Low Cost and High Throughput FME Interpolation for the HEVC Emerging Video Coding Standard

Vladimir Afonso, Henrique Maich, Luciano Agostini, Denis Franco
Federal University of Pelotas (UFPel)
Pelotas, Brazil
{vafonso, hdamaich, agostini}@inf.ufpel.edu.br, denis.franco@ufpel.edu.br

***Abstract*—The new demands for high resolution digital video applications are pushing the development of new techniques in the video coding area. This paper presents the hardware design of the sub-pixel interpolator for the Fractional Motion Estimation algorithm defined by the HEVC emerging standard. Based on evaluations using the HEVC reference software, a strategy was defined to be used in the architectural design. The designed architecture was described in VHDL and synthesized for Altera FPGAs. The hardware designed presents interesting results in terms of performance, being able to process QFHD videos (3840x2160 pixels) in real time.**

I. INTRODUCTION

There are currently many applications and devices with support to digital videos, like DVD and Blu-Ray players, digital TV, smartphones, among others. The real time processing of high definition digital videos is associated with a high computational complexity. This complexity is function of the coding techniques used to reduce the amount of data necessary to represent the video sequences. Software solutions targeting video coding demand processors with high performance, resulting in an expressive cost in energy consumption. Thus, video coders in software are impracticable for many applications, especially those directed to mobile devices. Therefore, the design of dedicated hardware is extremely important in the video coding area.

A video coder goes through many steps that allow the encoding of digital video sequences. Each step has the goal to explore some kind of information redundancy present in the data used to represent the sequences. The inter-frame prediction identifies and reduces the redundancy present among neighbor temporal frames in a video sequence. The motion estimation (ME) is the most important step of the inter-frames prediction. ME identifies, in previously processed frames (called reference frames), that block which is the most similar one with the block that are being encoded. When this block is located, ME generates a motion vector to identify the position of this block inside the reference frame. Then only the motion vector and the residual error between current block and the reused block must be sent to the other encoder modules.

The ME module is the one that brings more compression gains inside a video coder [1]. The ME can use several additional techniques to increase the encoder gains. In the emergent HEVC standard, the ME is applied over variable size blocks, represented as PUs (Prediction Units) [2]. Furthermore, it is possible to apply a refinement step in the ME, called Fractional Motion Estimation (FME). The FME allows an increase in the encoding efficiency using samples at sub-pixel positions beyond integer position [3]. Finally, the HEVC uses some innovative techniques to compress the motion vectors [2], rising the compression gain.

Basically, the FME can be divided in two steps: (a) the sample interpolation to generate the fractional positions, and (b) the search for the best match considering the fractional samples. This paper presents the hardware design for the HEVC FME interpolation, considering 8x8 blocks.

II. EVALUATION USING THE REFERENCE SOFTWARE

Since the HEVC inter-prediction complexity is very high, strategies to reduce this complexity are a relevant research topic, mainly for applications with processing and power consumption restrictions. But these strategies must maintain the quality and the compression rate in acceptable levels.

The main part of the HEVC inter-frames computational complexity is related with the variable PU size, since each PU size must be evaluated in terms of the rate-distortion cost [2] to define which the best PU size is. There are 24 PU sizes defined in the current version of HEVC, and this number can rise to 25 when the 4x4 size is enabled [2]. Then, some evaluations were done in this work aiming to discover which are the most selected PU sizes in the HEVC encoding process. The evaluations were done using the version HM-8.0rc2 of the HM (HEVC Model) reference software [4].

The test configurations defined by the JCT-VC [5] were used in the evaluation process. The JCT-VC defines 8 conditions to do the tests, with combinations of high efficient and low complexity tools in three configurations: (a) intra-only, (b) random-access and (c) low delay [2]. The evaluations were done considering the main profile. All 24 video sequences defined in [5] were encoded for the five classes in the random-access and low-delay configurations. The intra-only configuration was not used since ME is not employed in this configuration. The results of this evaluation have shown that the most used PU size was 8x8 in almost all scenarios, followed by 16x16 and 32x32 PU sizes.

978-1-4673-4900-0/13/$31.00 ©2013 IEEE

Another test was done to evaluate the losses in terms of image quality and bit-rate when using only the 8x8 PU size in the HEVC ME. This evaluation shows average bit-rate losses lower than 13.18%. The average image quality losses were lower than 0.44dB.

The last test was done to evaluate the relevance of the fractional ME when only the 8x8 block size was used. Then, the FME was disabled and all tests were done again. As result was possible to conclude that the use of FME provides an important gain in bit-rate and quality. The bit-rate gain was near to 9.73% and the quality gain was near to 0.18dB.

Considering the presented results, it was possible to verify that the FME is a very important tool in the HEVC encoder. Other important point about the FME is that it presents a low complexity when all encoder is considered. The FME calculations can be done in parallel with the integer ME of the next block and no other encoder loops are necessary. On the other hand, even providing expressive gains in compression and quality, the use of variable PU sizes increases a lot the encoder complexity, since all PU sizes must be evaluated. When only one PU size is evaluated the complexity is drastically reduced, since only one block size must be processed by further encoder steps and the decision of which is the best block is unnecessary. In this scenario, the global encoder complexity is reduced in more than 24 times, since the HEVC defines at least 24 PU sizes and also many intra-prediction modes. So, the losses in quality and compression are justified, especially for applications with processing and energy consumption restrictions.

After the analysis presented in this section, this work focused in a hardware design of the HEVC FME interpolation unit using a fixed block size of 8x8 samples.

III. FRACTIONAL ME IN THE HEVC

The most recent HEVC FME definitions are presented in documents [2] and [6]. In those documents are detailed the process to generate the sub-pixels positions according to the HEVC standard. Fig. 1 shows the samples of luminance in integer positions, as well as in fractional positions for the interpolation with precision of quarter pixel defined in the HEVC standard, considering 8x8 block size.

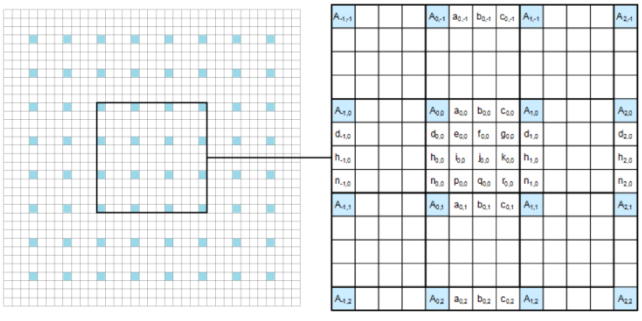

Figure 1. Samples at integer positions (shaded squares and with uppercase) and at fractional positions (with lowercase).

According with the FME defined in the HEVC [6], it is possible to generate the fractional positions $\mathbf{a_{0,0}}$, $\mathbf{b_{0,0}}$, $\mathbf{c_{0,0}}$, $\mathbf{d_{0,0}}$, $\mathbf{h_{0,0}}$ and $\mathbf{n_{0,0}}$ in Fig. 1, with the application of an 8-tap FIR filter using only the integer positions. The generation of the fractional positions $\mathbf{e_{0,0}}$, $\mathbf{f_{0,0}}$, $\mathbf{g_{0,0}}$, $\mathbf{i_{0,0}}$, $\mathbf{j_{0,0}}$, $\mathbf{k_{0,0}}$, $\mathbf{p_{0,0}}$, $\mathbf{q_{0,0}}$ and $\mathbf{r_{0,0}}$ requires, firstly, the calculation of the fractional positions $\mathbf{a_{0,i}}$, $\mathbf{b_{0,i}}$ and $\mathbf{c_{0,i}}$, where **i** varies from **-3** to **4** in vertical direction. Then, the fractional positions generated in the vertical direction are used to generate these samples also using an 8-tap FIR filter. The used FIR filter coefficients depend on the sample position, as shown in Table I.

TABLE I. FIR FILTER COEFFICIENTS

Fractional Positions	FIR Filter Coefficients
$a_{i,j}$, $d_{i,j}$, $e_{i,j}$, $f_{i,j}$, $g_{i,j}$	{–1, 4, –10, 58, 17, –5, 1, 0}
$b_{i,j}$, $h_{i,j}$, $i_{i,j}$, $j_{i,j}$, $k_{i,j}$	{–1, 4, –11, 40, 40, –11, 4, –1}
$c_{i,j}$, $n_{i,j}$, $p_{i,j}$, $q_{i,j}$, $r_{i,j}$	{0, 1 , –5, 17, 58, –10, 4, –1}

IV. HARDWARE DESIGN

An analysis of the interpolation algorithm was done in order to reduce the complexity of the hardware design. Firstly, was possible to observe a similarity among some fractional positions generation, where the FIR filter coefficients were the same in some cases. Therefore, in order to reduce the cost of the designed hardware, the sub-pixel positions were divided in three groups, according to the FIR filter coefficients and their positions related to the integer positions, following the same distribution presented in Table I. The names of these three types of filters are: Up, Middle and Down. Fig. 2, presents the location of these filters considering the interpolated positions.

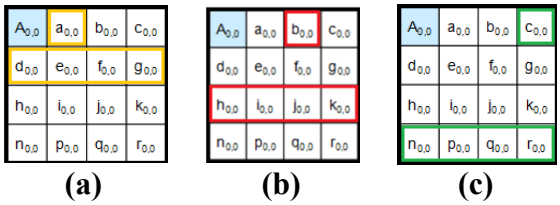

Figure 2. Filter types (a) Up, (b) Middle and (c) Down.

The next step was to optimize the filters with some algebraic manipulations. The multiplications by constants were transformed in shift-adds and common sub-expressions sharing were also considered.

The optimizations applied in the Middle filter are presented in this paper as an example, but the same process was done to the Up and Down filters. The interpolation input and output are represented as $\mathbf{a_0}$-$\mathbf{a_7}$ and **s**, respectively, as presented in equation (1). The first optimization was to transform the multiplications by constants in shift-adds, where all constants are represented as numbers in base 2, as presented in equation (2). This strategy allows a multiplierless design, decreasing the interpolator hardware cost and increasing its global performance. Since each constant can be replaced by many sets of shift-adds, the selected set of shift-adds considered the lowest use of additions and, consequently, the lowest cost of hardware. Then, the multiplications by the same factors were grouped, as can be seen in equation (3). This optimization allows a reduction in the number of shifts and a reduction in the adders bit width, since the additions are done first and the multiplications are done later. Other optimization is the sub-expression share. In this case, the common calculations R_1 and R_2, presented in equation (4) are

done only once and the result is reused as presented in equation (5). Finally, equation (6) presents the result of the optimization process, with the multiplications changed to shifts.

$$s = [-a_0 + 4*a_1 - 11*a_2 + 40*a_3 + 40*a_4 - 11*a_5 + 4*a_6 - a_7] >> 6 \quad (1)$$

$$s = [-a_0 + 4*a_1 - (8*a_2 + 2*a_2 + a_2) + (32*a_3 + 8*a_3) + (32*a_4 + 8*a_4) - (8*a_5 + 2*a_5 + a_5) + 4*a_6 - a_7] >> 6 \quad (2)$$

$$s = [1*(-a_0 - a_2 - a_5 - a_7) + 2*(-a_2 - a_5) + 4*(a_1 + a_6) + 8*(-a_2 + a_3 + a_4 - a_5) + 32*(a_3 + a_4)] >> 6 \quad (3)$$

$$R_1 = a_2 + a_5 \quad \text{and} \quad R_2 = a_3 + a_4 \quad (4)$$

$$s = [1*(-a_0 - a_7 - R_1) - 2*R_1 + 4*(a_1 + a_6) + 8*(R_2 - R_1) + 32*R_2] >> 6 \quad (5)$$

$$s = \{[(-a_0 - a_7 - R_1) - (R_1 << 1) + [(a_1 + a_6) << 2] + [(R_2 - R_1) << 3] + [R_2 << 5]\} >> 6 \quad (6)$$

The hardware design of the interpolation filters started considering the optimized equations. Three architectural versions were designed considering structural and behavioral descriptions: (a) purely combinational; (b) using two pipeline stages and (c) using four pipeline stages. All designed architectures consume in one clock cycle the input data necessary to generate one new interpolated sample. Then, the architectures are able to generate one sample per clock cycle (when the pipeline is full for those solutions which use pipeline). The integer input samples are 8 bit-wide but the input fractional positions are 10 bit-wide. This way, as the same filter could receive integer and fractional inputs, the bit width of the filters inputs were defined as 10 bits.

The pipeline versions were designed trying to best balance the stages, splitting the adders accordingly. Fig. 3 shows the architectural design of the Middle filter using four pipeline stages. The other filters used the same architectural template.

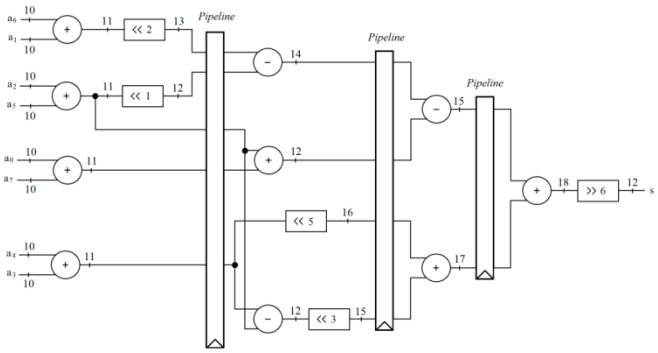

Figure 3. Hardware architecture for the Middle filter type.

The complete architecture designed for the FME interpolation is presented in Fig. 4 and it uses some instances of the three filters presented bellow and four buffers. Since the main goal of this work is to design a high throughput FME interpolation unit, the global architecture was designed targeting the generation of one complete line or column of fractional positions per clock cycle.

One buffer is used to store the input samples at integer positions, and the other three buffers are used to store the filters outputs in fractional positions. These buffers were implemented using register banks and are presented in Fig. 4, integrated with the interpolation filters. The information stored in the output filters are used in the next FME step, which is the search for the best match inside the interpolated area. The output information of the three filters is reordered inside the H-type (horizontal positions), V-type (vertical positions) and D-type (diagonal positions) buffers to be used in the next FME step. Fig. 4 also presents the filters inputs which are provided by the first buffer, containing the integer position samples, and by the H-type buffer, containing the interpolated positions used in the generation of the next interpolated samples.

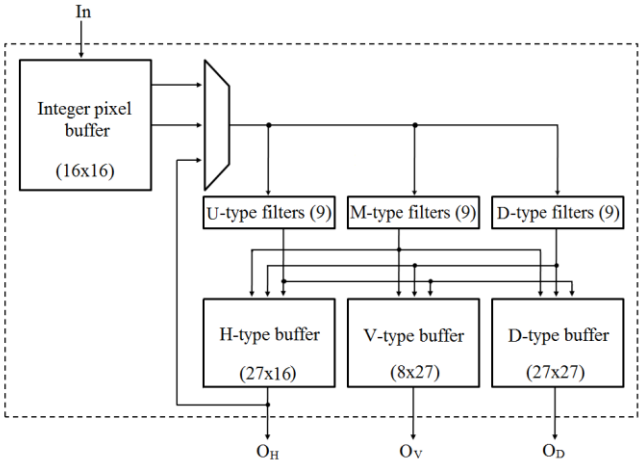

Figure 4. Hardware architecture for the FME interpolation.

The integer pixel buffer presented in Fig. 4 is able to store 256 positions with 8 bits each, because, besides the 8x8 block, it is necessary to store four additional samples around the block edge to allow the filtering process. Then the stored blocks with integer positions concerns 16x16 samples.

To allow the desired global architecture throughput, nine instances of each one of the three filters (Up, Middle and Down) were used, as presented in Fig. 4. Then, it is possible to calculate nine fractional positions per clock cycle in each group of filters. This means that 27 fractional positions are generated per clock cycle. It is important to notice that one 8x8 block with integer pixels generates 432 horizontal fractional position, 216 vertical fractional positions and 729 diagonal fractional positions.

The interpolation flow starts processing one line of the integer samples, where all horizontal fractional positions are calculated for this line (**a**, **b**, **c** samples in Fig. 1). This process is repeated for the 16 lines stored in the input buffer. The next step is the use of the columns of integer samples to generate the vertical fractional positions (**d**, **h**, **n** samples in Fig. 1) for this column. This process is repeated eight times, because only eight columns must be processed. The last step considers the horizontal fractional positions as inputs to generate the last fractional positions, called diagonal positions in this work. In this case, a total of 27 additional lines must be processed. According with the number of lines and columns that must be processed, it is possible to estimate an execution rate of 51 clock cycles to complete the interpolation of an 8x8 block.

V. SYNTHESIS RESULTS AND RELATED WORKS

Table II presents the synthesis results achieved with the designed hardware architectures for the FME interpolation filters, excluding the buffers. These results considered all 27 filters used in Fig. 4, with nine filters of each type (Up, Down and Middle). The VHDL description of these filters considering the three architectural versions designed in this work (combinational, two-stage pipeline and four-stage pipeline) were synthesized targeting a Stratix III EP3SE50F484C2 FPGA device [7]. The used synthesis tool was the Altera Quartus II [7].

TABLE II. SYNTHESIS RESULTS FOR THE FIR FILTERS

Interpolation Filter Version	Combinational ALUTs	Total Registers	Frequency (MHz)
Combinational	4,077	0	173.37
Pipeline and 2 steps	4,077	1,278	274.42
Pipeline and 4 steps	4,077	3,861	403.06

Since the hardware used in the buffers is common to the three architectural versions, the results were omitted in Table II. Hardware resources used in the buffers are separately presented in Table III.

TABLE III. HARDWARE ELEMENTS USED FOR BUFFERS.

Integer Samples Buffer	Vertical-type Buffer	Horizontal-type Buffer	Diagonal-type Buffer
256 8-bit REG + 256 4:1 MUX	**216 10-bit REG + 216 2:1 MUX**	**432 10-bit REG + 432 2:1 MUX**	**729 11-bit REG + 729 2:1 MUX**

Based on the operation frequency reached by the different filter versions it is possible to estimate the number of frames per second that can be processed, considering different resolutions. This estimation is presented in Table IV considering Full HD (1920x1080 pixels) and QFHD (3840x2160 pixels) resolutions. Considering these results it is possible to conclude that all developed filters can process Full HD videos in real time (at least 30 frames per second), but considering QFHD videos, the combinational version did not reach the needed performance.

TABLE IV. ESTIMATION OF FRAMES PER SECOND USING THE ARCHITECTURES DEVELOPED.

Developed Architecture	Full HD (1920x1080) (fps)	QFHD (3840x2160) (fps)
With Combinational Filters	104	26
With Pipeline Filters and 2 steps	166	41
With Pipeline Filters and 4 steps	243	60

To the best of our knowledge, there´s no other published work targeting the hardware design for the last HEVC interpolation definitions [6]. The work [8] presents a hardware design for the HEVC FME, but it is based on an older version of the standard. The latest standardization documents presented a lot of modifications in the FME, then a fairly comparison is not possible.

There are many other published works targeting the hardware design for the fractional motion estimation, but those works are focused on the H.264/AVC standard [9]. Some of these works are [10], [11] and [12]. Since the H.264/AVC FME is completely different than the HEVC FME, it is impossible to compare our solution with other works targeting the H.264/AVC standard.

VI. CONCLUSION

This work presented a hardware design for the FME interpolation process defined in the emerging HEVC video coding standard. This hardware design considered blocks with a fixed size of 8x8 samples intending to reduce the global encoder complexity.

The hardware design considered optimizations in the interpolation filters and then was possible to design a multiplierless and high throughput architecture. The synthesis results show that the designed hardware is able to interpolate QFHD (3840 x 2460) frames in real time.

The next step of this work will consider the design of the search step for the best match inside the fractional positions, generating a complete HEVC FME architecture.

REFERENCES

[1] A. Puri, X. Chen and A. Luthra, "Video Coding Using the H.264/MPEG-4 AVC Compression Standard," *Elsevier Signal Processing: Image Communication*, n. 19, pp. 793–849, 2004.

[2] I. Kim, K. McCann, K. Sugimoto, B. Bross and W. Han, HM8: High Efficiency Video Coding (HEVC) Test Model 8 Encoder Description (JCTVC-J1002), 2012.

[3] I. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia. Chichester: John Wiley and Sons, 2003.

[4] High Efficiency Video Coding (HEVC) Reference Software. Oct, 2012; https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/

[5] F. Bossen, Common Test Conditions and Software Reference Configurations (JCTVC-J1100), 2012.

[6] B. Bross, W. Han, J. Ohm, G. Sullivan and T. Wiegand, High Efficiency Video Coding (HEVC) text specification draft 8 (JCTVC-J1003), 2012.

[7] ALTERA. FPGA CPLD and ASIC from Altera. Altera Web Site. Oct, 2012; www.altera.com

[8] Z. Guo, D. Zhou, and S. Goto, "An optimized MC interpolation architecture for HEVC," IEEE International Conference on Acoustics, Speech and Signal Processing, March 2012.

[9] Joint Video Team (JVT) of ITU-T VCEG and ISO/IEC MPEG, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, May 2003.

[10] S. Yalcin, and I. Hamzaoglu, "A high performance hardware architecture for half-pixel accurate H.264 motion estimation," 14th International Conference on Very Large Scale Integration and System-on-Chip, October 2006.

[11] S. Oktem, and I. Hamzaoglu, "An efficient hardware architecture for quarter-pixel accurate H.264 motion estimation," 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, August 2007.

[12] C. Kao, H. Kuo, and Y. Lin, "High performance fractional motion estimation and mode decision for the H.264/AVC," IEEE International Conference on Multimedia and Expo, July 2006.