Journal of Computer Engineering
ISSN 10003428

Original Research Article
Vol 9, Issue 4, 2020 | ISSN: 1000-3428

Optimizing Processing Element Configurations for Efficient High Efficiency Video Coding

Min-Soo Kim and Ji-Hwan Cho

Department of Electrical Engineering, Seoul National University, Seoul, South Korea; and Department of Computer Science, Hanyang University, Seoul, South Korea

***Abstract*—**Motion estimation occupies the heaviest computation in HEVC (high efficiency video coding). Many fast algorithms such as TZS (test zone search) have been proposed to reduce the computation. Still the huge computation of the motion estimation is a critical issue in the implementation of HEVC video codec. In this paper, motion estimator architecture with optimized number of PEs (processing element) is presented by exploiting early termination. It also reduces hardware size by exploiting parallel processing. The presented motion estimator architecture has 8 PEs, and it can efficiently perform TZS with very high utilization of PEs.

I. Introduction

With the advent of UHD (ultra-high definition) images, a new efficient video coding technique has been required for video compression and transmission. HEVC [1], [2] is newly-established international standard for video coding by JTC1 (Joint Technical Committee) of ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) and VCET (Video Coding Experts Group) of ITU-T (International Telecommunication Union - Telecommunication Standardization Sector). It achieved significant improvement in compression ratio, but it suffers from huge computation due to complicated compression techniques.

Motion estimation [3]-[6] is the most computation-intensive process in HEVC. There are many types of motion estimation, but BMA (block-matching algorithm) [3] is widely used in the hardware implementation. BMA divides the current frame into several blocks. Each block (this is called the current block) is matched with some blocks (they are called search window blocks, and their positions are called search positions) in some neighborhood region (this is called a search window). The most similar block (this is called a matched block) is found by iterative matching. SAD (sum of absolute difference) is used as a matching criterion, and lower SAD means more similar. Two components are encoded instead of the whole reference block – (1) the block difference (this is called as residuals) between the current and matched blocks, and (2) their relative position displacements (this is called as motion vector). This is illustrated in Fig. 1 in detail.

FS (full search) [3] exhaustively matches all search positions in the search window. It shows the best compression ratio, but it suffers from huge computation when the image size is large. Many fast algorithms such as TSS (three-step search) [4], [5] and TZS [6] have been proposed to reduce the computation by skipping some search positions.

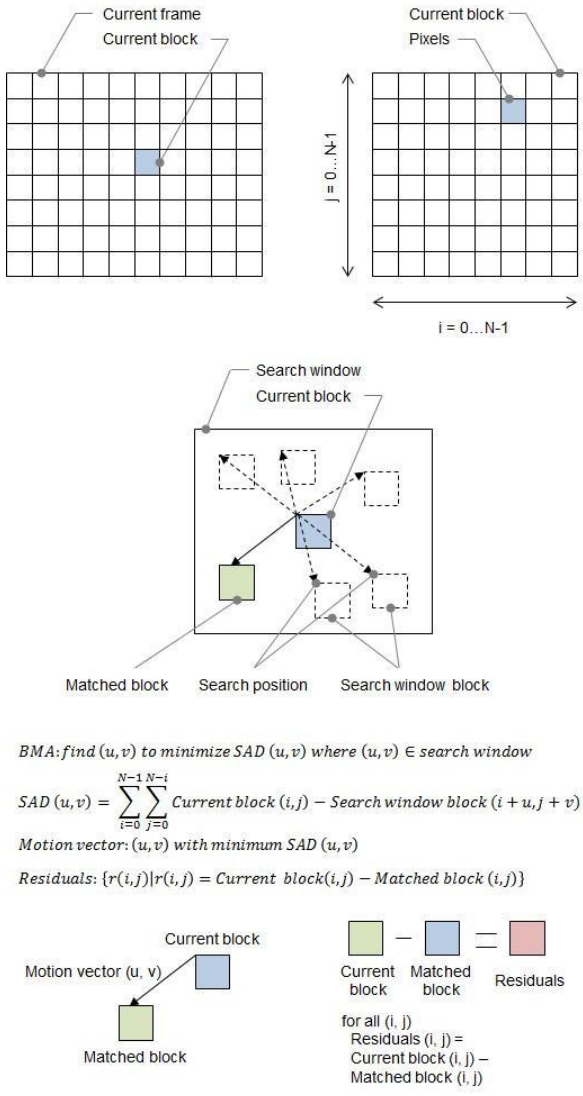

Fig. 1 Block matching algorithm

TZS is known as one of the most efficient motion estimation algorithms in HEVC. It first performs a sparse scan in the search window to find the most possible region where the moving object exists. Then it performs a medium

How to cite: Min-Soo Kim and Ji-Hwan Cho (2020). Optimizing Processing Element Configurations for Efficient High Efficiency Video Coding. Journal of Computer Engineering, 9(4), 8–10. DOI: https://doi.org/10.5281/zenodo.19341539

scan with subsampling. After that, it performs a precise scan to find the final matched block.

TZS significantly reduces the computation of FS, since it skips most search positions in the search window. Many researches have shown that its compression ratio is comparable with FS, while its computation is lower than 1/1000 with FS. However, TZS exploits several different stages with different block sizes, different search directions, and different resolution. In the hardware implementation of TZS motion estimator, these irregularities significantly increases the hardware complexity and they severely hinder the reuse of PEs (processing element)

This paper presents a motion estimator architecture for TZS search in HEVC coding. It fully supports early termination to reduce motion estimation computation. It has multiple PEs for parallel processing, and the number of PEs is optimized to achieve high utilization of hardware resources.

II. Test Zone Search

As illustrated in Fig. 2, TZS consists of five major stages, i.e. motion vector prediction, grid search, raster search, star refinement, and raster refinement. Note that either star refinement or raster refinement is exploited in general. Most TZS motion estimators have four stages.

(1) Motion Vector Prediction

As illustrated in Fig. 3, SADs are calculated with four motion vectors, i.e. motion vectors of neighboring blocks (*MV1*, *MV2*, and *MV3*) and their median (*MV4*). Motion vector with minimum SAD is determined as *PMV*.

(2) Grid Search

Whole search window is searched with diamond (or square) pattern, as illustrated in Fig. 4. Default pattern is diamond pattern, but square pattern can be also used. It is a single-step search, i.e. diamond (or square) pattern is used only once. Displacement between minimum SAD position and *PMV* position is *uiBestDistance*.

(3) Raster Search

Raster search is a subsampled full search with an interval of *iRaster* pixels, as illustrated in Fig. 5. When *uiBestDistance* is smaller than *iRaster*, this step is skipped.

(4) Star Refinement

Star refinement is a multi-step search using diamond (or square) pattern. In each step, the size of the diamond (or square) pattern halves until its size is 1 (=4 search positions). When its size reaches 1, additional two missing points are searched.

(5) Raster Refinement

Raster refinement is similar with star refinement, but only the outer 8 search positions of diamond (or square) pattern are searched in each step, as illustrated in Fig. 6.

When multiple PEs are exploited in the TZS motion estimator, ET (early termination) should be considered where some computation is skipped during TZS. When ET occurs, some PEs are idle due to the skipped computation. However, the execution time can be reduced only when all PEs are idle.

There are two types of early termination in TZS. Inter-stage early termination occurs when TZS skips some stages in the flowchart of Fig. 2. Intra-stage early termination occurs during block matching, as illustrated in Fig. 7.

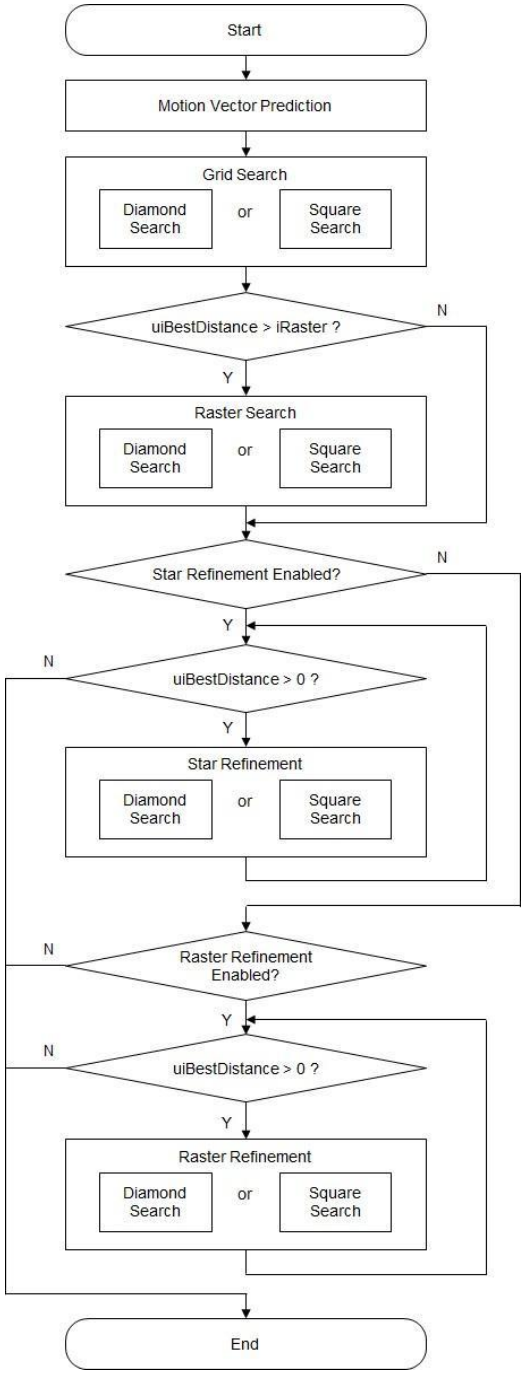

Fig. 2 TZS (test zone search) algorithm

In the multiple PEs, inter-stage early termination does not need to be considered, since all PEs are idle by skipping whole stage. However, intra-stage early termination is different. When many PEs are exploited for parallel processing, the execution time may not be reduced in proportion to the number of PEs. In this paper, the number of PEs is optimized by considering intra-stage early termination.

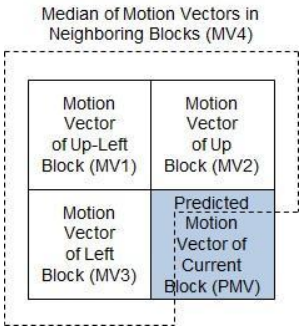

Fig. 3 Motion vector prediction

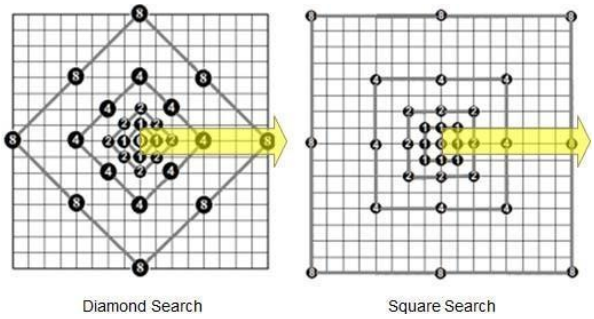

Fig. 4 Grid search

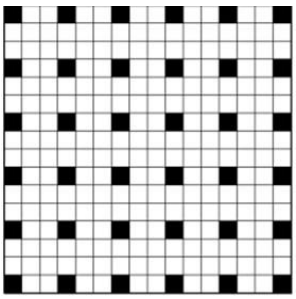

Fig. 5 Raster search

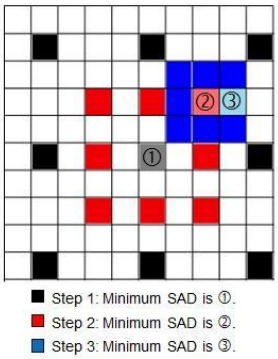

Fig. 6 Raster refinement

```
// N: current block size
// cur(x,y): pixel value of current block
// sw(x,y): pixel value of search window
// swx: search window x size
// swy: search window y size
// mvx: motion vector x position until now
// mvy: motion vector y position until now
// sadmin: minimum SAD value until now

sadmin = MAXINTEGER;
for (u=-swx; u<swx; u++)
 for (v=-swy; v<swy; v++) {
  sad = 0;
  for (i=0; i<N; i++) {
   for (j=0; j<N; j++)
    sad += abs (cur(I,j) – sw(i+u,j+v));
    if (sad > sadmin)
     break;
  }
  if (sad < sadmin) {
   mvx = u;
   mvy = v;
   sadmin = sad;
  }
}
```

Fig. 7 Intra-stage early termination

III. Proposed Motion Estimator Architecture

Most motion estimators exploit multiple PEs for parallel processing. Usually, each PE processes each search position. Number of PEs is important in the motion estimator architecture. More PEs reduces execution time by parallel processing, but more PEs are idle due to the following reasons.

(A) Some PEs are idle due to intra-stage early termination, and the number of idle PEs increases along with the number of PEs.

(B) Some PEs are idle when the number of currently processing search positions is smaller than the number of PEs.

For (A), the number of idle PEs and the reduction of execution time are dynamically changed on the image sequences. Therefore, this effect cannot be optimized. However, it is obvious that early termination significantly reduces execution time. Therefore, the proposed motion estimator architecture has been designed to support early termination in hardware level.

For (B), the number of idle PEs and the reduction of execution time are deterministic for given motion estimation stages. Tables I and II show the utilization of the PEs and the execution cycles per current block for various number of PEs, respectively.

From Tables I and II, it is obvious that the optimum number of PEs is 8. When the number of PE is larger than 8, the utilization of PEs significantly decreases, but the execution time does not decrease. When the number of PE is smaller than 8, the utilization of PEs increases the execution time significantly increases.

Fig. 8 shows the architecture of the proposed motion estimator with 8 PEs. It is simple but efficient, and it can support early termination. The proposed motion estimator is currently under design, and it will be implemented in SoC (system-on-chip) next year.

TABLE I
Utilization of PEs

Stages	4 PEs	8 PEs	12 PEs	16 PEs
Motion Vector Prediction	100.0%	50.0%	33.3%	25.0%
Grid Search	90.6%	87.5%	60.4%	45.3%
Raster Search	87.5%	87.5%	58.3%	43.8%
Star Refinement	96.9%	96.9%	64.6%	48.4%
Raster Refinement	96.9%	96.9%	64.6%	48.4%

TABLE II
Execution Cycles per Current Block

Stages	4 PEs	8 PEs	12 PEs	16 PEs
Motion Vector Prediction	1024 cycles	1024 cycles	1024 cycles	1024 cycles
Grid Search	8192 cycles	4096 cycles	4096 cycles	4096 cycles
Raster Search	9464 cycles	4732 cycles	4732 cycles	4732 cycles
Star Refinement	8192 cycles	4096 cycles	4096 cycles	4096 cycles
Raster Refinement	8192 cycles	4096 cycles	4096 cycles	4096 cycles

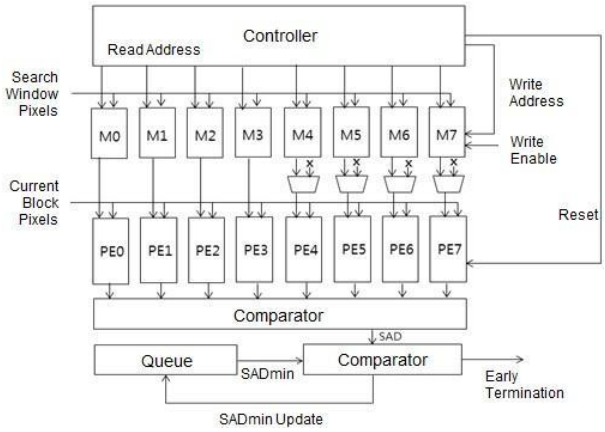

Fig. 8 The proposed motion estimator architecture

References

[1] G. Sullivan, J. Ohm, W. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[2] J. Lee and S. Lee, “8×8 HEVC Inverse Core Transform Architecture Using Multiplier Reuse”, Journal of IKEEE, vol. 17, no. 4, pp. 570-578, Dec. 2013.

[3] J. R. Jain and A. K. Jain, “Displacement measurement and its application in interframe image coding”, IEEE Transactions on Communications, vol. 29, no. 12, pp. 1799-1808, Dec. 1981.

[4] H. Jong, L. Chen, and T. Chiueh, "Parallel Architectures for 3-Step Hierarchical Search Block-Matching Algorithm," IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 407-416, Aug. 1994.

[5] H. Yang and S. Lee, “Motion Estimation Algorithm to Guarantee Hard Realtime Operation”, Journal of IKEEE, vol. 17, no. 1, pp. 36-43, Mar. 2013.

[6] X. Li, R. Wang, W. Wang, Z. Wang, and S. Dong, “Fast motion estimation methods for HEVC,” in Proc. of IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2014, pp.1-4.