according to a) and b), as the red cube shown in Figure 7.

case 4) The maximum and minimum points are on the four ends and not in a rectangular line. We assume the desired quarter point is the quarter pixel which is closer to the other three points of the two further points to the maximum point according to a) and b), as the red cube shown in Figure 8.

This algorithm provides the forward mode and backward mode's quarter accurate MVs without quarter scale motion search. On the other side the direct mode and the Symmetric mode also provide the quarter MV directly by calculating. So far all we need are integer-to-quarter interpolation units.

3. *Saltatory interpolation*

Normally the quarter pixels are dependent on the outcome of half interpolation. In order to take advantage of the vested quarter accurate MV, we propose to change

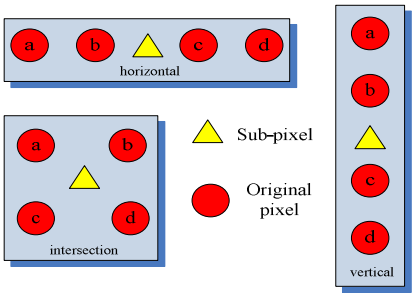


Figure 9. Sub-pixel Interpolation

the integer-half-quarter model to integer-to-quarter as a saltatory interpolation. The special interpolation units are for the Direct mode and Symmetric mode of AVS, because these two particular modes provide the quarter accuracy MVs that the interpolation units can use to achieve the quarter pixels without estimation. And the units also can be reused by the forward and backward reference modes which give out the quarter MVs using the algorithm proposed in chapter 2. In AVS the three types of sub-pixels are generated by 4-tap interpolation filter as shown in Figure 9.

Three diagrams can be unified as:

$$Sp = (W_1*a + W_2*b + W_3*c + W_4*d + W_5)/W_6 \quad (2)$$

The half and quarter interpolation can both be expressed as the above expression. The W* denotes the interpolation weight and the subscript denote the integer pixel positions. The half pixels take integer pixels as original and the quarters take half pixels as original. Replace the half pixels in the quarter interpolation formula with their integer expressions, the quarter pixels' integer expressions are derived. They are expressed as below:

$$Quarp = (W_1*I_1 + W_2*I_2 + \cdots + W_m*I_m)/W_n \quad (3)$$

In expression (3), I* is for the integer pixel and W* is for the weight.

We classify the quarter pixels into three types referring to Figure 3. Point a, b, c and i are the same type on the direct line with integer points. Point e, g, h and k are the same type that in the direct line only has half pixels. And d, f, j and l are the intersection ones. These three types instantiate the expression (3) with distinct integer pixels and weights:

$$Q_{a,b,c,i} = (W_{A1}*I_{A1} + \cdots + W_{A5}*I_{A5} + W_{A6})/W_{A7} \quad (4)$$

$$Q_{e,g,h,k} = (W_{B1}*I_{B1} + \cdots + W_{B20}*I_{B20} + W_{B21})/W_{B22} \quad (5)$$

$$Q_{d,f,j,l} = (W_{C1}*I_{C1} + \cdots + W_{C16}*I_{C16} + W_{C16})/W_{C17} \quad (6)$$

So we only use three computation units in above equations to implement quarter interpolation which can cover all quarter positions. In some cases the Symmetric mode's or Direct mode's MVs point to the half or integer candidates, we can also obtain the half or integer pixels by giving the W* different values. The Saltatory Interpolation

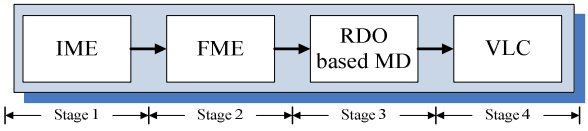


Figure 10. Four stages pipeline AVS encoder architecture

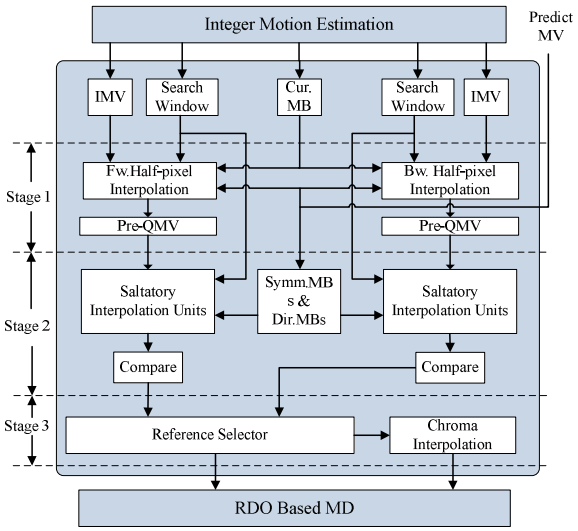


Figure 11. Proposed FME architecture

save the median half pixels' storage and reduce the computation units from 12 to 3.

III. ALGORITHM AND ARCHITECTURE

1. General Architecture

Our FME architecture can be adopted in the typical AVS encoder architecture include four stages that include IME, FME, Rate-Distorition-Optimization (RDO) based Mode Decision and Variable Length Coding (VLC) as shown in Figure 10 [8].

FME is at the second stage of the pipeline, whose inputs include search window, current MB and integer MVs from IME and outputs contain best matched reference pixels and quarter accuracy MVs to be transferred to MD. And the proposed FME architecture is shown in Figure 11. To meet the requested processing speed we adopt the three stages pipeline architecture.

The first stage contains the half-pixel interpolation units (IU) and the quarter MV prediction units. The forward section performs the FME with the forward reference frame for B frame and the first frame for the P frame. The backward section performs the FME with the backward reference frame for B frame and the second frame for the P frame. Each direction has three IUs of three different types of half pixels. And in the Pre-QMV unit the quarter MVs are predicted by the estimation of the cost of half candidates viewed as the optimal quarter MVs.

The second stage has two groups of Saltatory Interpolation units (SIU), the unit that processes the Symmetric mode

and Direct mode's MVs (SDMV) and the cost compare units. The SIUs get MVs both from half ME and the SDMV. There are three computation units of three different types of quarter pixels in each SIU. The forward, backward, symmetric and direct mode's quarter ME share the two groups of SIUs. Since in stage 1 we get the best error match cost of half pixel, we compared the quarter cost with the best half cost to improve the accuracy. The winner quarter pixels are provided to stage 3 to elect the better reference frame.

The third stage performs the reference frame selection and chroma interpolation. Chroma IU contains half IU and quarter IU. They use 4x4 block level IUs to generate chrominance sub-pixels. Both of the best matched luminance sub-pixels, chrominance sub-pixels and the quarter accuracy MVs are sent to MD in this stage.

2. *Pipeline Workflow*

In full mode inter frame prediction of AVS, the Forward, Backward and Symmetric modes all have four partitions (16x16, 16x8, 8x16, 8x8) to process. The Direct mode has one partition (8x8). We take an 8x8 size sub-block as a computing unit. The Forward and Backward 32 units go through the entire pipeline. The Symmetric and Direct 20 units are just performed in stage 2 and stage 3.

As shown in Figure 12, the workflow of forward and backward pipeline are much alike, the difference is that they process different partitions of Symmetric mode. The Symmetric 16 blocks and the Direct 4 blocks are placed at the beginning of second and third stage since their MVs are available ahead, one derived from IMVs and the others derived from neighboring blocks' MVs, and luma and chroma can be performed at the same time. Two reference frames of the Direct mode are performed separately in corresponding direction.

As shown in the red units, The 16 Symmetric units' have their own particularities. The Symmetric partitions only have backward quarter pixels to be estimated, so the 16 units of four partitions are placed evenly in the procession of stage 2 and 3 to equalize the Forward and Backward time consumption.

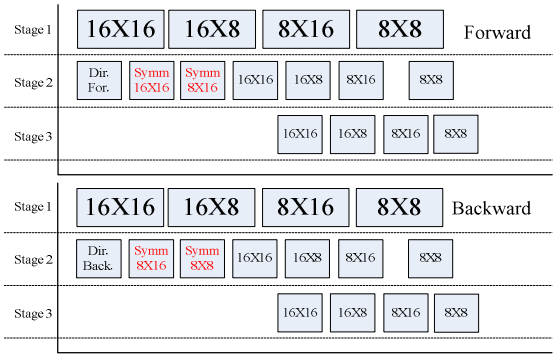


Figure 12. Pipeline Workflow

In stage 1, one partition's process time is 80 cycles in average. In stage 2 and 3 the time is 44 cycles in average. Because the four 8x8 units in partition 16x8, 8x16 and 8x8 have different combination mode, the pipeline can run forward without waiting for the final completeness of all the four 8x8 units of each partition. Totally the average time of the whole FME flow of one macro block is 400 cycles. So this FME architecture can perform real-time HD video compression at the frequency of 100MHz.

IV. EXPERIMENTAL RESULT

1. *Algorithm Analysis*

We analyzed the new algorithm by modifying the reference software of AVS (RM52K). The MV search range is $[\pm 128, \pm 96]$ and RDO is on. IBBP GOP structure is used and the tested QP is from 26 to 32. The original FME algorithm is adopted as reference. The test YUV sequences are Foreman, Akiyo, City, Blue sky and Crowd run. As shown in Fig. 13&14, the proposed algorithm has about 0.12dB PSNR drop compared with Full search for HD video@30f. From Table 1 and 2 we can see that compared with the full search, the proposed algorithm's average PSNR drop is 0.12dB and the rate increase is 2.6%, but with a reduced computational complexity by 68%.

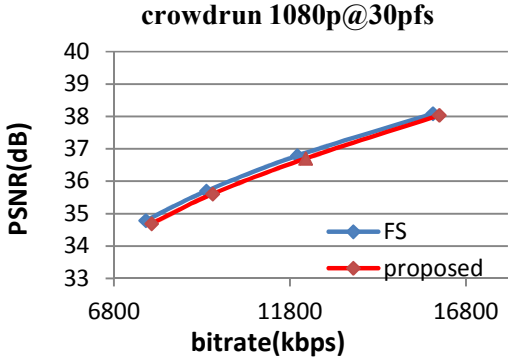


Figure 13. Performance of different FME algorithm (Crowd run)

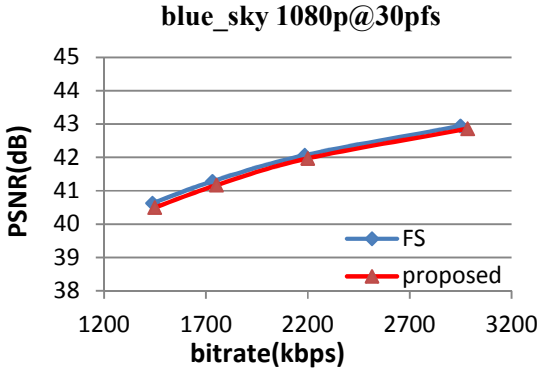


Figure 14. Performance of different FME algorithm (Blue_sky)

Table 1.Performance Comparison

	Foreman cif				City 720p			
	26	***28***	***30***	***32***	***26***	***28***	***30***	***32***
Full	39.82	38.61	37.60	36.80	41.01	40.11	41.11	39.39
New	39.74	38.47	37.48	36.69	41.88	40.01	39.96	39.24
drop	0.08	0.14	0.12	0.11	0.12	0.10	0.15	0.15
	Blue sky 1080p				**Crowd run 1080p**			
	26	***28***	***30***	***32***	***26***	***28***	***30***	***32***
Full	42.84	42.05	41.28	40.61	38.09	36.79	35.70	34.79
New	42.84	41.94	41.14	40.47	38.95	36.67	35.59	34.69
drop	0.10	0.11	0.14	0.14	0.14	0.12	0.11	0.10

Table 2.Rate-Distortion Comparison

	Foreman	**City**	**Blue sky**	**Crowd run**
Δrate (%)	2.9	2.5	2.2	3.1

Table 3.Circuit Units Comparison

	Half IUs	**Quar. IUs**	**Half-pel Storage units**	**Quar-pel Storage units**
Full	18	72	48	18
New	4	6	2	2

2. Hardware Implementation

Synthesis is made with TSMC 90nm technology, for real-time HD application, our design costs 300K logic gate with max frequency at 300MHz. The circuit implementation comparison between prior typical full search and 3-stage pipeline FME architecture and the new proposed architecture is shown in Table 3. It shows the crucial units' differences in quantity. In the architecture all VBS modes including 16x18, 16x8, 8x16 and 8x8 and all four inter prediction modes containing Forward prediction, Backward prediction, Symmetric prediction and Direct prediction are supported. The design contains chroma interpolation to make it a complete FME module.

The proposed algorithm and architecture is also embedded in an AVS HD encoder which is implemented on a Xilinx Virtex–XC6VLX760 FPGA prototyping system. It costs 18% slice LUTs and 18% slice registers to support 1080pHD@30fps, while the whole encoder takes 82% slice LUTs and 84% slice registers. The frequency is 110MHz for the encoder system including our proposed FME architecture to support real-time encoding of HD video.

V. SUMMARY

In summary, an efficient FME architecture for real-time HD video compression of AVS is proposed. The experimental results confirm that the novel algorithm and architecture saved 75% circuit area and 68% computational complexity with acceptable performance lose, and that successfully support the real-time HD coding @30fps with the frequency of 100Mhz.

REFERENCES

[1] Z. Zhou and M. T. Sun, "Fast macro block inter mode decision and motion estimation for H. 264/MPEG-4 AVC," in Proc. Int. Conf. Image Process., vol. 2. 2004, pp. 789–792.

[2] W. Lin, K. Panusopone, D. M. Baylon, and M.-T. Sun "A new class based early termination method for fast motion estimation in video coding," in Proc. IEEE Int. Symp. Circuits Syst., May 2009, pp. 625–628.

[3] H.-Y. C. Tourapis and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in Proc. Int. Conf. Multimedia Expo, vol. 3. 2003, pp. 517–520.

[4] Xiangyung Ji, Debin Zhao, Wen Gao, Qingmin Huang, Siwei Ma. Yun Lu, "New Bi-Prediction Techniques for B Pictures Coding" in ICME, 2004, 101−104.

[5] Yu-Jen Wang, Chao-Chung Cheng, and Tian-Sheuan Chang, "A Fast Algorithm and Its VLSI Architecture for Fractional Motion Estimation for H.264/MPEG-4 AVC Video Coding" in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 5, May 2007.

[6] Chung-Ming Kuo, Yu-Hsin Kuan, Chaur-Heh Hsieh, and Yi-Hui Lee, "A Novel Prediction-Based Directional Asymmetric Search Algorithm for Fast Block-Matching Motion Estimation" in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 19, No. 6, June 2009.

[7] Liu Ying and Zhang Rui, "Optimization on Motion Estimation and DSP Algorithm Based On AVS Encoding" in Computer Applications and Software Vol. 27, No.12, Dec 2010.

[8] H. B. Yin, H. G. Qi, H. Z. Jia, D. Xie, W. Gao, "Efficient macro block pipeline structure in high definition AVS video encoder VLSI architecture" in Proc. ISCAS, 2010, pp. 669-672.

[9] Jeong Jechang, "Fast sub-pixel motion estimation having lower complexity" in IEEE International Conference on Consumer Electronics, 2003. ICCE. 17-19. June 2003.174-175.

[10] Weiyao Lin, Krit Panusopone, David M. Baylon, Ming-Ting Sun, Zhenzhong Chen and Hongxiang Li, "A Fast Sub-Pixel Motion Estimation Algorithm for H.264/AVC Video Coding" in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 21, No. 2, February 2011.