Signal Processing: *Image Communication* 26 (2011) 85–92

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

High performance fractional motion estimation in h.264/avc based on one-step algorithm and 8×4 element block processing

Nam Thang Ta*, Jun Rim Choi

School of Electrical Engineering and Computer Science, Kyungpook National University, Republic of Korea

ARTICLE INFO

Article history:
Received 18 March 2010
Accepted 20 December 2010
Available online 30 December 2010

Keywords:
h.264
Fractional motion estimation
One-step
VLSI
Video signal processing

ABSTRACT

Conventional two-step algorithm, long latency of interpolation and various motion vectors are three factors that mainly induce high computation complexity of fractional motion estimation and also prevent it from encoding high-definition video. In order to overcome these obstacles, a high performance fractional motion engine is proposed in this paper with three techniques. First, based on high correlation between motion vector of a block and its up-layer as well as relationship of integer candidates, one-step algorithm is proposed. Second, an 8×4 element block processing is adopted, which not only eliminates almost redundancies in interpolation, but also still ensures hardware reusability. Finally, a scheme of processing 4×4 and 4×8 block with free of cycles is presented, so that the number of motion vectors can be reduced up to 59%. Experimental results show that the proposed design just needs 50% of gate count and 56% of cycles when compared with previous design while nearly maintaining the coding performance.

1. Introduction

The latest H.264/AVC video coding standard, jointly developed by ITU-T Video Coding Experts Group (VCEG) and ISO/IEC 14496-10 AVC Moving Picture Experts Group (MPEG), has higher compression performance with bit-rate optimisation than previous video coding standard like MPEG-4, H.263 and MPEG-2. These outstanding features come mainly from the new inter-prediction technique that removes temporal redundancy [1]. To enhance inter-prediction, motion estimation (ME) with quarter pixel resolution, variable block sizes and multiple reference frames are adopted. The motion estimation in the latest video standard H.264/AVC is comprised of two parts: first the integer motion estimation (IME) is used to find integer motion vector (IMV), then the fractional motion estimation (FME) is performed to refine this motion vector (MV) to fractional precision. In fractional part, the half pixel refinement is first performed around the best integer search position and then quarter pixel is carried out around the best half position. Though these techniques bring the high encoding performance, the ME engine requires huge amount of computational complexity, especially for high-definition (HD) video application. As a consequence, the accelerated ME engine for HD real time application is obligated.

Previous dedicated architectures adopted several techniques such as parallel ME architecture, fast ME algorithm and also various schemes of reusing data. Ref. [3], for example, used eight processing elements (PEs) for calculating the group of eight consecutive reference candidates. The data reuse schemes in [3] are within the current search area and also among adjacent search window, they are so called level A and level C, respectively, as defined in [4,5]. For fractional motion estimation, conventional FME approach needs two-sequential-step interpolation for both half-pel and quarter-pel refinements. Though this two-step brings the high

* Corresponding author. Postal address: 702-701 Dae-gu, San-kyuk Dong 1370, Kyung-buk University, Building IT3, Room-414, Republic of Korea. Tel.: +82 01 7738 6599; fax: +82 53 950 5505.
E-mail address: thangta@ee.knu.ac.kr (N.T. Ta).

0923-5965/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2010.12.001

encoding performance, it introduces the long latency as well as the huge computational complexity for searching seventeen fractional points for each of forty-one MVs. As a result, the computational load for fractional pixel motion estimation takes a major portion in the total ME, and thus FME has recently become the bottleneck of the whole H.264/AVC encoding system, especially for encoding HD application. Some papers in literature focused on these existing FME issues. The literature [6] adopts nine processing units (PUs) in parallel manner, so that searching nine fractional positions is performed around integer position simultaneously. All blocks are decomposed into 4×4 block so that they can be fully reused in 4×1 pel interpolation and 4×4 Hadamard transform. The 4×1 pel interpolation in [6] eliminates fully the redundancies in vertical direction. However, the redundancies in horizontal direction still exist. Motivated by the high correlation of cost between neighbouring fractional candidate positions, the number of fractional candidates in [7] can be reduced to eight or nine instead of seventeen candidates as reference software JM [10]. Ref. [7] not only reduces the computation complexity by 50% compared with reference software, but also saves hardware cost. However, the issue of long latency has not been solved yet because of two-sequential-step algorithm. To shorten the latency, one-step algorithm is proposed in our previous work [8]. It is a straightforward design and is suitable for hardware with fixed twenty-one fractional-pel positions. Though this architecture can reduce the long latency of interpolation processing by half, it occupies quite a large amount of hardware area.

Obviously, the FME issues mainly come from two-step procedure, long latency of interpolation step and various MVs in fractional precision. In this paper, we propose one-step FME algorithm based on early predicted fractional point. An 8×4 based block processing is also presented to reduce the interpolation redundancies in both vertical and horizontal directions. By exploiting the same motion vector between a block and its up-layers, 4×8 blocks and 4×4 blocks are still processed with free of cycles. These techniques can reduce the FME latency by 78% compared with previous design while almost maintaining the coding performance. In addition, due to huge amount of computation complexity and memory bandwidth requirement in HD hardwire motion engine, our design target aims at 1080p Baseline Profile, level 4.0, one reference frame and disable rate distortion optimisation (RDO).

The rest of the paper is organised as follows. First we review the previous approaches in Section 2. In Section 3, we propose the fractional motion engine with one-step algorithm and 8×4 based block with free of cycles 4×8 and 4×4 processing, followed by its architecture in Section 4. The synthesis results and comparisons are presented in Section 5. Finally, a conclusion is given in Section 6.

2. Related works

To reduce the complexity of fractional processing, Center Biased Fractional Pel Search (CBFPS) was proposed in [9] and has also been adopted into the reference software. Its idea is to simplify search pattern by the predicted fractional motion vector. First this method predicts the fractional-pel MV of the current block using the median MV of neighbouring blocks. Then it examines the zero sub-pel MV and the predicted fractional-pel MV. The one that yields the minimum matching error is chosen to be the start position. Finally, the approach in [9] employs a diamond search pattern to refine the fractional MV. The diamond search just stops with specified number of steps or minimum cost point located at the centre. However, for large blocks, the predicted MVs are not accurate enough. Thus, the H.264 reference software still uses full search for the large blocks including 8×16, 16×8 and 16×16 blocks.

By examining the neighbouring integer positions surrounding the best integer-pel position, the prediction based directional refinement algorithm for fractional pixel motion search strategy is proposed in [11]. A half-stop judgment is also adopted to stop the search process when the current minimum cost is below a certain threshold. According to this approach, the computational load of fractional-pel motion estimation can be reduced from 17.4% to 34.7% of the original full fractional-pel search algorithm. However, this method only estimates four sub-SAD surrounding the best SAD integer and ignores four other positions in diagonal directions. Moreover, this approach seems to be suitable for software implementation only, it cannot be directly adapted to hardware implementation due to the requirement of four surrounding SAD integers. Same as the idea in [11], the approach in [12] early eliminated one half-pel position by estimating four surrounding sub-SAD, but it has two drawbacks. One is that the approach in [12] has to estimate four integer-pel positions twice, while saving only one half-pel position. The other is that all predicted directions of forty-one MVs are imposed by the predicted direction of a 16×16 block, which causes inaccurate prediction of fractional candidates. The authors in [13] propose one-step algorithm by reusing half-pel SATD for calculating approximate value of quarter-pel SATD. This method is based on assumption that rounding function and Hadamard transform function are linear ones. As a consequence, it may cause performance degradation significantly. Several fast FME algorithms in [14,15] are proposed to speed up the FME processing. However, these software-oriented algorithms are not suitable for hardware design due to irregular data flow and memory access manner. Our previous work [8] is more suitable for hardware implementation with one-step algorithm. It is a straightforward design with fixed twenty-one fractional-pel positions. Though this architecture can reduce the long latency of interpolation processing by half, it occupies quite a large amount of hardware area.

3. Proposed FME algorithm

3.1. One-step FME algorithm

3.1.1. Selection of predicted fractional motion vector

As earlier analysed, the predicted MV that is obtained from the median MV of the adjacent blocks on the left, top and top-right (or top-left) is not accurate enough in the large blocks. It also needs the MV of block on the left that