AN EFFICIENT VLSI ARCHITECTURE FOR MC INTERPOLATION IN AVC VIDEO CODING

Deng Lei[1], Gao Wen[2] Hu MingZeng[1], Ji Zhen Zhou[1]

[1]Department of Computer Science and Engineering
Harbin Institute of Technology, Harbin 150001, china
[2]Institute of Computing Technology, Chinese Academy of Science,
Beijing, 100080, China
E-Mail: ldeng@jdl.ac.cn

ABSTRACT

Advance Video Coding (AVC) has employed a 6-tap interpolation FIR filter in its motion compensation (MC) part for high coding efficiency. But it is accompanied by increasing the complexity in calculation and the number of memory access. And this problem makes MC one of the bottlenecks in the AVC system's VLSI implementation, especially for SDTV/HDTV which aggravate the problem heavily. Unfortunately, most FIR filter [8][9][10] have too low of input bandwidth to deal with it. In this paper, an efficient architecture for MC interpolation is described, and experimental results show that this architecture satisfies AVC decoder applications such as SDTV or HDTV.

1. INTRODUCTION

AVC standard also known as MPEG-4 Part 10 and H.264, is jointly developed by ISO and ITU-T—Joint Video Team (JVT). Compared with previous video coding standards like MPEG-2 and H.263, AVC obtains higher coding efficiency [1][2] with advanced features and functionality like different block sizes, lagrangian coder control, multiple reference frames, fractional pixel precision, etc, at an increased implementation cost. The partition of the AVC's inter-coded block can be one of seven types identified by lumina block sizes: 16×16, 16 ×8, 8×16, 8×8, 8×4, 4×8, 4×4. MC interpolation of the AVC employs a 6-tap FIR filter that needs the samples not only inside the current block but also outside it. This surely requires more memory bandwidth. The amount of memory access for MC interpolation is about 50% in the AVC decoder [3] [4]. And the time consumed by interpolation processing is about 25% in the AVC decoder major subsystems [7]. So MC becomes one of the most data intensive parts of the AVC decoder, and a bottleneck of implementation. The memory access and the high transfer rate are the main troubles in the VLSI design of the MC interpolation.

According to the limit of AVC level-4 [11] which is SDTV or HDTV application level, the maximum macro-block (MB) rate is 245760 MB/s (30 frame/s and 8192 MB in a picture). If using 150MHz clock, the time budget is only 610 clock cycles assigned for the processing of one MB which has four luma blocks and two chroma blocks. And only slightly more than 100 cycles for a single block averagely.

Realization of FIR filters can vary widely from one that uses dedicated hardware multipliers and adders [8][9] to one that uses code executed by a general purpose processor. A combination of hardware and software that allows sharing of hardware units like adders and multipliers can also be used [10]. But these FIR filters don't fulfil the requirement of the interpolation in this paper. Extra clock cycles are required by mentioned FIR filters above between the processing for adjacent row or column of MB to displace the MB data inside their architectures. Extra clock cycles lead to the loss of processing time significantly, and cut down the filter efficiency. Moreover, these FIR filters have so insufficient data input bandwidth that they only allow to input one pixel data per cycle. For a 8×8 interpolated outcome the algorithm or the filters require to deal with 169 input pixel data [11], that means at least 169 cycles should be spent here. Thinking about the time budget of a single block—100 cycles, these FIR filters are so time-consuming that they can not meet the requirement. After analyzing their architecture thoroughly, we found that most of these FIR filters only have one-dimension data transfer way. But MC interpolation needs two-dimension MB data for calculation. So these FIR filters have low behavior on this interpolation.

The data transfer scheme adopted by The *AB2 type* architecture [5][6] for motion estimation in video encoder is very similar to the requirement of interpolation in this

paper. Compared with [6], Architecture of [5] reduced the hardware demands, and was implemented more economically.

The main goal presented in this paper is to arrange the MB data transfer properly and exploit the large bandwidth and high parallel architecture for MC interpolation of AVC decoder applications with larger frame sizes such as SDTV and HDTV. In section 2 of this paper, mc interpolation algorithm is described. Section 3 describes details of the implementation for the architecture of mc 6-tap interpolation. Experimental results are given in Section 4. The paper closes with a conclusion in Section5.

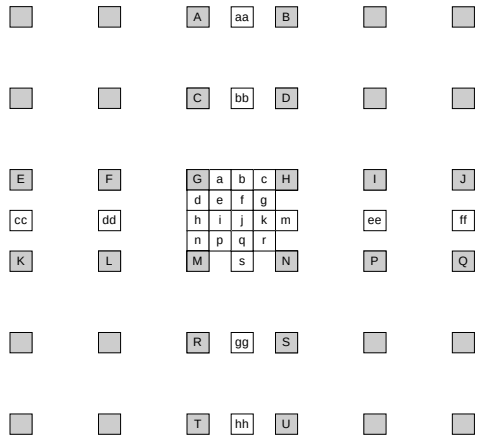

Fig.1

2. MC 6-TAP INTERPOLATION ALGORITHM

The aim of interpolation is to get the fractional samples from the integer samples according to the motion vector. In the AVC coding standard, the motion vector can point to the quarter-pel-accuracy location by the last two bits. Fig.1 shows the positions of integer samples (Squares with uppercase letters) and fractional samples (Squares with lowercase letters) for MC interpolation inside the given two-dimensional sample array. In Fig.1 Squares marked 'b', 'h', 'j', 'm' and 's' are the same as those labeled with double lowercase letters such as 'aa' which are positions at half-pel-accuracy location. And those others left and labeled with single lowercase letter are quarter-pel-accuracy locations. According to different locations pointed by motion vectors, fractional samples have different interpolation processing. Here, this processing is classified by five cases described below.

Case1: this is a special case, if the motion vector point to the integer sample of 'G', no processing needs to do here. And 'G' is directly output.

Case2: this case includes 'b', 'h', 's' and 'm'. And a 6-tap interpolation filter (1, -5, 20, 20, -5, 1) needs as:

$z' = x_{-2} - 5x_{-1} + 20x_0 + 20x_1 - 5x_2 + x_3$, and

$z = clip1((z'+16) >> 5)$.

Where z` represents intermedia variables of 'b``', 'h``', 's``' or 'm``', and z represents 'b', 'h', 's' or 'm'. The subscripts of *x* index horizontal or vertical neighbouring integer-pixel locations. "*clip1*" stands for clipping between[0, 255]. They shall be available after one 6-tap filter execution time.

Case3: this case includes 'a', 'c', 'd', 'n', 'e', 'g', 'p', 'r', which at least use one filtered half sample such as 'b' in *case2* (see table 1). And also shall be derived after one 6-tap filter execution time.

Case4: only 'j' belongs to this case. 'j' shall be obtained as follow:

$j' = aa - 5bb + 20b' + 20s' - 5gg + hh$, and

$j = clip1((j` + 512) >> 10)$

Where, variables of 'aa', 'bb', 'b``', 's``', 'gg' and 'hh' are available in the same manner of the intermedia variable 'z``' in *case2*. Because both 'b``' and 'j``' are derived by 6-tap filter, 'j' shall be obtained after two serial 6-tap filter execution times. So if 'j' is the final result of the interpolation, the latency of hardware shall be twice as much as 'b' in *case2*.

Case3		Case5	
a	(G + b + 1) >> 1	f	(b + j + 1)>>1
c	(H + b + 1) >> 1		
d	(G + h + 1) >> 1	i	(h + j + 1)>>1
n	(M + h + 1) >> 1		
e	(b + h + 1) >> 1	k	(m + j +1)>>1
g	(b + m + 1) >> 1		
p	(h + s + 1) >> 1	q	(s + j + 1)>>1
r	(m + s + 1) >> 1		

Table 1

Case5: this case includes 'f', 'i', 'k', and 'q', which use the half sample 'j' in *case4*. And so they shall be derived after 'j' is done (table1).

Moreover, from the description above, it is clear to see the relations among the five cases. Firstly, fraction samples in *case3* and *case4* depend on those in *case2*, and also *case5* depend on *case4*. Secondly, fraction samples in c*ase5* and *case4* have near the same interpolating complexity as twice much as those in *case2* and *case3*.these relations are very useful to optimize the design of the interpolation architecture.

3. THE ARCHITECTURE OF 6-TAP INTERPOLATION FOR MC

The data transfer scheme of the *AB2 type* architecture [5][6] is very suitable for the two-dimension data transfer process such as the MC interpolation. Base on it, the new architecture is devised and showed as Fig.2. In the architecture the 'N' and 'M' is decided by the partition of the MB. Two parts are included in this architecture. One

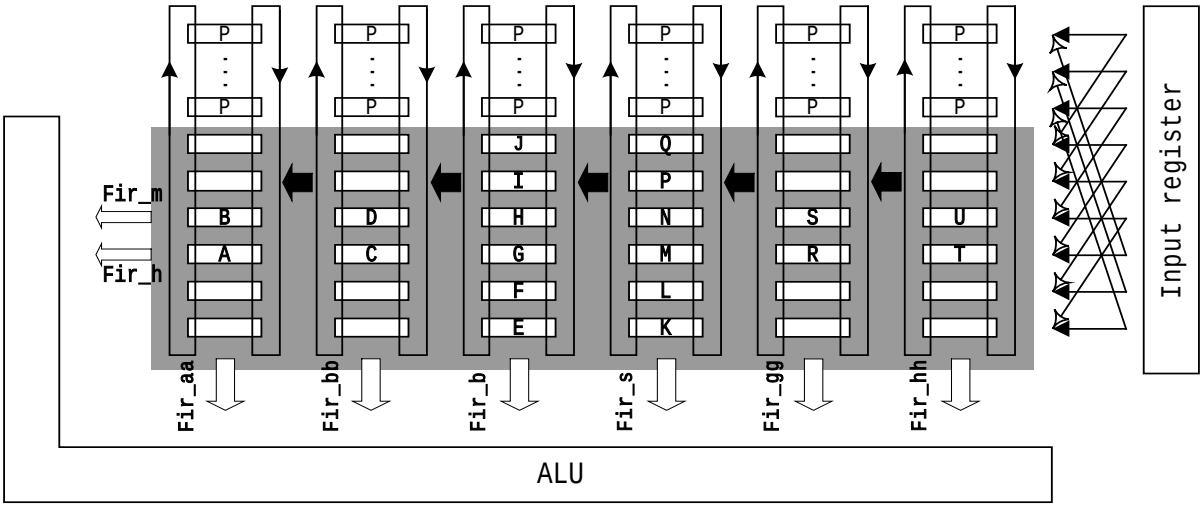

Fig.2 interpolation architecture of N×M block

is the part of the pixel data transfer, and the other is the part of ALU.

3.1 Pixel Data Transfer Scheme

The scheme has a register array which has N+5 row and 6 column registers used to reserve the data for the current or later processing. There are two type registers in the array, one is the active pixel register (A_{ij} register) which serves the current processing. And the other is the passive pixel register (P register) which stores the data for the later processing. Active rectangle signed by the shadow area in Fig.2 has 6×6 active pixel registers used to hold the MB samples for the current calculation of 1/2 or 1/4 interpolation. Passive rectangle composed of (N-1)×6 passive pixel registers is used to buffer the pixel data for later calculation. The data transfer in three ways: upwards, downwards and to the left.

*To the left :*one column with N+5 pixels is shifted into the most right side of the array through the set of input registers, at the same time each pixel in the array is shifted one position to the left.

Downwards: pixels are shifted downward (in Fig.2) one position per cycle.

Upwards: pixels are shifted upward (in Fig.2) one position per cycle.

For a N×M block partition, the pixels transfer scheme is presented as below:

Initial data in the buffer: 6 operations of "to the left"

For (I= 0; I <=M/2; I++){

N-1 operations of "downwards";(for achieving results of an even line interpolation)

1 operations of "to the left";

N-1 operations of "upwards"; (for achieving results of an odd line interpolation)

1 operations of "to the left";

}

So total cycles needed by interpolating a partition of N×M is:

$cycle_{N\times M}=6\times cycle_{left} + (N-1+cycle_{left})\times M + cycle_{delay}$(1)

Where $cycle_{left}$ represents the cycles of the operation "to the left", and $cycle_{delay}$ represents the cycles delayed by ALU showed in Fig.3.

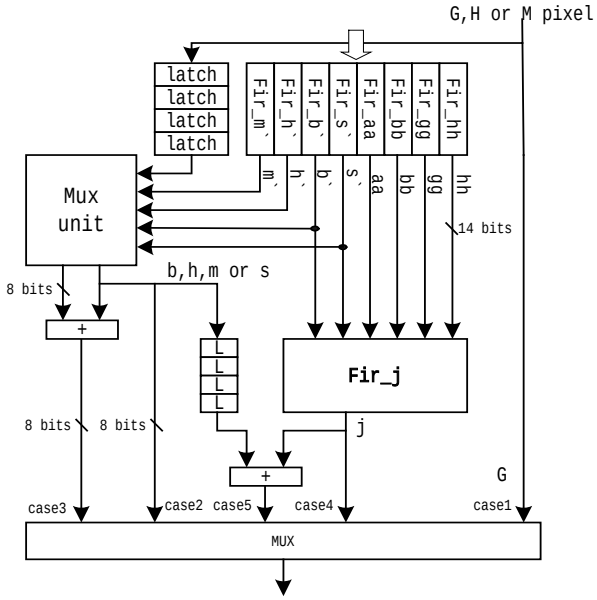

Fig.3 the ALU

3.2 the ALU

The ALU (in Fig.3) is responsible for the calculation of the fractional samples. According section 2, these samples are classified by five cases. The ALU has Night filters in order to have the parallelization in all conditions of

motion vector. Block named “Fir_x” is the special used filter for processing the fractional sample ‘x’. And the architecture of “Fir_x” is shown in Fig.4 in which an adder tree is adopted instead of the multiplication. The data for calculating a certain fractional sample are derived directly from the active rectangle in Fig.2 simultaneously and inputted into the ALU at the same time also. In Fig.3, Some full samples or half samples must be delayed for matching the latency of filters, such as ’G’ for calculating ‘a’ and ‘h’ for ‘i’. The function of *MUX UNIT* is to select the output pixels for later processing according to motion vector when result belongs to case2, case3 or case5. The operation of “*clip1”* is in the *MUX UNIT*, and applied before pixels are exported.

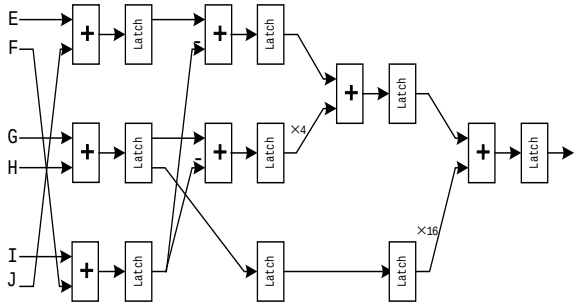

Fig.4 the architecture of filter

4. EXPERIMENTAL RESULTS

Since the whole AVC decoder pipeline is based on 8×8 blocks, the 8×8 block partition is adopted in the implementation of the architecture in Fig.2. However, by control, the implementation can fit all the partitions.

The implementation was described using Verilog-HDL, and synthesized with synopsys tools using 0.25um Standard Cell Library. The total area is about 287475.875 um^2. The $cycle_{delay}$ of the formula (1) in the ALU is eight cycles at most and the $cycle_{left}$ is one cycle in the implementation. So the maximum number of the cycles in 8×8 partition is 78. And the critical path of the architecture is 4.84*ns* which may satisfy the decoder applications such as SDTV or HDTV.

5. CONCLUSION

An efficient architecture for MC interpolation of AVC is proposed in this paper. The data transfer scheme of this architecture solves the problem of memory access in MC perfectly. The experimental results show that the proposed architecture can meet the need for the real-time implementation of AVC decoder for SDTV or HDTV.

6. REFERENCE

[1] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, G. Sullivan, “Performance comparison of video coding standards using Lagrangian coder control” in *Proc. International Conference on Image Processing*, Vol. 2, pp: 501–504, 2002.

[2] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, G.J. Sullivan, “Rate-constrained coder control and comparison of video coding standards” *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 13, pp: 688–703, July 2003.

[3] K. Denolf, C. Blanch, G. Lafruit, J. Bormans, “initial memory complexity analysis of the avc codec” *IEEE Workshop on Signal Processing Systems (SIPS)*, pp: 222–227, 2002.

[4] K. Sato, Y. ragasaki, “Adaptive mc interpolation for memory access reduction in JVT video coding” in *Proc. Seventh International Symposium on Signal Processing and Its Applications*, Vol. 1, pp: 77 –80, July 1-4, 2003.

[5] Nuno Roma, Leonel Sousa: “A New Efficient VLSI Architecture for Full Search Block Matching Motion Estimation” *VLSI-SOC*, pp:253-264, 2001: Montpellier, France.

[6] L. vos and M. Stegherr, “Parameterizable VLSI Architectures for the full-search Block-Matching Algorithm”, *IEEE Transactions on Circuits and Systems*, 36(10):1309-1316, October 1989.

[7] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, “H.264/AVC baseline profile decoder complexity analysis” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.13 Issue: 7, pp: 704 –716, July 2003.

[8] J. Park, K. Muhammad and K. Roy, “High Performance FIR Filter Design Based on Sharing Multiplication,” *IEEE Transactions on VLSI Systems (TVLSI)*, Vol.11, Issue: 2, pp: 244-253, April 2003.

[9] H. Samueli, “On the design of optimal equiripple FIR digital filters for data transmission applications” *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1542-1546, Dec 1988.

[10] N. Sankarayya, K. Roy, and D. Bhattacharya, “Algorithms for low power and high speed FIR filter realization using differential coefficients” *IEEE Trans. Circuits Syst.*, vol. 44, pp. 488-497, June 1997.

[11]JVT:ISO/IEC and ITU-T, “Draft ITU-T Recommendation and Final Draft international Standard of Joint Video Specification”, Doc.JVT-G050r1, Geneva, Switzerland, May, 2003.