Implementation and optimization of sub-pixel motion estimation on BWDSP platform

Shangzhu Jia, Wenhui Lang, Feiyang Zeng and Yufu Liu

School of Computer and Information, Hefei University of Technology, HeFei, 230009, China

Abstract. Sub-pixel Motion estimation algorithm is a key technology in video coding inter-frame prediction algorithm, which has important influence on video coding performance. In the latest video coding standard H.265/HEVC, interpolation filters based on DCT are used to Sub-pixel motion estimation, but it has very high computation complexity. In order to ensure the real-time performance of hardware coding, we combine the characteristics of BWDSP architecture, using code level optimization techniques to realize the sub-pixel motion estimation algorithm. Experimental results demonstrate that In the BWDSP simulation environment, the proposed method significantly decreases the running clock cycle and thus improves the performance of the encoder.

1 Introduction

The High Efficiency Video Coding (HEVC) standard is the most recent joint video project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC). the HEVC standard is finalized in January 2013. Comparing with H.264 / AVC, HEVC make the compression ratio doubled when the video quality is similar [1-2]. Inter prediction [3] technology is the use of image time domain in the adjacent frame between the temporal correlation of video information in the removal of temporal redundancy information. The main principle is to find a best matching block in the previously coded image for each pixel block of the current image, called motion estimation. The result of the motion estimation is to obtain the difference between the current block and the reference block and the current pixel block Motion vector; the prediction of motion vector is achieved through Merge technology [4-5] and AMVP technology [6-7]. Motion estimation algorithm is generally divided into: pixel recursion, regional matching and block matching method, compared to the pixel recursive method, regional matching method, block matching method to take into account the accuracy and complexity of motion estimation, So the motion estimation in HEVC adopts the block matching method. In the Motion estimation algorithm, motion search accounts for more than 50% of the total coding time [8]. In HEVC, the method of full search in the sub-pixel search region is used to determine the optimal motion vector according to the rate distortion cost (Rdcost) [9]. Full search can get the best results, but a huge amount of computation. BWDSP [10-11] is designed by the China Electronics Technology Group Corporation No. 38 Research Institute, it is a multi-core high-performance general-purpose floating-point DSP. This paper based on the characteristics of BWDSP architecture, code-level optimization techniques are used to realize the motion estimation algorithm [12]. Which improves the performance of motion estimation.
2 Motion estimation

In some specific application environments, the transmission of video coding is very demanding on real-time, and the computational complexity of motion estimation is usually relatively high, so it is very important to find high performance and low complexity motion search algorithm. Currently used search algorithms are: full search algorithm, diamond search algorithm, TZSearch algorithm. The full search algorithm refers to all the possible positions in the search window to calculate the matching error of the two blocks. The MV corresponding to the minimum matching error must be the global optimal MV. However, the full search algorithm is extremely complex and cannot meet the real-time edge coding. In addition to the full search, the rest of the search algorithm collectively referred to as the fast search algorithm, the fast search algorithm has a little bit of search speed, but its search process easily fall into the local optimal point, which cannot find the global optimal point. In order to avoid this phenomenon, try to search for more points in every step of the search algorithm. Diamond search algorithm, TZSearch search algorithm is currently used more than others algorithms.

In HEVC, Sub-pixel motion estimation mainly includes sub-pixel interpolation and sub-pixel search. HEVC uses a discrete cosine transform (DCT) interpolation filter to perform 1/2 pixel and 1/4 pixel interpolation. The distribution of the pixels after interpolation is shown in Figure 1. Where the square represents the integer pixel, the circle represents 1/2 pixel, the diamond represents 1/4 pixel. Figure 1 shows only a 1/2 pixel point around the integer pixel point 8 and a 1/4 pixel point around the 1/2 pixel 1.

![Figure 1: motion estimation algorithm.](image)

The motion estimation procedure is as follows:
(a) First, the integer pixel search to find the best integer pixel within the search range, calculate the search point Rdcost value, to find the best integer pixel, assuming that the integer pixel 8 is the best integer pixel search point.
(b) With the integer pixel 8 as the center, interpolated by the current prediction block of 1/2 pixel value, Calculate the Rdcost values for the eight 1/2-pixel points around pixel 8, Find the best 1/2 pixel, assuming point 1.
(c) With the best 1/2 pixel 1 as the center, interpolated by the current prediction block of 1/4 pixel value, Calculate the Rdcost values for the eight 1/4-pixel points around pixel 1, find the best 1/4 pixels, assuming point 6.
(d) And finally the position of the point 6 is regarded as the optimum dot position, complete the subpixel motion estimation.

3 Sub-pixel interpolation

Since the motion of the natural object is continuous, the motion between the two adjacent images is not necessarily the whole pixel, but it is possible to use the half pixel, 1/4 pixel or even 1/8 pixel unit. At this point if only the use of the whole pixel precision motion estimation will appear inaccurate
matching problem, resulting in a larger range of motion compensation residual, affecting the coding efficiency. In order to solve the above problem, the accuracy of the motion estimation should be raised to the sub-pixel level, which can be achieved by interpolating the reference image pixel. 1/4 pixel accuracy is significantly improved compared to 1/2 pixel accuracy, but the coding efficiency at 1/8 pixel accuracy compared to 1/4 pixel accuracy is not significantly different from the case of high bit rate. Increased and 1/8 pixel precision motion estimation is more complex. So the existing standard H.264 and HEVC are using 1/4 pixel accuracy for motion estimation.

The sub-pixel interpolation is divided into the interpolation process of the luminance block and the interpolation process of the chroma block.

The process of the luminance component interpolation algorithm can be described in detail with reference to the template shown in Figure 2, which can be divided into two main steps:

1) Interpolate the rows or columns of the entire pixel. A0,0, b0,0, c0,0, d0,0, h0,0, n0,0, where a0,0, b0,0, c0 are the same as the sub-pixel points near A0,0. The value of 0 can be calculated from the whole pixel in the horizontal direction. The values of d0,0, h0,0, n0,0 can be calculated from the whole pixel in the vertical direction. The calculation process is as follows:

\[
\begin{align*}
    a_{0,0} &= -A_{-3,0} + 4A_{-2,0} - 10A_{-1,0} + 58A_{0,0} + 17A_{1,0} - 5A_{2,0} + A_{3,0} \\
    h_{0,0} &= -A_{-3,0} + 4A_{-2,0} - 11A_{0,-1} + 40A_{0,0} + 40A_{0,1} - 11A_{0,2} + 4A_{0,3} - A_{0,4}
\end{align*}
\]

The remaining position pixels can be calculated from the corresponding filter.

2) Interpolate the remaining sub-pixel positions, for pixels that are not in the entire pixel row or column position, such as e0,0, f0,0, g0,0, i0,0, j0,0, k0,0, p0,0, Q0,0, r0,0, need to use the first step to get the whole pixel line sub-pixel values (a, b, c position) to calculate, calculated as follows:

\[
\begin{align*}
    e_{0,0} &= (-a_0, -3 + 4a_0, -2 - 10a_0, -1 + 58a_0, 0 + 17a_0, 1 - 5a_0, 2 + 4a_0, 3) >> 6 \\
    j_{0,0} &= (-b_0, -3 + 4b_0, -2 - 11b_0, -1 + 40b_0, 0 - 11b_0, 2 + 4b_0, 3 - b_0, 4) >> 6 \\
    r_{0,0} &= (c_0, -2 - 5c_0, -1 + 17c_0, 0 + 58c_0, 1 - 10c_0, 2 + 4c_0, 3 - c_0, 4) >> 6
\end{align*}
\]

The remaining position pixels can be calculated from the corresponding filter.

![Figure 2: Fractional positions in luma interpolation](image)

Chroma block interpolation process is similar.
The sub-pixel interpolation at the pixel block boundary is interpolated using the method of boundary value expansion. The processing of sub-pixel interpolation at the pixel block boundary is shown in Figure 3.

![Figure 3: Pixel Interpolation Schematic of Pixel Block Boundary](image)

The figure shows a 4×4 size chroma block in which the black squares 1, 2, 3 and 4 represent the four largest pixels in the first row, and the four hexagons represent the chrominance blocks 1/2 pixel position corresponding to the interpolation coefficient, black small circle on behalf of the 1/2 pixel. It is now necessary to interpolate the 1/2 pixel between the pixels 1 and 2. The first two sections of the algorithm description are not applicable here, in the chroma 4×4 block pixels 1 and 2 between the 1/2 pixel interpolation process, the second tap coefficient with the first integer pixel Aligned, so that there will be no pixels at the left and the first tap coefficient corresponding to the problem, beyond the boundaries of the pixel block. In the code, the boundary value expansion method is used to interpolate the boundary point, that is, according to the number of tap coefficients, the corresponding boundary pixel shift outward expansion, so that with the corresponding tap coefficient for. As shown in the figure above, when the tap coefficient 2 corresponds to the whole pixel 1 when the interpolation operation is performed on 1/2 pixel between the whole pixel 1 and 2, the tap coefficient 1 cannot find the corresponding Point, which is only need to shift the entire pixel 1 outward expansion of the pixels, which is the whole pixel 1 and 2 between the 1/2 pixel interpolation operation. For the other boundary interpolation of pixel blocks, the same method can be used for arithmetic processing.

4 Architecture of BWDSP

BWDSP processor is a 32bit static superscalar processor, using 16 transmit, SIMD (single instruction stream, multiple data stream) architecture. Processor instruction bus width of 512bit, the internal data bus using asymmetric full-duplex bus, the internal data read bus width is 512bit, the internal data write bus width is 256bit, a total of 11 internal water, operating frequency 1GHz. BWDSP processor core structure called eC104, eC104 core contains four basic execution Macros (Element operation Macro, Macro), each execution Macro consists of 8 arithmetic and logic unit (ALU), 4 multipliers (MUL), 2 shifters (SHF), 1 supercomputer (SPU), and 1 general purpose register group. The data format supported by the arithmetic unit complies with the IEEE754 standard. It supports 16-bit fixed-point, 32-bit fixed point, 32-bit floating point, 16-bit and 32-bit fixed point complex number and 32-bit floating point complex number. Figure 4 shows the framework of BWDSP.
5 Code-level optimization

The BWDSP contains four execution macros, each with eight multipliers, eight adders, four shifters, and one supercomputer, and the four execution macros can be run in parallel, which makes the BWDSP has a larger throughput capacity. By using BWDSP assembler instruction \{x, y, z, t\} Rs + 3: s = [Un + Um, Uk], sixteen 32-bit data can be read or written at one time. If the pixel data is 8-bit data storage, then a 32-bit data can store four eight-bit data, then one can read 16 * 4 = 64 data. In addition, the command ACCs + = ABS (ORm - ORn) (U) divides Rm and Rn into four 8-bit numbers, corresponding to subtraction to get four results, and accumulates the corresponding value of the original ACC into the [9: 0], [19:10], [29:20], [39: 30] bits of the ACC corresponding to the ALU. Use this instruction to achieve the difference between two pixels and take absolute value operation, and the results stored in the ACC accumulator. Using the above instruction can make full use of BWDSP resources, the implementation of the following code 1 time, and then accumulate the value of ACC accumulator together, you can complete a 16*16 pixel SATD calculation.

The above assembly language uses a small amount of code to solve the motion estimation SATD calculation, the optimized clock cycle as shown in Table 1.

Table 1: SATD 16×16 clock cycle comparison function table

| Optimization function | Before optimization | After optimization |
|------------------------|---------------------|-------------------|
| SATD                   | 56605               | 659               |

The sub-pixel search process contains a large number of addition, subtraction, multiplication and shift operations, very suitable for instruction set optimization. Combined with architecture of BWDSP, similar to the SATD operation, the effect of interpolation process by the assembly instruction optimization is obvious. The optimized clock cycle is shown in Table 2.

Table 2: 16×16 block interpolation filter clock cycle comparison function table

| Optimization function | Before optimization | After optimization |
|------------------------|---------------------|-------------------|
| Interpolation Filter   | 500890              | 1932              |
6 Conclusion
In the process of sub-pixel motion estimation, there are a large number of Rdcost calculations, such as motion matching process and sub-pixel interpolation process. By combining these algorithms with BWDSP architecture, we use the optimization of assembly language. The optimization of the sub-pixel motion estimation process is optimized, the optimization effect is obvious, and the efficiency of the encoder is improved.

References
[1] Sullivan G J, Ohm J R, Han W J, et al. “Overview of the High Efficiency Video Coding (HEVC) Standard”, IEEE Transactions on Circuits & Systems for Video Technology, 22(12), pp. 1649-1668, (2012).
[2] Ohm J R, Sullivan G J, Schwarz H, et al. “Comparison of the coding efficiency of video coding standeds---including high efficiency video coding (HEVC). IEEE Trans”. On Circuits and System for Video Technology, 22(12), pp. 1669-1684, (2012).
[3] Lin J L, Chen Y W, Tsai Y P, et al. “Motion vector coding techniques for HEVC[C]”, IEEE, International Workshop on Multimedia Signal Processing, pp. 1-6, (2011).
[4] Oudin S, Helle P, Stegemann J, et al. “Block merging for quadtree-based video coding”, 26(1), pp. 1-6, (2011).
[5] Helle P, Oudin S, Bross B, et al. “Block Merging for Quadtree-Based Partitioning in HEVC”, IEEE Transactions on Circuits & Systems for Video Technology, 22(12), pp. 1720-1731, (2012).
[6] L. Zhao, X. Guo, S. Lei, S. Ma, D. Zhao and W. Gao, Non-CE9: Simplification of AMVP, JCTVC-H0316, 8th JCT-VC Meeting, San Jose, CA, USA, pp. 1-10 February, (2012).
[7] A. Fujibayashi, “Simplified Motion vector prediction”, JCT-VC D231, 4th Meeting: Daegu, Korea, pp. 20-28 January, (2011).
[8] Bossen F, Bross B, Suhring K, et al. “HEVC complexity and implemention analysis”, IEEE Tran. On Circuits and Systems for Video Technology, 22(12), pp. 1685—1696, (2012).
[9] Li H, Li B, Xu J. “Rate-Distortion Optimized Reference Picture Management for High Efficiency Video Coding[J]”, IEEE Transactions on Circuits & Systems for Video Technology, 22(12), pp. 1844-1857, (2012).
[10] Guang dong Lin, Guanghong Huang, Yan Zhu. “Design of C Language Level Debugger for BWDSP Processor [J]”, 25 (Z1), pp. 69-75 (2016).
[11] Xiangqian Wang, Yi Hong, Hao Wang, et al. “Design and Optimization of Compiler for DSP of Soul Core [J]”, Acta Electronica Sinica, 43(8), pp. 1656-1661 (2015).
[12] Wei Wang, Jinheng Xiao, Guoxiong Liu, et al. “A fast subpixel motion estimation algorithm based on HEVC [J]”. Systems Engineering and Electronics, (12), pp. 2543-2549, (2014).