AN OPTIMIZED MC INTERPOLATION ARCHITECTURE FOR HEVC

Zhengyan Guo, Dajiang Zhou, Satoshi Goto

Graduate School of Information, Production and System LSI, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan
E-mail: ocean.milo@ruri.waseda.jp

ABSTRACT

In the latest draft video compression standard, HEVC, a new 8-tap MC interpolation filter is adopted. For this component, we propose an efficient VLSI design which is composed of a reconfigurable filter, an optimized pipeline engine organization, and a filter reuse scheme. This results in 30% area saving from a non-optimized design. The proposed implementation supports a maximal throughput of QFHD@60fps. Our results also demonstrate the implementation cost of a well optimized HEVC interpolation component can be comparable to that of H.264, despite of the enhanced coding performance.

Index Terms— HEVC, interpolation, motion compensation

1. INTRODUCTION

High Efficiency Video Coding (HEVC) [1] is a draft standard under development by Joint Collaborative Team on Video Coding (JCT-VC). It is aimed at 50% bit rate reduction compared with H.264 standard. Many new features are proposed for better coding efficiency, including applying an 8-tap filter for high-accuracy motion compensation interpolation. Motion estimation/compensation is significant for video compression, by removing the temporal redundancy of video contents to a large extent. In a video decoder, interpolation is the most computation intensive component of motion compensation (MC).

In HEVC, a new 8-tap interpolation filter is adopted for fractional sample value prediction in MC [2]. The accuracy for luma interpolation is 1/4 pixel, so 15 positions should be calculated. It requires 11x11 reference pixels for one 4x4 sub-block prediction in the worst case. Compared with the 6 tap filter used in H.264 standard, the 8-tap filter will cost more area in hardware implementation. So designing an efficient architecture for MC luma interpolation is necessary for real time VLSI implementation for high quality video. There are some previous works focusing on designing efficient architecture for H.264 MC interpolation [3][4][5][6].

This research was supported in part by the Knowledge Cluster Initiative (2nd Stage) and Waseda University Ambient SoC Global COE Program of MEXT, Japan, and by the JST CREST Project.

In this paper, firstly, we propose a reconfigurable filter to reduce the implementation area of MC interpolation for 16%. Secondly, We propose an optimized pipeline organization for the interpolation engine. Thirdly, A filter reuse scheme is proposed for parallelized design, which can further reduce 17% area. In total, 30% area can be reduced by applying our proposal compared with the non-optimized design.

The rest of this paper is organized as follows. In section 2, the luma interpolation algorithm of HEVC will be introduced. In section 3, the reconfigurable filter, the optimized pipeline organization and filter reuse scheme are illustrated in detail. The implementation results are analyzed in section 4. Finally, the conclusion is presented in section 5.

2. INTERPOLATION ALGORITHM

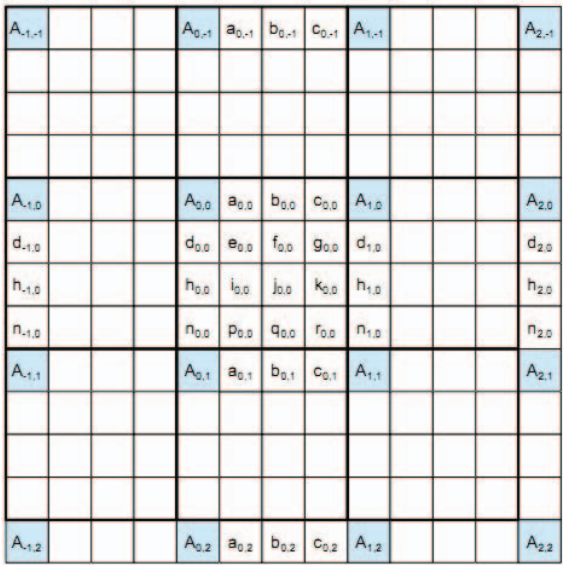


Fig. 1. Integer,half and quarter positions of luma pixels.

In the latest standard HEVC, three types of 8-tap filters are adopted as shown in equation (1)-(3). The detail of the equations can be found in [1]. According to the fractional position to be predicted, one of the three filters is applied for

interpolation.

$$A_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 10 * A_{-1,0} + 57 * A_{0,0} + 19 * A_{1,0} - 7 * A_{2,0} + 3 * A_{3,0} - A_{4,0} + 32) >> 6 \quad (1)$$

$$B_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 11 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 11 * A_{2,0} + 4 * A_{3,0} - A_{4,0} + 32) >> 6 \quad (2)$$

$$C_{0,0} = (-A_{-3,0} + 3 * A_{-2,0} - 7 * A_{-1,0} + 19 * A_{0,0} + 57 * A_{1,0} - 10 * A_{2,0} + 4 * A_{3,0} - A_{4,0} + 32) >> 6 \quad (3)$$

Fig. 1 shows the integer, half and quarter positions of luma component. Capital letters represent the integer position, it is directly output without calculation. a, b, c are calculated by applying the equation (1)-(3) to the nearest integer pixels in horizontal direction, respectively. And d, h, n are calculated by applying the above equation to the integer pixels in vertical direction, respectively. Rest fractional position pixel are calculated by applying above equation to the unrounded intermediate of d, h, n. For example, e, f, g is calculated by applying the equation (1)-(3) to the unrounded intermediate value of $d_{i,0}$, i=-3, -2.. 4 in the horizontal direction, respectively. The calculation process for i, j, k, p, q, r is the same as e, f, g.

Thus, there are three calculation patterns for luma interpolation. One pattern is in position of integer, a, b and c, only 1 filter operation is needed to obtain the predicted values. The second pattern is d, h, n and the third pattern is the rest 9 fractional positions. So in the worst case, 11x11 reference pixel values are required for interpolation of one 4x4 sub-block. A pipelined architecture should be designed suitable for these two calculation patterns. Besides, since the hardware architecture of the filter essentially determines the performance of luma interpolation engine, an optimized filter architecture is required to be proposed to reduce implementation area.

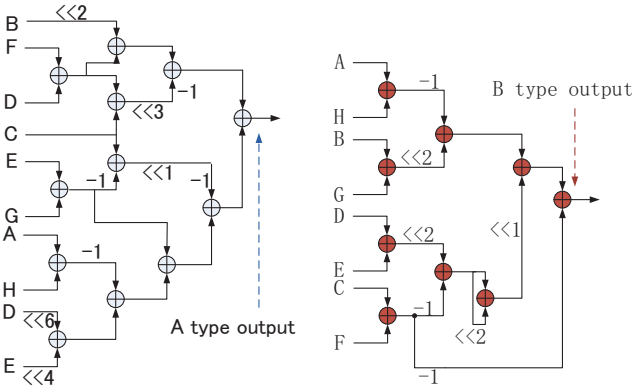
3. PROPOSED ARCHITECTURE

3.1. Reconfigurable Filter

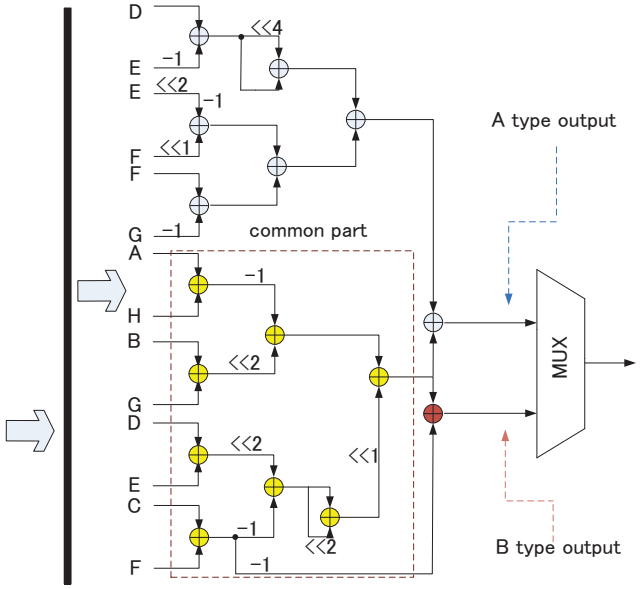
Type	Coefficients
A type	[-1,4,-10,57,19,-7,3,-1]
B type	[-1,4,-11,40,40,-11,4,-1]
C type	[-1,3,-7,19,57,-10,4,-1]

Table 1. Three types of 8-tap filters.

Table 1 shows the coefficients of 8-tap filters. We can find out that the coefficients of A and C type are symmetry, so the interpolation of A and C type filters can be done with the same hardware architecture by only reversing the order of input reference pixels. Based on equation (1), (2), 12 adders are required to realize the A type filter and 9 adders are required for B type filter. Thus, the 8-tap filter needs at least 21 adders to realizing hardware implementation, in total. Fig.2 (a) and Fig.2 (b) show the architectures of optimum A and B type filters, respectively. Since only one type of the three filters is used at one time, we can design an architecture to realize the three filters together. So we propose a reconfigurable filter aimed to reduce the area of the whole architecture.



(a) A type filter (12 adders) (b) B type filter (9 adders)



(c) Reconfigurable filter (16 adders)

Fig. 2. A and B type filters and the reconfigurable filter.

Fig.2 (c) shows the architecture of the proposed reconfigurable filter, there is a common part of A and B type filters, which is indicated by yellow adders. We define expression of the common part of A and B type filter as follows:

$Com. = -(A+H) + ((B+G) << 2) + ((D+E) << 2 - (C+F)) << 3 + ((D+E) << 2 - (C+F)) << 1;$

So by using common part, A and B type filters can be expressed as:

$AType = Com. + (((D-E) << 4) + (D-E) - (E << 2) + (F << 1) + (F-G));$

$BType = Com. - (C + F)$;

By using this common part, A and B type filters can be merged into one reconfigurable filter with a less number of adders. A, B, C, D, E, F, G, H are eight input reference pixels, their order will be reversed when using the coefficients of C type filter. The *afir output* and *bfir output* are the interpolation results of A and B type filters, a mux is used to select the output for one particular position. The proposed filter is composed of 16 adders, it can save 5 adders of the non-optimized 8-tap filter (21 adders). The structures of horizontal and vertical filters are the same. The different between them is that the horizontal filter deal with 8-bits input reference pixels while the vertical filter deal with 15-bits input data.

3.2. Optimized Pipeline Engine Organization

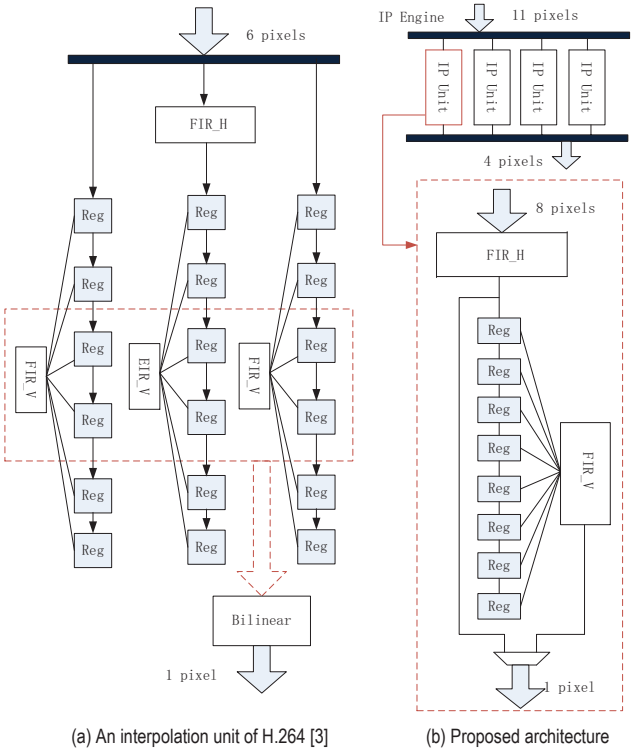


Fig. 3. Comparison of interpolation unit of H.264 and HEVC.

Fig.3 (a) shows the H.264 luma interpolator structure proposed by [3]. For N-pixel parallelism, it needs N horizontal filters, (2N+1) vertical filters and (2N+1) shift register arrays. In this paper we propose an optimized Pipeline HEVC luma interpolation engine. Fig.3 (b) shows the architecture of the interpolation unit, it is composed of 1 horizontal filter, 1 vertical filter and 1 shift register array. For N-pixel parallelism, it needs N horizontal filters, N vertical filters and N shift register arrays. Less modules are required in hardware implementation mainly due to the less calculation patterns in HEVC. In H.264 there are 5 calculation patterns based on fractional position, while in HEVC there are 3. We also reorder the interpolation algorithm that the intermediate unrounded values of a, b, c are used instead of d, h, n for the interpolation of e, f, g, i, j, k, p, q, r. It is convenient for hardware implementation but still confirms to the specification.

In the proposed architecture, for a, b, c, the values of reference pixels are transferred into the horizontal filter. Based on the predicted position, the coefficients of horizontal filter are selected and the rounded results of a, b, c are transferred to the output. For d, h, n, the shift register array is used to store the 8 reference pixels of vertical direction. They are transferred into the vertical filter together. For e, f, g, i, j, k, p, q, r, the shift register array is used to store the unrounded intermediate value of a, b, c.

To predict one 4x4 sub-block by this pipelined structure, 4x11 reference pixels are needed for interpolation of G, a, b, c and 11x11 pixels are needed for interpolation of rest fractional positions. Since the speed of this pipeline architecture is determined by how many rows of reference pixels are needed. Thus, the execution time of proposed engine is 11x16x2 = 352 cycles/MB for bi-directional prediction, in the worst case. The longest delay from input to output is 9 clock cycles. The throughput is 256/352 = 0.73 pixel/cycle, in the worst case.

3.3. Filter Reuse Scheme

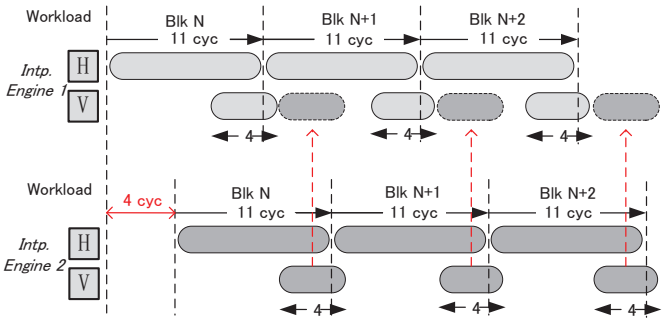


Fig. 4. Filter reuse scheme by sharing vertical filter.

Vertical filters cost more than 30% hardware of the whole design. And in the process of interpolation, the best workload of vertical filter is only (4 cyc/11 cyc). We propose a filter reuse scheme to reuse the vertical filters in parallelized design, as shown in Fig.4.

Parallelized design is implemented to improve the throughput of this subsystem. We call it dual-engine structure. When two interpolation engines do interpolation of two different 4x4 sub-blocks, we set one engine starts 4 clock cycles earlier than the other one. Thus, the vertical filters can be shared in a time-division manner, resulting in the reduction of logic circuits area. Half vertical filters are reduced in every two engines.

	Wang's [3]	Sze's [4]	Zhou's [5]	Non-optimized HEVC	Proposed single engine	Proposed dual-engine
Standard	H.264		H.264, AVS	HEVC		
Technology	0.18um	65nm	0.18um	90nm	90nm	90nm
Gate account	20686	N/A	26300	23304	19600	32496
Parallelism	4x	8x	4x	4x	4x	8x
Execution time cycles/MB	288	144	288	352	352	176
Area efficiency (pixel/cycle)/k gate	0.043	N/A	0.043	0.031	0.037	0.045
Freq. for 1080p@30fps	100MHz	31.5MHz	133MHz	85.5MHz	85.5MHz	42.7MHz
Freq. for QFHD@60fps P frame only	N/A	126MHz	N/A	N/A	N/A	171MHz

Table 2. Result comparison of previous works[3][4][5], non-optimized architecture, proposed single-engine and dual-engine. Area efficiency = throughput / gate account.

4. IMPLEMENTATION RESULTS

The proposed architecture is implemented in Verilog HDL and synthesized using SMIC 90nm cell library, with its clock constraint set to 250MHz. Table 2 shows the implemented result of proposed architecture and non-optimized one. Dual-engine is a proposed architecture with reconfigurable filter and filter reuse scheme for 8-pixel parallelism, and single engine is for 4-pixel parallelism.

In single engine, by applying the reconfigurable filter, the gate account of the whole proposed architecture is 19.6k at 250MHz. 16% area is reduced compared with non-optimized HEVC MC interpolation subsystem. In dual-engine, the two interpolation engines share 4 vertical filters by applying filter reuse scheme, this contributes to the area reduction of 17% compared with single engine and 30.4% compared with non-optimized architecture.

Table 2 also compares the implemented result of H.264 and HEVC. The throughput of HEVC MC luma subsystem is 0.73 pixel/cycle, which is 18% lower than wang's work[3] of 0.89 pixel/cycle. The lower throughput is because that 11 reference pixel rows are needed for one 4x4 sub-block prediction in HEVC, while in H.264 only 8 rows are needed. Besides, the implementation area of them are different. So we use area efficiency to compare the result of H.264 and HEVC. By applying our proposed reconfigurable filter and filter reuse scheme, the area efficiency is improved to 0.73x2/32.5 = 0.045 (pixel/cyle)/k gate. It is comparable with H.264's area efficiency of 0.043.

Though using 8-tap filters in HEVC contributes to bit rate reduction and the area is smaller by applying our proposal, the throughput of HEVC MC is also lower than H.264. So the hardware implementation cost of the optimized subsystem is comparable with H.264.

5. CONCLUSION

In this paper, we propose a MC luma interpolation architecture with reconfigurable filter and filter reuse scheme for HEVC decoder. It supports 1080p@30fps at the frequency of 42.7MHz. By using our proposal, 30% area is reduced in total. It can support QFHD (3840x2160)@60fps. Our results also demonstrate the implementation cost of a well optimized HEVC interpolation component can be comparable to that of H.264, despite of the enhanced coding performance.

6. REFERENCES

[1] ITU-T, WD3: Working Draft 3 of High-Efficiency Video Coding, JCTVC-E603, March, 2011.

[2] E. Alshina, "CE3: Experimental results of DCTIF by Samsung", JCTVC-D344, 20-28 January, 2011.

[3] S.-Z.Wang, T.-A.Lin,T.-M.Liu, and C.-Y.Lee," A new motion compensation design for H.264/AVC decoder", in Proc.IEEE Int. Symp. Circuits Syst., May 2005, vol. 5, pp. 4558-4561.

[4] V. Sze, D. F. Finchelstein, M. E. Sinangil, and A. P. Chandrakasan, A 0.7-V 1.8mW H.264/AVC 720 p video decoder, IEEE J. Solid-State Circuits, vol. 44, no. 11, Nov. 2009.

[5] D. Zhou and P. Liu, A Hardware-Efficient Dual-Standard VLSI Architecture for MC Interpolation in AVS and H.264, IEEE International Symposium on Circuits and Systems (ISCAS 2007), 2007.

[6] C.-Y. Tsai, T.-C. Chen, T.-W. Chen, , and L.-G. Chen, "Bandwidth optimized motion compensation hardware design for H.264/AVC HDTV decoder,"in Proceedings of 2005 International Midwest Symposium on Circuit and Systems (MWSCAS'05), 2005.