High Performance VLSI Architecture of Fractional Motion Estimation for H.264/AVC

Meihua GU¹, Ningmei YU¹†, Wenhua JIA¹, Wei LU¹

¹ School of Automation and Information Engineering, Xi’an University of Technology, Xi’an 710048, China
² School of Electronic Information, Xi’an Polytechnic University, Xi’an 710048, China

Abstract

Due to its high computational complexity, fractional motion estimation (FME) needs acceleration for real-time high-resolution applications. This paper proposes a high performance and cost efficient VLSI architecture for full-search FME in H.264/AVC. For half-pixel ME, 112 4×4 processing elements are processing in parallel, the included half-pixel interpolator is based on 4×4 block, and is realized with a novel efficient structure, 4×4 and 8×8 sum of absolute transformed difference (SATD) generator are adopted to realize transform adaptation in H.264 high profile algorithm. The architecture of quarter-pixel ME is similar to that of half-pixel ME, except that a novel table-loop-up method is adopted, a tricky address generator structure is designed for the quarter-pixel interpolation, and internal buffers and latency reduction can be attained. After logic synthesis using SMIC 0.13 μm standard cell library, under a clock frequency of 300 MHz, the FME architecture can encode a 1920×1080 (HDTV) video at 113fps with only 200k gate counts.

Keywords: H.264/AVC; Fractional Motion Estimation; Full Search; VLSI Architecture

1. Introduction

H.264/AVC [1] is the latest video coding standard jointly developed by the ITU-T Video Coding Experts Group and the ISO/IEC MPEG. For better coding efficiency, H.264/AVC adopts variable block size motion estimation (VBSME) with quarter-pixel accuracy. About 30% coding efficiency is increased by using quarter-pixel motion vector compared with half-pixel motion vector [2]. VBSME in H.264/AVC achieves superior coding efficiency at the expense of huge amount of computation and memory traffic. fractional motion estimation (FME) will occupy almost over 30% of the computation complexity of H.264 encoding process. Therefore, a hardware accelerator is indispensable for VBSME in high-end applications. So far, several full search FME architectures for the H.264/AVC have also been proposed. Reference [3] proposed an architecture with nine 4×4 processing units (PUs) for simultaneously processing nine candidates around the refinement center. The processing capability is enhanced from standard-definition television to 720HD [4]. Reference [5] presented an architecture for HD1080p video. This design adopts a short-latency 16-pixel-wide interpolator to increase throughput and eliminate redundant interpolation. However, a fourfold increase in data width only results in 1.67 times of throughput gain. Therefore, the hardware utilization is low when processing small-size blocks (4×4 and 4×8). Reference [6] proposed a

† Corresponding author.

Email addresses: yum@xaut.edu.cn (Ningmei YU).

1553-9105/ Copyright © 2011 Binary Information Press
August, 2011
high-performance FME hardware architecture, which consists of three parallel processing engines, one for 4×4 and 8×8 blocks, one for 8×4 and 4×8 blocks, and another for the remaining type of blocks. This design can encode a 1920×1080 video at 30fps with 321K gate when running at 154 MHz.

In the FME architecture proposed in this paper, 112 4×4 half-pixel PEs working in parallel can produce 112 4×4 residue blocks at the same time, half-pixel interpolator is included in each 4×4 half-pixel PE, and is in charge of the half-pixel interpolation of one 4×4 block, and keeping the 11×11 half-pixel inserted block in the registers for quarter-pixel ME re-use; 41 SATD comparators process 41 SATDs in parallel and renew the minor SATDs and the associated half-pixel MVs. 4×4 half-pixel PEs, SATD generators, and SATD comparators are processing in pipeline to improve the structure efficiency. In addition, a tricky address generator unit is adopted to provide the needed address for the table-loop-up pixels for the 4×4 quarter-pixel interpolation, this table-loop-up method can improve the quarter-pixel interpolating speed and reduce some hardware resources such as controls and registers. Experimental results reveal that the proposed architecture is able to support up to the high definition TV (HDTV) frame format at the frame rate 113 fps under SMIC 0.13 μm CMOS standard cell technology.

2. Fractional Motion Estimation with Full-search

After IME performs motion search to find an IMV for each of 41 sub-blocks, FME performs motion search around the center pointed to by IMV and further refines 41 IMVs into quarter-pixel precision. FME consists of two search flow: half-pixel search and quarter-pixel search. The half-pixel ME refinement is performed around the best integer-pixel search positions of 41 sub-blocks at all reference frames, and quarter-pixel ME, as well, is performed around the best half-pixel search positions. Nine positions are searched in both half refinement (one integer-pixel search center pointed to by IMV and eight half-pixel positions) and then quarter refinement (one half-pixel position and eight quarter-pixel positions). Half pixels do not exist in the original reference pictures and should be interpolated by the sample values of the nearest 6 integer pixels with a 6-tap FIR. Quarter pixels are interpolated by the nearest 2 half or integer pixels with a 2-tap bilinear filter.

To evaluate the cost at different points, the differences of MB's samples and reference samples (maybe the interpolated values) need to be transformed to obtain the sum of absolute transformed difference (SATD), which is commonly used as the main matching criterion.

\[ \text{diff}_T = H \ast \text{diff} \ast H, \quad SATD(i, j) = \left( \sum_{i,j} |\text{diff}_T(i, j)| \right) / 2 \]  

(1)

Where (i, j) is the one of the search candidate, and diff is the residue block between the candidate block and the current block, H is a symmetrical matrix, whose transposition is itself. The position with minimum residue cost (SATD) is chosen as the best match. All above procedures should be performed for each of 41 sub-blocks with different block size modes. This algorithm is also commonly employed in hardware implementation.

3. Architecture Design of FME

Fractional motion estimation procedure contains half-pixel refinement and quarter-pixel refinement, which are processed sequentially. The architecture of the two modules will be described in the following sections.
3.1. Half-pixel ME

Fig.1 shows the block diagram of half-pixel ME. There are 41 half-pixel ME modules to refine the 41 IMVs of the sub-blocks to half-pixel precision in parallel. The inputs are 41 IMVs, with which 41 reference search window can be determined. The output of this module are 41 half-pixel MVs and the 16×7=112 reference half-pixel inserted window data based on 4x4 primitive block.

![Fig.1 Block Diagram of Half-pixel ME](image)

The 41 half-pixel ME modules are in charge of the half-pixel refinement for each of the 41 IMVs. The proposed half-pixel ME architecture is based on 4×4 primitive processing element (PE), as shown in Fig.2(a). The number of this PE contained in each half-pixel ME module in Fig.1 is different according to the size of the processing sub-block. For example, since half-pixel ME module_0 is in charge of 16×16 block’s MV refinement, this module contains 16 4×4 sub-block processing units, and so forth. There are total 7×16×112 4×4 half-pixel PEs working in parallel. The block diagram of 4×4 half-pixel PE is shown in Fig.2(a). The input of this PE is the reference integer-pixel data already stored in the registers, and can be obtained according to the corresponding IMV, and the output is 4×4 residue block and 11×11 half-pixel inserted window data.

Half interpolator is used to realize the 6-tap FIR filter for the half-pixel interpolation of 4×4 block. The output half-pixel inserted reference data is the half-pixel searching window, and should be kept in the specified register file (RF) for the subsequent quarter-pixel interpolation.

The following half-pixel full search unit is responsible for checking the 9 candidate points, and output 9 4×4 residue blocks sequentially. The 4×4 half-pixel PE in the dashed box is the basic architecture in each of the half-pixel ME module. In our architecture, half-pixel ME module with the block size larger than or equal to 8×8 is based on 8×8 SATD, the others with the block size smaller than 8×8 is based on 4×4 SATD, 8×8 SATD and 4×4 SATD structures are shown as Fig.2(b) and Fig.2(c) respectively.

The last unit in the half-pixel ME module is SATD comparator, after performing 9 times, the minimum SATD and best half-mv(denoted as row and line) can be obtained. In each of the half-pixel ME module, 4×4 half-pixel PEs, SATD generator units, adder arrays, and SATD comparators are pipeline processing, and the full-search half-pixel ME can be completed within 20 cycles.
3.1.1. Half-pixel Interpolator

H.264 employs 6-tap FIR filter to implement half-pixel interpolation, to interpolate one $m \times n$ block to half-pixel precision, $(3+m+3) \times (3+n+3)$ integer pixels should be prepared, here $m=n=4$. Therefore, the input of this unit is $10 \times 10$ integer pixels, as shown in Fig.3. The shade square stands for the integer pixel, the $4 \times 4$ integer pixels in the center dark-gray dashed box is the best matching $4 \times 4$ integer reference block. Therefore, the 9 candidate search points are centered by the left-top integer pixel within the minimum square frame, and the 9 corresponding reference $4 \times 4$ half-pixel blocks are all within the second outer square frame. The numeric squares are half-pixels to be interpolated. The numbers stands for the processing order of the interpolation procedure. The half-pixels with numeric 1 in line 0 can be interpolated with the adjacent horizontal 6 integer pixels in the same line following (2), while the half-pixels with numeric 1 in the middle of line 2 and 3 are interpolated with the adjacent vertical 6 integer pixels in the corresponding column by (2). Then the half-pixels with numeric 2 can be interpolated simultaneously, and so forth. The interpolation of all the numeric half-pixels can be completed within 6 cycles.

To prepare for the latter quarter-pixel interpolation of the reference $4 \times 4$ block, the integer and half pixels inside the outermost box with the size of $11 \times 11 \times 8$ bits should be stored in the RF to improve data reuse efficiency.

$$half = round(E - 5F + 20G + 20H - 5I + J)/32$$ (2)

Since (2) can be rewritten as (3), half-pixel interpolator can be implemented with three-level 3-2 compressors and 3 adders, as shown in Fig.4.

$$half = round(E - F - 4F + 4G + 16G + 4H + 16H - I - 4I + J)/32$$

$$= round((E + J + 4G) + (4H + 16G + 16H) - (F + 4F + I) - 4I)/32$$ (3)
3.1.2. 4×4/8×8 SATD Generator

According to the computation characteristic of SATD, there are two modules in the SATD generator, one is 2-D 4×4(8×8) Hadamard transform unit, the other is summation unit, as shown in Fig.5. This architecture features in high parallelism and pipelining processing. 2-D Hadamard transform is implemented by 2-level 1-D Hadamard transforms. Each level contains 4 or 8 parallelized 1-D Hadamard units for the block size of 4×4 or 8×8, one 1-D Hadamard units is responsible for 1-D transform of one 4×1 or 8×1 data. The first level is in charge of realize the 1-D transform of each row data of the input residue data, the second level is in charge of the 1-D transform of each column data of the input data, which is the transposition of the output of the first level units. Here the register array between the two level 1-D transform units is responsible for the transposition. The transformed coefficients can be obtained synchronously, then transmitted to ABS units to get the 4×4 or 8×8 absolute value at the same time. SUM units are composed by multi 3-2 compressors, which can realize the summation of 8 data, and features in fast speed compared to the traditional adder. With the pipelining processing, 4×4 SATD architecture can process 16 pixels/cycle, and the 4×4 SATD value can be obtained within 6 cycles; 8×8 SATD architecture can process 64 pixels/cycle, and the 4×4 SATD value can be obtained within 8 cycles.

3.2. Quarter-pixel ME

After half-pixel refinement for the 41 sub-blocks’ MVs of the current MB, quarter-pixel refinement should be performed. Fig.6 shows the general block diagram of this module. 41 sub-blocks’ MVs refinement are
performed in parallel. With the input half-pixel MV, the quarter-pixel search region can be determined within the half-pixel inserted search window which is pre-stored in the RFs. Since the half-pixel inserted search window is based on 4×4 primitive block, and actual size of the window is 11×11 pixels. There are total 16×7=112 windows in the RFs.

The architecture of quarter-pixel ME module is also based on 4×4 primitive block, and is termed as 4×4 quarter-pixel process element(PE), as shown in Fig.7. With the half-pixel mv, the address of the best matching point in the half-pixel inserted search window for the included 4×4 primitive block can be determined, and the other 8 quarter-pixel search region can be specified. Address_generator is in charge of generating the 16 pairs of addresses, and with which 16 pairs of pixels can be obtained from the half-pixel inserted window and fed into the quarter-pixel interpolator to get the 4×4 quarter-pixel reference block, which is fed into the next unit to produce the 4×4 residue block. Since there are 9 quarter-pixel candidate search points (include one half-pixel center point), the above procedure should be performed 9 times sequentially.

There are total 112 4×4 quarter-pixel process elements (PEs) working in parallel, and the number of this PEs contained in each quarter-pixel ME module is different in accordance with the size of the processing sub-block. Similar to that of half-pixel ME module, the cost of sub-block with the size larger than or equal to 8×8 is based on 8×8 SATD, and the cost of other sub-block with the size smaller than 8×8 is based on 4×4 SATD.

### 3.2.1. Address Generator

According to the input h_mv, address_generator can specify the best half-pixel search center(h3) in the half-pixel inserted window, and can locate the other surrounding 8 candidate quarter-pixel search points, which do not exist in the search window, as shown in Fig.8, i1-i4 are the integer pixels, h1-h5 are the already inserted half-pixels, among which i4 is the best integer position IMV points to, and h3 is the best
location \( h_{mv} \) points to, and the q1-q8 are 8 candidate quarter-pixel points which are to be interpolated by the adjacent half or integer pixels following formulas in Fig.8.

When receiving one \( h_{mv} \), address_generator first generates the address of \( h_1 \), denoted as \((row0, line0)\), According to Fig.8, q1-q8 can be interpolated with the values of h1-h5, and one quarter-pixel is determined by 2 half-pixel values. The associated 9 pairs of address-combination for 9 pairs of quarter-pixels (q1-q8, h3) are abide by a tricky pattern rule, and can be realized by the structure as shown in Fig.9. According to the input \( h_{mv} \), the address of \( h_1 \) is generated and denoted as \((row0, line0)\), one of the output is the line of \( h_1, h_2, \) and \( h_3 \) sequentially, which is controlled by the control signal \( x \), data \((h_2-h_4)\) are the pixel data of \( h_2, h_3 \) and \( h_4 \), and which one can be output is up to \( x \), which is the left-shift parameter, when \( x=0 \), the output are line0 and the value of \( h_2 \), when \( x=1 \), the output are line0 and the value of \( h_3 \), when \( x=2 \), the output the value of \( h_4 \).

\[
q_1 = (h_1 + h_2 + 1) \gg 1, \quad q_5 = (h_3 + h_4 + 1) \gg 1 \\
q_2 = (h_1 + h_3 + 1) \gg 1, \quad q_6 = (h_2 + h_5 + 1) \gg 1 \\
q_3 = (h_1 + h_4 + 1) \gg 1, \quad q_7 = (h_3 + h_5 + 1) \gg 1 \\
q_4 = (h_2 + h_3 + 1) \gg 1, \quad q_8 = (h_4 + h_5 + 1) \gg 1
\]

Fig.8 Quarter-pixel Search Candidates

3.2.2. Quarter-pixel Interpolation

With 16 pairs of addresses, 16 pairs of reference integer or half pixels can be obtained with table-loop-up method from the half-pixel inserted window, and fed into quarter-pixel interpolator, whose architecture is shown in Fig.10. There are 16 groups of 2-tap bilinear filters in this unit, each filter employs one adder and disposes the right binary bit to generate one quarter-pixel value as Fig.8 does. These 16 filters are processing in parallel, and the 16 interpolated quarter pixels can be obtained simultaneously.

4. Experimental Results

The proposed VBSME architecture is described and verified with Verilog-HDL in VSC environment and synthesized by Synopsis Design Compiler using SMIC 0.13 \( \mu \)m CMOS standard cell library. The implementation results in table 1 shows the comparison of our design and previous fractional ME circuits.
The proposed FME design shows better performances in every aspects than reference [6]. This design consumes 360 cycles for one MB in average, and can encode a 1920×1080 113-frames-per-second (fps) video when running at 300 MHz. From the dynamic simulation, the whole design of FME can provide processing capacity up to 833k MB/sec which is enough for 1080HD (1920×1080@30fps:245k MB/sec) real-time video systems.

<table>
<thead>
<tr>
<th>Table 1 Performance Comparison of the H.264/AVC FME Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference[6]</td>
</tr>
<tr>
<td>Process</td>
</tr>
<tr>
<td>Gate counts</td>
</tr>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>Throughput</td>
</tr>
<tr>
<td>Cycles/MB</td>
</tr>
</tbody>
</table>

5. Conclusion

This paper presents memory-efficient and highly parallel VLSI architecture for fractional-pixel ME with FS-VBS in H.264, which is based on 4×4 half-pixel PE and 4×4 quarter-pixel PE. Half-pixel and quarter-pixel interpolator based on 4×4 primitive block are designed sequentially to reduce memory size. 3-level 3-2 compressor is adopted for the realization of half interpolator, a tricky address generator and table-loop-up methods are adopted in the implementation of quarter interpolator. With the parallel processing and pipelining techniques techniques, the whole architecture of FME allows the real-time processing of 1920×1080 (HDTV) video at 113 fps.

Acknowledgement

This work is supported by Special Scientific Research Program of Shaanxi Provincial Education Bureau (No.2010JK558) and Special Funds of Shaanxi Major Subjects Construction (No.107080903).

References