# PMRME: A PARALLEL MULTI-RESOLUTION MOTION ESTIMATION ALGORITHM AND ARCHITECTURE FOR HDTV SIZED H.264 VIDEO CODING

*Chia-Chun Lin, Yu-Kun Lin, and Tian-Sheuan Chang*

Institute of Electronics
National Chiao-Tung University
HsinChu, Taiwan
{cclin, yklin, tschang}@twins.ee.nctu.edu.tw

**ABSTRACT**

The paper presents a hardware-efficient fast algorithm and its architecture for large search range motion estimation (ME) used in HDTV sized H.264 video coding. To solve the high cost and latency in large search range case, the proposed algorithm processes ME in parallel multi-resolution levels instead of serial process in the previous approach. This enables high data reuse for lower bandwidth and low memory cost. Further combining with our previous proposed mode filtering and bit truncation, the algorithm only increases the bit rate within -0.58% and 3.06% and at most 0.04dB and 0.07dB PSNR degradation for 720p and 1080p sequences respectively. The hardware implementation can save up to 49.5% of area cost and 65% of memory cost compared to the previous approach for large search range to [-128, 127].

***Index Terms***— H.264**,** motion estimation, multi-resolution

## 1. INTRODUCTION

MPEG-4 AVC/H.264 video coding [2] is the latest video coding standard that provides good coding efficiency and is widely adopted in emerging multimedia devices. In which, the variable block size motion estimation (VBSME) and its improved sub-pixel precision ME contributes a lot for coding efficiency but also dominates the computational loading of the whole encoding process. Thus VLSI realizations of VBSME have been widely proposed to speedup the process. However, most of them are only applicable for SDTV size or below. For HDTV size applications that requires large search range up to [-128, 127] or even larger, previous approaches will consume too much area cost and computational cycles.

To support large size research range, many fast integer ME algorithms have been proposed. However, most of them are not suitable for hardware implementation due to nearly prohibited memory bandwidth resulted from the poor data reuse flow. To solve this problem, one promising approach is the multi-resolution ME as one proposed in [1]. In [1], they use three hierarchical levels for search and refine the motion vector from the coarse level to the finest level. However, this approach has several disadvantages for hardware implementation. First, the motion vector found in the higher level needs to be further refined in the lower level. It means the search is a sequential process that will increase the cycle counts, and decrease the hardware utilization and throughput. Second, a full search range sized buffer is still needed because the dependency between the three hierarchical levels. It greatly increases the hardware costs for large search range design and diminishes the benefits of multi-resolution ME. Third, the required bandwidth is still quite large due to poor data reuse of the refinement process.

To solve above problem, this paper proposes a parallel multi-Resolution ME (PMRME) algorithm and its architecture. The proposed algorithm uses three independent levels for search. The first two levels with data subsampling cover the large search range to find the rarely occurred large search vector. These two levels have good data reuse by fixing the searching center at (0, 0). On the other hand, the third level without data subsampling covers the search range with the most occurred MV. This level has the search center at the motion vector predictor. The concept behind our algorithm is the unequal distribution of motion vector that most of them are near the motion vector predictor. Thus, a fine search around the motion vector predictor can find the most of motion vector while the rest of motion vector can be found in the coarse search. With above approaches, we can save at least 92.4% memory buffer compared with the previous method. Besides, data within two out of three memory buffers are highly reused, and thus can save about 64.8% of memory bandwidth.

The rest of the paper is organized as following. We first introduce our algorithm in Section 2 and its architecture in Section 3. The results and comparisons are presented in Section 4. Finally, a conclusion is made in Section 5

## 2. THE PROPOSED ALGORITHM

### 2.1 Parallel Multi-Resolution ME (PMRME)

PMRME includes three levels and all of them are independent to each other, as illustrated in Fig. 1. In the coarsest level, level 2, the SR is the largest, [-128~124], and centered on the original point (0, 0). This enables regular memory reuse between successive MB processing. This level uses the 16:1 sampling and thus we only choose the 16x16 mode (mode 1 in Table 1) since other modes will contain too fewer pixels for SAD calculation.

In level 1, the SR is reduced to [-32 ~ +30] and also centered on (0, 0) for the same reason as level 2. This level uses the 4:1 sampling and thus we only choose the 16x16 to 8x8 mode (mode 1 to 4 in Table 1) since other modes will contain too fewer pixels for SAD calculation.

In the finest level, level 0, the SR is set to [-8~+7]. However, unlike the other two levels with (0, 0) center, we choose the predictive motion vector (MVP) as the center due to its higher probability for final MV. Thus, we do not subsample data in this level and thus enable search for all variable block size modes.

In the three parallel levels, the level 2 provides a large search range for high motion blocks with coarse precision. It is useful for very high motion blocks, and can find a good enough though approximate motion vector candidate. Also, the level 1 can provide a medium search range but a finer precision. The level 2 and level 1 are complementary to each other. With these two large search levels, the algorithm can rapidly converge for the motion search of the level 0 by effects of MVP. If only the level 0 is used, it is difficult to trace the high motion blocks because the MVP cannot follow up the real motion effectively in this case.

### 2.2 Mode Filtering

To further reduce the complexity, we use our previously proposed mode filtering method [1]. Thus, only two modes in the integer ME stage will be passed to the fraction ME stage to significantly reduce the fractional ME cycle. Besides, the method also increases the overall ME pipelining efficiency.

### 2.3 Bit Truncation

Two degrees of bit truncation [4] are used in our design. In our analysis (as in Table 4), five bits precision is enough to provide a good coding efficiency for the 720p sequences. However, at least six bits is needed for the 1080p sequences because of the very high definition characteristics. In our analysis, five bits for 1080p will cause larger bit rate increasing.

By using the bit truncation method, about 38% and 28% hardware cost are saved for “5 bits precision” and “6 bits precision” respectively.

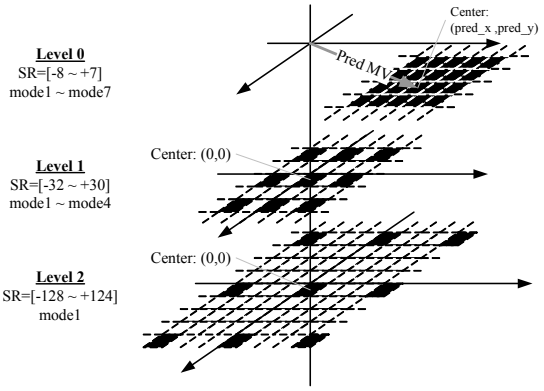


**Fig. 1** the three level new multi-resolution algorithm.

**Table 1** the mode type and its block size for H.264

| Mode | Block size |
|---|---|
| Mode 1 | 16x16 |
| Mode 2 | 16x8 |
| Mode 3 | 8x16 |
| Mode 4 | 8x8 |
| Mode 5 | 8x4 |
| Mode 6 | 4x8 |
| Mode 7 | 4x4 |

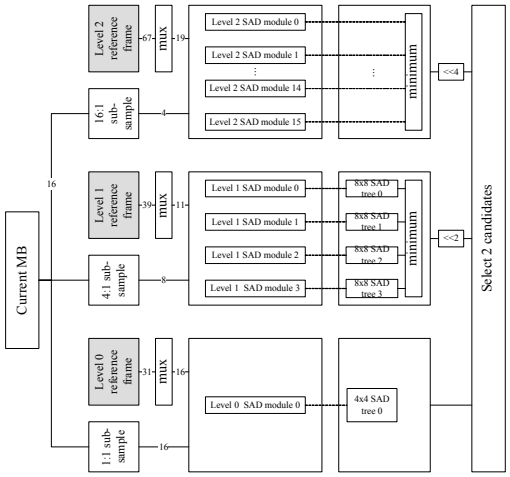


**Fig. 2** the proposed architecture.

## 3. THE PROPOSED ARCHITECTURE

Fig. 2 shows the proposed architecture. A 16x16 current block is shared for the three levels. The memory size and bandwidth for three reference frame buffers are listed in Table 2. The bit width of memory buffer of level 1 and level 2 are also truncated, while that of the level 0 is not. The reason for this is that the level 0 data can be reused by the following fraction ME hardware if the best motion vector falls in the level 0, which can also save the bandwidth.

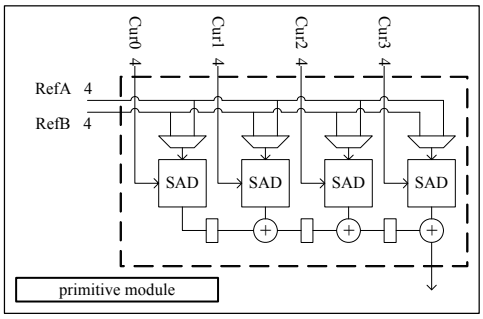


**Fig. 3** the primitive module

**Table 2** memory and bandwidth for different frame size

| Memory cost | for 720p | | for 1080p | |
|---|---|---|---|---|
| | buffer size | BW(per MB) | buffer size | BW(per MB) |
| Level 0 (Kbyte) | 0.992 | 0.992 | 0.992 | 0.992 |
| Level 1 (Kbyte) | 0.975 | 0.312 | 1.170 | 0.312 |
| Level 2 (Kbyte) | 2.8475 | 0.268 | 3.417 | 0.268 |
| Total (Kbytes) | 4.8145 | 1.572 | 5.579 | 1.572 |
| Direct design | 73.712 | 4.336 | 73.712 | 4.336 |
| Saving (%) | 93.46 | 64.8 | 92.43 | 64.8 |

**Table 3** hardware cost comparison

| | | SR | Cell area(K) | Memory (Kbyte) | Frequency (MHz) |
|---|---|---|---|---|---|
| T.C. Chen [5] | | H:+-64 V:+-32 | 305 | 13.71 | 108 |
| Ours | for 720p | H:+-128 V:+-128 | 154 | 4.8 | 100 |
| | for 1080p | | 180.1 | 5.6 | |
| Saving | for 720p | | 49.5 % | 65.0% | |
| | for 1080p | | 41.0% | 59.1% | |

In this architecture, all computations are decomposed as the combinations of 4x4 blocks, denoted as "primitive module". This is the basic module to compute the SAD as depicted in Fig. 3. With this, every level can be easily implemented by regularly composed module. Thus, level 0 has 16 primitive modules respectively. For level 1 with 4:1 subsampling, only four primitive modules are needed for one "level 1 SAD module" for one search point. To further speedup the processing, we adopt four "level 1 SAD module". Thus, totally 16 primitive modules are used as that in level 0. Similarly for level 2 with 16:1 subsampling, only one primitive module is needed for one "level 2 SAD module" for one search point. We also speedup the level 2 by 16 "level 2 SAD module" and thus have the same area cost as level 0. The reason for such speedup to balance the computation cycles for different levels. Thus, computations for all levels can be done in 256 cycles.

The SADs generated from the SAD modules are further summed up to generate the SAD of different block size. Level 0 has the most complex summation trees for combination of the seven kinds of block types. For the level 1, four "8x8 SAD tree" are used for combination of the mode 1 to mode 4 block types. But in level 2, only comparators and registers are needed to select the minimum SAD cost. Finally, the selection module will choose the best two SAD costs from different levels for the fractional ME module.

## 4. EXPERIMENTAL RESULTS

Table 4 shows the simulation result for different parameters, PARME only, PARME with mode filtering (PARME+MF), PARME+MF with bit truncation. At last, we also test the algorithm performance for high motion sequences by skipping two frames.

The simulation environments are as following: No rate-distortion optimization (RDO); sequence type IPPP and SR is [-128, 127]. All of the simulation results are compared with that of the Fast-Full-Search (FFS) Algorithm in JM9.0 [6]. The result in this table only shows the average performance under different QPs due to the page limit. For 720p, the test sequence including: Stockholm, parkrun, and shields. The frame rate is 50 and 50 frames are coded. For 1080p, the test sequence including: station2, rush_hour, and sunflower. The frame rate is 25 and 100 frames are tested. The 1920x1080p image is truncated to an image of 1920x1072 to fit the multiples of 16.

The simulation shows the PMRME alone can achieve the similar video quality as the FFS. However, the distortion is larger under high OP because the blocky effect will be more serious for high QP and mislead the subsampling method. Further combining with MF, we can find that the bit-rate for low QP case sometimes lower than the FFS with the penalty of PSNR loss. If the truncation method is combined, the performance is a little worse. For 720p sequences, it has 0.04dB PSNR loss but 1.62% of bite-rate decrease in average. As for 1080p sequences, it has 0.07dB PSNR loss and up to 1.20% of bit-rate increase in average. However, with slightly quality loss, we can save a lot of hardware cost as described in the previous section. For high motion sequences simulated by 2 frame skipping, the proposed design keeps the similar quality. It means the algorithm does well even for high motion sequences.

In this table, the term "hit rate" means the percentages of motion vector falls in layer 0. With this, the memory data of level 0 can be directly reused by fraction ME and thus save a lot of bandwidth. In our design, the hit rate is at least 87%, and the higher QP will have higher hit rate and thus can save more power and BW. In our analysis, the hit rate for 720p and 1080p is quite the same, so we conclude the search range 128 is enough for the 720p and 1080p sequence. However, the bit truncation method has stronger impact for 1080p. As we can see, the bit truncation method will lead to 1.20% bit-rate increasing for 1080p sequence. The rate-distortion (RD) curves of 720p and 1080p sequence are shown in Fig. 4 and Fig. 5. As we can see, the RD curves are almost overlapped with that by FFS.

The proposed design has been implemented and synthesized. Table 3 shows the total hardware cost of our design and comparison to other design. Our design can save at least 40% of area costs and 60% of memory costs.

## 5. CONCLUSION

In this paper, we propose a parallel multi-resolution algorithm and architecture of integer ME for H.264/AVC. The algorithm uses PMRME, MF and bit truncation to support large search range within 256 cycles. With data reuse and parallel multi-resolution, we can save at least 92.43% of memory buffer and 64.8% of bandwidth. The resulted hardware can save up to 49.5% of area cost and 65% of memory cost compared to the previous approach for 720P processing. With above features, the proposed design is very suitable for larger search range application such as HDTV in a more economical way.

## 6. REFERENCES


[1] C.C. Lin and et al,"A fast algorithm and its architecture for motion estimation in MPEG-4 AVC/H.264 Video Coding", *Asia Pac. Con. on Cir. and Sys. (APCCAS'06)*, pp.1250-1253, Dec. 06

[2] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ ISO/ IEC14496-10 AVC), Mar. 2003

[3] Jae Hun Lee and Nam Suk Lee, "Variable block size motion estimation algorithm and its hardware architecture for H.264/AVC," *Int. Sym. on Cir. and Sys. (ISCAS'04)*. Vol. 3, pp. 741-4, May 2004

[4] Z.-L. He and et al,"Low-power VLSI design for motion estimation using adaptive pixel truncation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 669–678, Aug.2000

[5] T.C. Chen and et al, "Analysis and Architecture Design of an HDTV720p 30 Frames/s H.264/AVC Encoder," *IEEE Trans. Cir. Syst. Video Tech.*, vol. 16, no. 6, pp. 673–688, June 2006

[6] Joint Video Team Reference Software JM9.0, ITU-T


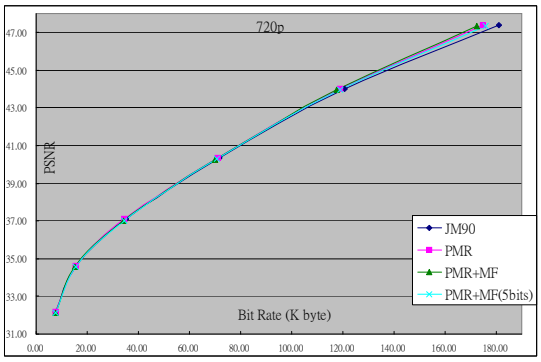


**Fig. 4** The RD curve of 720p sequence

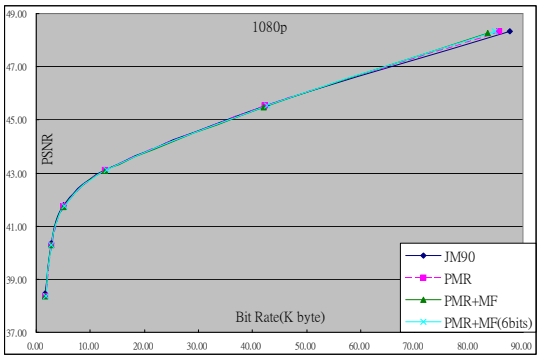


**Fig. 5** The RD curve of 1080p sequence

Table 4 The simulation results

| | Frame size | 720p | | | | | 1080p | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| QP | | PMRME | PMRME+MF | PMRME+MF (5 bits) | PMRME+MF (5 bits) Skip 2 frame | Hit rate (%) | PMRME | PMRME+MF | PMRME+MF (6 bits) | PMRME+MF (6bits) Skip 2 frame | Hit rate (%) |
| QP12 | PSNR inc.(db) | -0.01 | -0.02 | -0.02 | -0.02 | 88.62 | 0.00 | -0.08 | -0.09 | -0.05 | 87.69 |
| | Bit rate inc. (%) | -3.53 | -4.87 | -2.97 | -4.10 | | -2.14 | -4.62 | -3.00 | -3.93 | |
| QP16 | PSNR inc.(db) | 0.00 | -0.05 | -0.03 | -0.04 | 90.41 | 0.00 | -0.07 | -0.06 | -0.06 | 89.57 |
| | Bit rate inc. (%) | -1.33 | -2.56 | -1.56 | -2.38 | | -0.49 | -1.09 | 0.47 | -0.47 | |
| QP20 | PSNR inc.(db) | 0.01 | -0.09 | -0.07 | -0.07 | 91.67 | -0.01 | -0.04 | -0.04 | -0.03 | 92.33 |
| | Bit rate inc. (%) | -0.94 | -2.40 | -1.38 | -2.13 | | -0.44 | -0.22 | 2.65 | 1.89 | |
| QP24 | PSNR inc.(db) | 0.00 | -0.07 | -0.06 | -0.06 | 93.72 | -0.03 | -0.06 | -0.06 | -0.07 | 93.19 |
| | Bit rate inc. (%) | -1.26 | -2.73 | -1.63 | -2.69 | | -0.40 | -0.94 | 1.83 | 1.03 | |
| QP28 | PSNR inc.(db) | 0.00 | -0.05 | -0.04 | -0.05 | 95.14 | -0.06 | -0.07 | -0.08 | -0.07 | 93.47 |
| | Bit rate inc. (%) | -0.86 | -2.09 | -1.57 | -2.91 | | 0.40 | 0.19 | 2.20 | 1.64 | |
| QP32 | PSNR inc.(db) | -0.01 | -0.05 | -0.04 | -0.05 | 95.63 | -0.10 | -0.09 | -0.10 | -0.09 | 93.39 |
| | Bit rate inc. (%) | 0.27 | -0.96 | -0.58 | -1.75 | | 1.68 | 1.59 | 3.06 | 2.54 | |
| Avg. | PSNR inc.(db) | 0.00 | -0.06 | -0.04 | -0.05 | 92.53 | -0.03 | -0.07 | -0.07 | -0.06 | 91.61 |
| | Bit rate inc. (%) | -1.28 | -2.60 | -1.62 | -2.66 | | -0.23 | -0.85 | 1.20 | 0.45 | |