Design of an MC Interpolation Architecture for H.264 Video Decoder

Yi-Shiung Jang Kuen-Cheng Chiang Jean Jyh-Jiun Shann

Department of Computer Science and Information Engineering,
College of Electrical Engineering and Computer Science,
National Chiao Tung University,
Hsinchu, Taiwan, R.O.C.

changyihs.iic94g@nctu.edu.tw kcchiang@csie.nctu.edu.tw jjshann@cs.nctu.edu.tw

Abstract*-H.264/AVC is the newest video coding standard. Compared with MPEG-2, MPEG-4, and H.263, H.264/AVC provides highest coding efficiency and better video quality. Motion compensation is one of the techniques for enhancement video quality in video decoder standard that includes 6-tap FIR filter and 1/4-pel precision for luminance and 1/8-pel precision for chrominance. In H.264/AVC decoder, the motion compensation consumed 39% decoding time, which is the most important influence of decoding system because of its complex computation. For this reason, in this paper, we proposed an efficient motion compensation interpolation architecture for reducing computational complexity in H.264/AVC decoder. Since the luminance computation occupies 80% of the whole motion compensation interpolation computation, consequently, we propose two algorithms to reduce the computational complexity for luminance component — the 6-tap Mean Filter, which applies 6-tap FIR filter method and Mean concept, and the 4-tap Mean Filter, which applies 4-tap FIR filter method and Mean concept. Furthermore, we propose a novel data supply mechanism, called Snake Path, in order to save hardware area. Presently, we adopt parallel and pipelined architecture for our proposed method in hardware implementation. Through the software simulation, the reconstructed video quality shows approximate image quality compare with traditional 6-tap FIR filter adopting our proposed 6-tap Mean Filter algorithm. Our designs are synthesized with TSMC 0.13 μm technology. The synthesized results show that two proposed algorithms with Snake Path control can save hardware cost 48% and 68%, respectively.*

Keywords: Motion Compensation, H.264/AVC, FIR Filter, MC Interpolation, Snake Path

1. Introduction

To come up against the rapidly growing demands of multimedia applications, many generations of video encoding/decoding standards, such as MPGE-1, MPEG-2 and MPEG-4, swere developed by the ISO/IEC moving picture experts groups. For providing a video stream in higher quality via limited transmission resources, a better video coding standard, H.264/AVC, was jointly developed by ISO/IEC and International Telecommunications Union - Telecommunications Standardization Sector (ITU-T) with higher compression efficiency in stream bit-rate and better video quality. Compared with MPEG-4 advanced simple profile and H.263 high latency profile, H.264/AVC saved about 37% and 48% bit-rate with the same quality of video, respectively [1].

H.264/AVC is a high efficiency coding standard based on a motion compensated hybrid Discrete Cosine Transform algorithm. To achieve the requirements of high quality and low bit rate, it adopts many advance and precision coding skills. The coding process of H.263/AVC consists of the following major tasks: Inversed Integer Discrete Cosine Transform, Context Adaptive Binary Arithmetic De-Coding, Context Adaptive Variable Length De-Coding, De-blocking Filter, Variable Block Size Motion Compensation (MC), and Quarter-pixel Precision Motion Vector [2, 3]. With deeper resolution of image reconstruction into level of Quarter Pixel, the computational complexity of MC interpolation is increased compared with previous video standards [4,5]. Lappalainen et al. [6] analyzed the execution behaviors of H.264 coding and decoding process and indicated that the decoder expanded large portion of computing power, over 39%, in performing the MC interpolation. Obviously the highly demanded computation resources for MC interpolation dominated performance of entire H.264/AVC decoder.

In this paper, we proposed two filtering algorithms and designed associated hardware with effective parallel computation mechanism for MC interpolation in H.264/AVC decoder.

2. Background and Related Work

The key technique of motion compensation interpolation is based on n-tap FIR filtering for evaluating the value of quarter pixel to improve the quality of image with higher resolution. Before discussing computation behaviors for the interpolation in detail, we first explained a general executing flow of H.264/AVC decoder briefly. Since the interpolation process for both luminance (luma) and chrominance (chroma) components was very similar, we illustrated the process only for luma component as an example.

2.1 Basic concept of H.264/AVC decoder

The processing steps of a H.264/AVC decoder is consisted of Entropy Decoder (ED) , Reorder (RO), Inverse Scan and Quantization (IS/IQ), Inverse Discrete Cosine Transform (IDCT), Deblocking Filter (DBF), Intra Prediction (IP) and Motion Compensation (MC), as shown in Figure 1. The encoded bitstream comes from a broadcast system or video file source in a variable length coding style. After ED processed the bitstream, motion vectors and header information are isolated for MC and IP steps. One the other hand, the pixel data is sent to the RO, IS/IQ, and IDCT one by one to re-generate the image pixels in macroblocks. Based on motion vectors and header information, the MC and IP properly reconstruct the original pictures with one or more reference pictures. For a better video quality, a DBF is employed to remove blocking effect cause by DCT and Quantization. Finally, the resulting frame picture is shown in front of user via display devices.

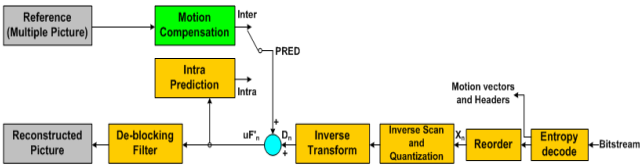

Figure 1. General Structure of a H.264 Decoder

H.264/AVC is a macroblock-based system adopted a tree structure motion compensation method. Unlike previous standards, such as MPEG-4 and H.263, H.264/AVC supports variable motion compensation block sizes which ranged from 4x4 to 16x16, and fine sub-pixel sample motion vectors [8]. Each macroblock and sub-macroblock partition in an inter-mode is predicted base on one or more previously encode video frames via motion estimation. In decoder, the predicted samples are generated according the motion vectors through motion compensation and according the residual data to produce the reconstructed macroblocks. Most of computations are executed on the sub-pixel sample interpolation process and the process was the most time consumed step of the decoder steps.

2.1.1. Sub-pixel Sample Interpolation

The execution of sub-pixel sample interpolation is based on a 6-tap FIR filter with six filtering weights (1, -5, 20, 20, -5, 1) [9]. Figure 2 shows the locations of integer-pixel, sub-pixel and quarter-pixel samples in the luma component interpolation scheme. The integer-pixel samples in the figure are labeled as *A1-A6*, *B1-B6*, *C1-C6*, *D1-D6*, *E1-E6* and *F1-F6*. The outer sub-pixel samples, b, h, s and m, can be derived by applying 6-tap FIR filter using integer-pixel samples as inputs. For example, the horizontal sub-pixel *b* is computed according *A3*, *B3*, *C3*, *D3*, *E3* and *F3* by applying equation (2.1). The final prediction values b is derived through the Clip operation that clips the resulting value between [0, 255]. In the same manner, other sub-pixel samples *s*, *h* and *m* can be generated by using integer-pixel samples located in the same directions.

$$b_1 = (E3 - 5\times F3 + 20\times A3 + 20\times B3 - 5\times C3 + D3)$$
$$b = Clip((b_1 + 16) >> 5) \quad (2.1)$$

The inner sub-pixel sample *j* is derived by calculating the intermediate value denoted as j_1 first by applying the 6-tap FIR filter to the intermediate values of the adjacent six half sample positions in either the horizontal or vertical direction, as shown in (2.2).

$$j_1 = (cc - 5\times dd + 20\times h_1 + 20\times m_1 - 5\times ee + ff)$$
$$j_1 = (aa - 5\times bb + 20\times b_1 + 20\times s_1 - 5\times gg + hh)$$
$$j = Clip((j_1 + 512) >> 10) \quad (2.2)$$

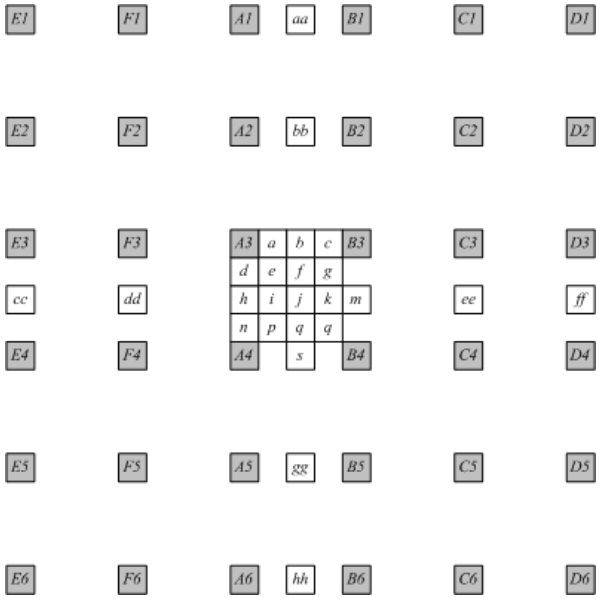

Figure 2. Integer-pixel and Sub-pixel Sample Locations in Luma Component Interpolation Scheme

2.1.2. Quarter-pixel Sample Interpolation

The luma values at quarter sample positions can be derived by averaging one full nearest sample and one half nearest sample [10]. For example, the quarter-pixel *a* is computed by *A3* and *b* as shown in equation (2.3). By averaging with upward rounding of the two neighboring samples located at integer and half sample positions, the samples located at quarter sample positions labeled as *a*, *c*, *d*, *n*, *f*, *i*, *k*, and *q* are calculated by applying the same formula.

$$a = (A3 + b + 1) >> 1 \quad (2.3)$$

By averaging with upward rounding of the two neighboring samples locate at half sample positions in the diagonal direction, the samples locate at quarter sample positions labeled as *e*, *g*, *p*, and *r* can be derived by applying equation (2.4).

$$e = (b + h + 1) >> 1 \quad (2.4)$$

2.2 Related Works

For the execution flow of H.264/AVC decoder, many designs were proposed to improve the performance of the process. Wang and Lie were designed new architectures for executing interpolation and computational processes

efficiently, respectively.

2.2.1. Interpolation Architecture [11]

Wang use multi-stage sub-pixel interpolation architecture for luma component. The sub-pixel interpolation architecture is composed of reference data buffer and reference data feeding architecture. 9x9 pixels valid reference data is stored into reference data buffer in order to correspond with one 4x4 luma data interpolation. The reference data feeding architecture composed of four parts as show in Figure 3. According to the experimental result and our observation shows the multi-stage sub-pixel interpolation architecture is realized with traditional 6-tap FIR filter may consume more hardware cost.

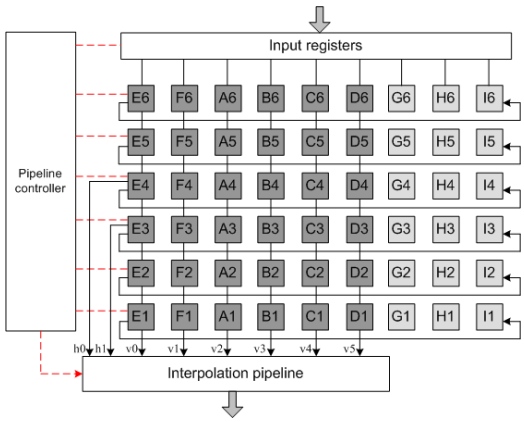

Figure 3. Reference Data Feeding Architecture

2.2.2. Computational Architecture [12]

Lie proposed an alternative design using 4-tap diagonal FIR filter for MC interpolation in luma component. The luma values at half-pixel sample positions are derived by applying a 4-tap FIR filter with filter weights (-1, 5, 5, -1). Noticeable, the samples located at half sample position labeled as *j* can be derived by using diagonal integer pixels *C2*, *B3*, *A4*, and *F5*. Compared with traditional 6-tap FIR filter, Lie's algorithm has disadvantage on quality degradation in image PSNR performance because of the mismatched filtering structure between encoder and decoder. According the software simulation results, the reconstructed frames may cause the sharpen effect, as shown in the following. Figure 4 (a) is an original CIF (352x288) frame of News and Figure 4 (b) is using 4-tap diagonal filter FIR interpolation frame. Obviously, the newscaster's face is sharpened of reconstructed frame compared with original frame.

(a) Original (b) 4-tap Diagonal Filter

Figure 4. Original Frame and 4-tap Diagonal Filter Interpolation Frame

3. Design

Under considerations of hardware implementation, we employed a parallel and pipelined architecture to implement the MC interpolation. The architecture, as shown in Figure 5, is consisted of Reference Data Buffer (RDB), Input Data Array (IDA), Pipeline Controller (PC), and Fractional Interpolation Pipeline Architecture (FIPA).

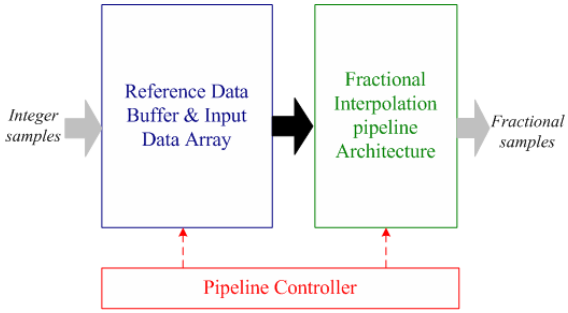

Figure 5. Motion Compensation Interpolation Architecture

3.1. Proposed Algorithms for MC Interpolation

We proposed two algorithms, STM Filter and FTM Filter, to retain the reconstructed video with the same level of quality as the video processed by traditional 6-tap FIR filter and moreover to improve the computational efficiency. The concept of STM filter is applying a 6-tap FIR filter method and Mean concept. The luma values at half-pixel sample positions can be derived by applying a 6-tap FIR filter with filter weights (1, -5, 20, 20, -5, 1) and by averaging four nearest samples at half sample positions. The inner half sample position labeled as *j* in Figure 2 is derived by averaging with upward rounding of the four closest samples locate at half sample positions as shown in (3.1).

$$j = ((b + h + s + m + 2) >> 2) \quad (3.1)$$

The STM filter, on the other hand, applies a 4-tap FIR filter method and Mean concept. The luma values at half-pixel sample positions can be derived by applying a 4-tap FIR filter with filter weights (-1, 5, 5, -1) and averaging four neighboring samples at half sample positions.

3.2. Reference Data Buffer and Input Data Array

In order to achieve the parallel requirements, we divided the reference data into two parts, the reference data buffer store the reference data for a 4x4 luma block interpolation and the input data array store the necessary integer pixels for fractional interpolation pipeline architecture, as shown in Figure 6. According the formula of 6-tap FIR filter, the filter requires a larger area of pixel data for interpolating a 4x4 block area. Two-pixel extension on the left and top side of the 4x4 block and three-pixel extensions on both right and bottom sides are required. Therefore, the hardware needed to buffering sixty-five neighboring pixels in registers for possible references. In order to handle the frame whose boundaries or size are not a multiple of four or eight, we divided the large block size into multiple 4x4 blocks, and then the frame edges are filled with the duplicates boundary pixels to receive straight reference data in the filtering [13].

The FTM filter method is based on 4-tap FIR filter, when enforced a 4x4 block interpolation, the original 4x4 block needs to plus one-pixel extension to the left and top and 2-pixel extension to the right and bottom except 4 pixels of four corners. Consequently, the hardware needs to buffer forty-five neighboring pixels in registers for possible references. Similarly, we used input data array between reference data buffer and fractional interpolation pipeline architecture.

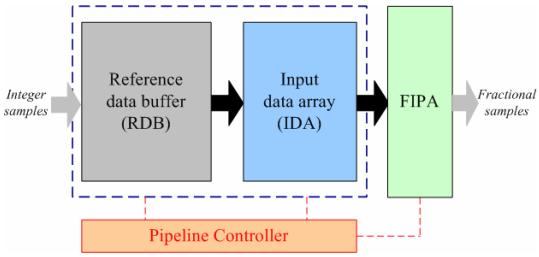

Figure 6. RDB and IDA of MC Interpolation Architecture

3.3. Pipeline Controller of MC Interpolation Architecture

We proposed a new concept for pipeline controller, which controlled and managed the data updating procedure, called Snake Path, thereinafter abbreviation S Path. In STM filter method, as show in Figure 7, the blue dot-real rectangle represents S Path proceeding. That consists of two streams, the red dotted line represents control stream, and black arrow represents data stream that includes three parts, the data input from reference data buffer by three directions, the inner data stream of input data array, and output data to FIPA by horizontally directions (second and third row) and vertically directions (second and third columns). In FTM filter method, because of using 4-tap FIR filter for fractional sample interpolation, only forty-five pixels reference data and twelve pixels reference data is supplied for RDB and IDA, respectively. The structure of Snake Path for FTM filter shows in Figure 8.

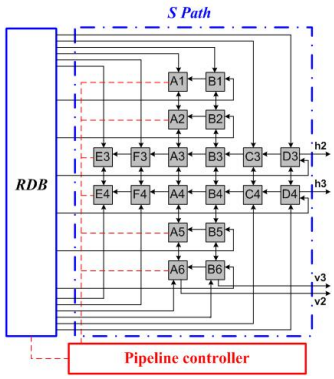

Figure 7. Structure of Snake Path for STM Filter

The S path of STM filter method can be divided into three portions: Shift-up, Shift-left and Shift-down. In FTM filter method, the operations are the same as STM filter method except the reference data buffer with forty-five integer pixels and twelve integer pixels input data array.

3.4. Fractional Interpolation Pipeline Architecture (FIPA)

To achieve design simplicity, we chose pipeline architecture to implement the fractional interpolation. For pipeline interpolation architecture, we divide the fractional interpolation process into three stages: Half-level, Center half-level and Quarter-level.

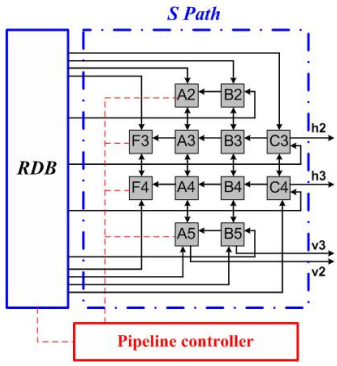

Figure 8. Structure of Snake Path for FTM Filter

The block diagram of fractional interpolation pipeline architecture was shown in Figure 9. First, "Half-level" handles two vertical and on second and third row column and two horizontal on second and third row of input data array by the SixTapFilterArray or FourTapFilterArray in parallel; middle values of half-level future deliver to "Center half-level" to enforce the intermediate half-pixel filter by the MeanFilter; afterward, if the goal is the quarter-pixel sample location then two half-pixel samples or one half-pixel and one integer-pixel sample are filtered by BilinearFilter. According to current motion vector, the multiplexer choice which one of the samples on integer-pixel sample location or sub-pixel sample location output to next stage. The input data array, pipeline controller, and PelOut are also included to compose of a complete pipelined architecture. The detailed design of each stage except BilinearFilter in the architecture is explained in the following paragraphs. As for the design of BilinearFilter, it is the same as that of the bilinear filter in the related work.

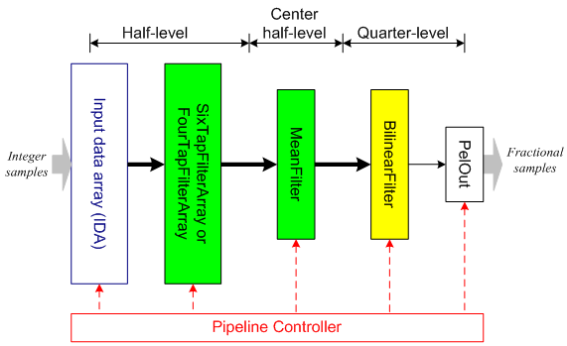

Figure 9. Block Diagram of Fractional Interpolation Pipeline Architecture

3.5. Architecture of SixTapFilterArray

In STM filter design, we divide the half-pixel interpolation process into two parts, SixTapFilterArray and MeanFilter. For the half-level stage, we combine four 6-tap FIR filter into a SixTapFilterArray to correspond with half-pixel samples interpolation in horizontal and vertical direction. Figure 10 shows that the architecture of

SixTapFilterArray, the input reference data is loading from input data array, *E3* to *D3* and *E4* to *D4* are loading from second and third row; *B1* to *B6*, and *A1* to *A6* are loading from second and third column, and output two horizontal (*b*, *s*) and vertical (*m*, *h*) half-pixel samples to next stage.

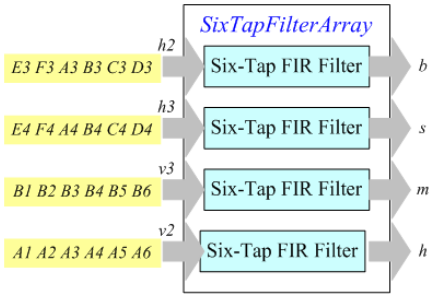

Figure 10. Architecture of SixTapFilterArray

3.6. Architecture of FourTapFilterArray

In FTM filter design, we also divide the half-pixel interpolation process into two parts, FourTapFilterArray and MeanFilter. We integrate four 4-tap FIR filter to build a FourTapFilterArray in order to correspond with half-pixel sample interpolation in horizontal and vertical direction of half-level stage. Figure 11 shows that the architecture of FourTapFilterArray, twelve integer samples are loaded from input data array, *F3* to *C3* and *F4* to *C4* are loaded from second and third row; *B2* to *B5* and *A2* to *A5* are loaded from second and third column, then output a 4x1 half-pixel array to next stage.

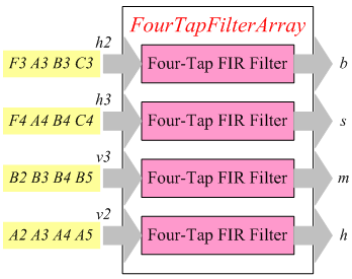

Figure 11. Architecture of FourTapFilterArray

3.7. Architecture of MeanFilter

The mean filter is used for intermediate half-pixel sample interpolation process on center-location. Figure 12 shows the related position of the reference and fractional interpolation pixels of mean filter. The gray blocks represent the sample on full-pixel location and green block represent the sample on half-pixel location. In our design, the center half-pixel interpolation process needs adjacent four half-pixel samples on horizontal and vertical location to produce one center sample on half-pixel location. The function representations of mean filter with one time filter coefficient and the regulative result can be written as equation (3.2). The architecture of mean filter with four adders and one shifter is shown in Figure 13.

$$j = ((b+s+m+h)+2) \div 4)$$
$$j = ((b+s+m+h)+2) \gg 2) \quad (3.2)$$

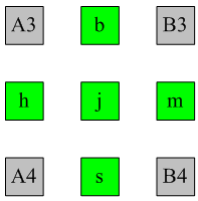

Figure 12. Related Pixels of Mean Filter

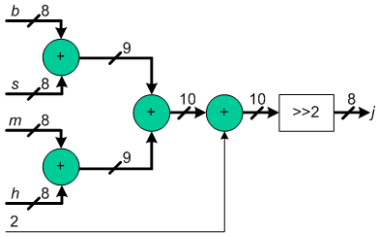

Figure 13. Architecture of Mean Filter

Our proposed methods use four 6-tap filters; each 6-tap filter with 7 adder (add.) and 3 shifter (shif.), or four 4-tap filters; each 4-tap filter with 5 adder and 2 shifter, and one mean filter with 4 adder and 1 shifter, and one bilinear filter with 2 adder and 1 shifter, and two cross registers structure for storing unfiltered pixels. Obviously, our methods can reduce half hardware cost compared with 6-tap FIR filter method. The number of filters, adders, shifters, and registers may mirror the bulk of hardware cost. Figure 14 illustrated the input data transfer and how the luma filter worked of STM filter method. We sequentially read in integer pixels from reference data buffer to input data array. As soon as, we obtained enough data, we can start to filter the integer pixels. For example, red blocks represent 4x4 luma partition, black block represent reference data extension, and circular number shows MC interpolation location. First read four pixels A1, B1, A2, and B2 form the first and second row. We needed to read successive integer pixels until we got twenty pixels from the first to the sixth row. Then, we had enough data to filter and output the interpolation data while reading and filtering integer pixels.

For FTM filter method, we will need fewer cycles for interpolating integer pixels and there was slight different in data transfer, such as read data from second and third row in cycle 0 and cycle 1 respectively, and read data from first and fourth row in cycle 2 as show in Figure 15.

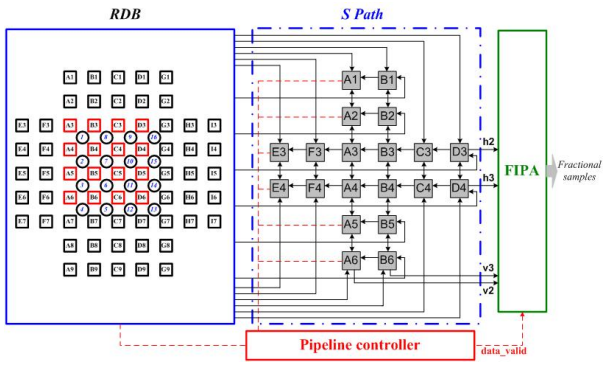

Figure 14. Reference Data Supplying Architecture of STM Filter

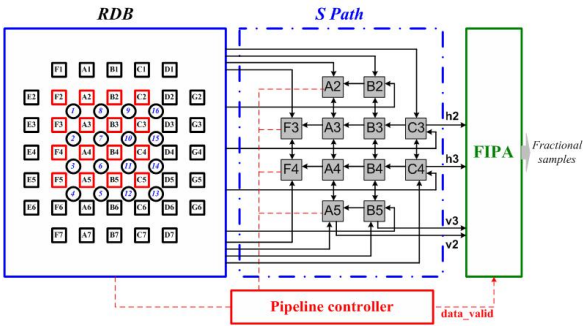

Figure 15. Reference Data Supplying Architecture of FTM Filter

The time management of one 4x4 block of STM filter method was shown in Figure 16. We needed 4 cycles for reading reference data. Then, after finish reading, we started to process interpolation. The interpolation pipeline delay is consumed 4 cycles, among process periods, the last cycle overlapped the first interpolation output pixel of interpolation computing, and 16 cycles consumed on full luma pixel interpolation computing. After we complete interpolation, we start to process the next block.

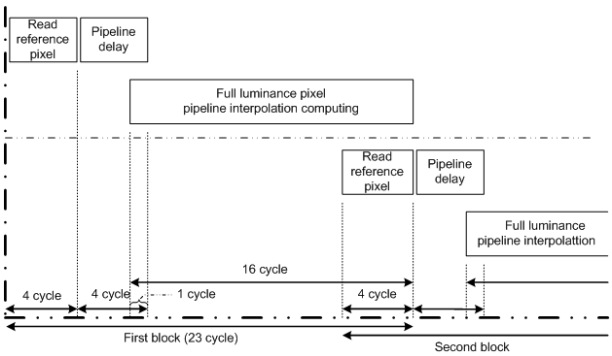

Figure 16. Time Management of One 4x4 Block of STM Filter

4. Experimental Results

The verification environment of our designs was based on H.264 reference software JM11.0 at High profile which was providing by JVT [14]. All sequences was make up of only one I-frame at the starting of a sequence, and two B-frames were inserted between each two uninterrupted P-frames, the QP of three types were set to 25. Full search motion estimation with a range of ±16 and hadamard transform was used. The test samples, includes eight QCIF and CIF sequences : "Akiyo", "Carphone", "Mobile", "Foreman", "News", "Coastguard", "Container", "Silent", one QCIF sequence : "Salesman", and two CIF sequences : "Stefan", "Football" were used for tests. Each sequence, composed of 100 QCIF (176x144 pixels) frames and 300 or 258 CIF (352x288 pixels) frames. Other specification such as frame rate was set to 30, entropy coding was set to CABAC, and the RD-optimized mode decision is disabled.

Since the H.264/AVC has been become the standard, in the software simulation, we kept the interpolation method of encoder and investigate the feasibility of adopting different interpolation filters (i.e., 4-tap diagonal FIR filter [9] and our proposed FTM filter and STM filter) in MC interpolation of decoder.

We modified the source code of H.264/AVC decoder, using 4-tap diagonal filter [9] and our proposed methods instead of traditional 6-tap filter algorithm in MC interpolation. The peak signal-to-noise ratio (PSNR) performance for the QCIF sequences and the CIF sequences are shown in Table 1 and Table 2. We only compare the PSNR performance of luma element (PSNRY) between the frames constructed by using 6-tap filter, 4-tap diagonal filter, FTM filter, and STM filter. The gain of table is the difference in PSNRY between 6-tap filter and employing interpolation methods. Form simulation results, the 4-tap diagonal filter caused average 4.65 dB and 8.2 dB PSNRY degradation for reconstructed video quality of QCIF sequences and CIF sequences, respectively. Relatively, our proposed FTM filter and STM filter average less than 4.47 dB and 2.02 dB of QCIF sequences, and less than 7.39 dB and 3.18 dB of CIF sequences.

Table 1. PSNR Performance for the QCIF Sequences

Sequence	6-tap	4-tap diagonal	FTM	STM
	PSNRY (dB)	PSNRY (gain)	PSNRY (gain)	PSNRY (gain)
Akiyo	40.84	38.04 (-2.80)	37.90 (-2.94)	39.88 (-0.96)
Carphone	39.50	34.10 (-5.00)	34.56 (-4.94)	37.50 (-2.00)
Mobile	36.07	25.38 (-10.69)	24.38 (-11.69)	27.99 (-8.08)
Foreman	38.56	30.82 (-7.74)	31.97 (-6.59)	35.41 (-3.15)
News	39.21	36.28 (-2.93)	36.96 (-2.25)	38.16 (-1.05)
Coastguard	36.56	33.01 (-3.55)	32.92 (-3.64)	34.88 (-1.68)
Container	38.51	33.94 (-4.57)	33.95 (-4.56)	38.46 (-0.05)
Silent	38.22	35.94 (-2.28)	36.09 (-2.13)	37.53 (-0.69)
Salesman	38.07	36.18 (-1.89)	36.51 (-1.56)	37.53 (-0.54)
Average	38.39	**33.74 (-4.65)**	**33.92 (-4.47)**	**36.37 (-2.02)**

Subsequently, we further discussed the restructured video quality of different interpolation methods. We could obviously realized not only the PSNRY degradation as show in above tables, but also damaged and caused the sharpen effect for reconstructed frame. Such as the roof and the eyes of foreman sequence, and the newscaster faces of news sequence. There are two reasons may cause the effects, first reason was the mismatch between encoder and decoder, another was the 4-tap diagonal filter using diagonal direction integer-pixel samples to process intermediate half-pixel sample, which may created lower correlation with 4-tap filter coefficients compared with using horizontal or vertical direction integer-pixel samples with 6-tap filter coefficients in traditional 6-tap filter. This signifies the reconstructed frame may cause larger error under mismatch situations, nevertheless since the FTM

filter using four nearest half-pixel samples with mean method to deal with the intermediate half-pixel sample, that enhanced the correlation about interpolation pixel, accordingly, there were lower sharpen influence for reconstructed frame beside 4-tap diagonal filter.

Table 2. PSNR Performance for the CIF Sequences

Sequence	6-tap	4-tap diagonal	FTM	STM
	PSNRY (dB)	PSNRY (gain)	PSNRY (gain)	PSNRY (gain)
Akiyo	41.91	32.61 (-9.30)	34.39 (-7.52)	39.11 (-2.80)
Carphone	39.69	31.32 (-8.37)	32.56 (-7.13)	37.11 (-2.58)
Mobile	36.58	22.63 (-13.95)	23.18 (-13.40)	27.99 (-8.59)
Foreman	38.43	31.11 (-7.32)	32.66 (-5.77)	35.79 (-2.64)
News	40.35	32.62 (-7.73)	33.60 (-6.75)	38.27 (-2.08)
Coastguard	37.03	28.57 (-8.46)	29.48 (-7.55)	32.26 (-4.77)
Container	38.35	27.97 (-10.38)	28.06 (-10.29)	37.53 (-0.82)
Silent	38.11	32.81 (-5.30)	34.29 (-3.82)	37.02 (-1.09)
Stefan	37.75	28.03 (-9.72)	28.27 (-9.48)	32.15 (-5.60)
Football	38.22	36.71 (-1.51)	35.99 (-2.23)	37.35 (-0.87)
Average	38.64	**30.44 (-8.20)**	**31.25 (-7.39)**	**35.46 (-3.18)**

For the same reason as FTM filter method, the STM filter using nearest four half-pixel samples to process the intermediate half-pixel sample, and the four half-pixel samples in horizontal and vertical direction were processed by using 6-tap FIR filter, hence, the sharpen effect may be improved for reconstructed frame. Such combination may create almost video quality for reconstructed frames contrast with traditional 6-tap filter and reduced the consumption of hardware cost. Obviously, not all of the PSNR performances are superiority of the reconstructed frames by STM filter over the reconstructed frames by 4-tap diagonal filter. The video quality by STM filter was nearly with original frame and reconstructed frame by other interpolation methods. Considering the nearly video quality for H.264/AVC decoder, STM filter was the better choices on the lower computing complexity video decoding system.

5. Conclusion

In this paper, we proposed two luma interpolation algorithms for MC interpolation in H.264/AVC decoder--FTM filter and STM filter and designed effective parallel and pipelined hardware architecture for the proposed algorithms, which included a new data supply method--Snake Path controller for realization in hardware. The average PSNR performance of luma element in FTM filter, by which that is less than 7.39 dB and higher than 0.81 dB compared with 6-tap FIR filter and 4-tap diagonal FIR filter at penalty of slightly sharpen effect; the average PSNR performance in STM filter, by which that was less than 3.18 dB and higher than 5.02 dB compared with traditional 6-tap FIR filter and 4-tap diagonal FIR filter at nearly video quality. Both the FTM and STM MC interpolator were synthesized with TSMC 0.13 um technology. Our design employs fewer adders, shifters, and registers for realization in hardware, the hardware cost of FTM filter and STM filter could save 68% and 48% compared with 6-tap FIR filter, and that could save 21% and increase 29% compared with 4-tap diagonal FIR filter. If the computational complexity was a top priority and the video quality was next consideration, the FTM filter may be a better choice with lower hardware cost design; but if the video quality is a top priority and the computational complexity is next consideration, the STM filter may be another better choice with nearly reconstructed video quality design.

6. Reference

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer and T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity," IEEE on Circuits and Systems Magazine, vol.4, pp. 7-28, 2004.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, 2003.

[3] A. Azevedo, B. Zatt, L. Agostini, and S. Bampi, "Motion compensation sample processing for HDTV H.264/ACV decoder" in IEEE Norchip, pp. 110-113, Nov. 2005.

[4] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 704-716, July 2003.

[5] H. W. Feng, Z. G. Mao, J. X. Wang, D. F. Wang, "Design and implementation of motion compensation for MPEG-4 AS profile streaming video decoding," in Proc. 5th Int. Conf. ASIC, vol. 2, pp. 942-945, Oct. 2003.

[6] Ville Lappalainen, Antti Hallapuro, and Timo D.Hamalainen, "Complexity of Optimized H.26L Video Decoder Implementation", Circuits and Systems for Video Technonlogy, IEEE Transactions, July 2003.

[7] I. E. G. Richardson, H.264 and MPEG-4 video compression: Video Coding for NextGeneration Multimedia. Chichester, UK: John Wiley & Sons, 2003.

[8] Philip P. Dang. Embedded architecture for fast implementation of H.264 subpixel interpolation. in: Subramania Sudharsanan, V. Michael Bove. Embedded Processors for Multimedia and Communications II. San Jose, CA, USA: SPIE PRESS, 72-78, 2005.

[9] T. Wedi and H. G. Musmann, "Motion- and Aliasing-compensated Prediction for Hybrid Video Coding," IEEE Trans. Circuit Syst. Video Technol., vol.13, no.7, pp.577-586, July 2003.

[10] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC", JVTG050, 2003.
[11] Ronggang Wang, Mo Li, Jintao Li, Yongdong Zhang, "High Throughput and Low Memory Access Sub-pixel Interpolation Architecture for H.264/AVC HDTV Decoder" IEEE Transactions on Consumer Electronics. vol.51, issue.3, pp.1006-1013, Aug. 2005.
[12] W. N Lie, H. C. Yeh, C. F Chen, and etc., "Hardware-Efficient Computing Architecture for Motion Compensation Interpolation in H.264 Video Coding" in Proc. of IEEE Int. Symp. On Circult and Systems, Kobe, Japan, pp. 2136-2139, May 2005.
[13] T. Sihvo, and J. Niittylahti, "H.264/AVC Interpolation Optimization" in Proc. IEEE Workshop on Signal Processing Systems Design and Implementation, 2003, SIPS 2005. 2-4 pp. 307-312, Nov. 2005.
[14] Video Team H.264/AVC Reference Software Version JM11.