A Dedicated Hardware Solution for the HEVC Interpolation Unit

[1] Vladimir Afonso, [1] Marcel Moscarelli Corrêa, [1] Luciano Volcan Agostini, [2] Denis Teixeira Franco

{vafonso, mmcorrea, agostini}@inf.ufpel.edu.br, denisfranco@furg.br

[1] UFPel – Federal University of Pelotas
[2] FURG – Federal University of Rio Grande

Abstract

The increasing demand for high-resolution digital video for many different types of applications stimulated the efforts to improve the performance in video coding. This work presents a hardware architecture developed for the Interpolation Unit necessary for the Fractional Motion Estimation step defined in the emerging High Efficiency Video Coding (HEVC). The architectural solution was fully described in VHDL and obtained good results of processing rate and cost, achieving a maximum frequency of 187 MHz and processing up to 28 QFHD (3840x2160) frames per second.

1. Introduction

Video coding is currently an important research area in function of the increase demand for high-definition digital videos. There are different video coding standards, and these standards primarily define two things: (1) a coded representation (or syntax), which describes the visual data in a compressed form, and (2) a method to decode the syntax to reconstruct the visual information [1].

Nowadays, the most efficient video coding standard available is the H.264/AVC (Advanced Video Coding) [2]. However, video resolutions and the amount of information that must be processed are continuously increasing. Because of this demand, a new video coding standard, designed to be more efficient than the H.264/AVC, is being developed in a joint effort of the ITU-T and ISO/IEC, through the JCT-VC (Joint Collaborative Team on Video Coding) group. This new standard is currently known as HEVC (High Efficiency Video Coding) [3], and represents the state-of-the-art of video coding tools.

There are some important factors that should be considered in video coding. One of these factors is the computational complexity, since the real-time processing requires a very high calculation power to compress high-definition videos even modern general-purpose processors are not able to reach this restriction. In addition, software solutions running in an embedded system are not appropriate for applications like video encoders, because it would be necessary a very high-performance processor, prohibitively increasing the energy consumption. This is a general problem, but it is critical for mobile devices. Therefore, it is necessary the development of high-efficiency ASICs (Application Specific Integrated Circuits) with expressive processing rates and low energy consumption to support the HEVC standard.

Video encoders have different modules to explore each redundancy type present in visual information. The interframe prediction block uses the Motion Estimation (ME) process and it is responsible for the best results in terms of compression rates in the encoder, but it is also the most complex process [4]. It explores and reduces the temporal redundancy, which is the similarity among sequential frames and it works splitting the current frame into several blocks and then searching in the previous coded frames (reference frames) for the block that is most similar to the current one. After this search, a motion vector (MV) is generated to indicate the offset of the selected block inside the reference frame. The goal of ME is to find the best MV whilst keeping the computational complexity inside acceptable limits [1].

An important technique that can be used in ME is the Fractional Motion Estimation (FME). The FME allows an even greater efficiency by applying an interpolation process between integer position samples in reference frames, allowing a search for better matches in fractional positions. The FME is composed by two units: the Interpolation Unit, that generates the fractional position samples (sub-pixels), and the Search Unit, which searches for better matches composed by sub-pixels.

The HEVC emerging standard proposes new algorithms for the FME process, making it more efficient than the H.264/AVC FME. Therefore, the design of high-performance hardware architectures for the HEVC interpolation process is an important research effort.

2. State-of-the-Art and Related Works

In the HEVC proposal, some modifications were included in the interpolation filters used in the Interpolation Unit to improve the results obtained with the FME. While the H.264/AVC standard uses two dependent steps to generate the luminance samples in quarter-pixel positions, the HEVC produces the samples in quarter-pixel positions in a single step.

In H.264/AVC, the first step is responsible to obtain the samples in half-pixel positions from luminance samples in integer positions using a 6-tap FIR filter (Finite Impulse Response). After the interpolation, a search process in half-pixel positions to find a better match must take place. In the second step, the luminance samples in quarter-pixel positions are obtained through a bilinear filter using the data from the last search.

In HEVC, the process used to generate luminance samples for quarter-pixel positions applies an 8-tap DCT filter (Discrete Cosine Transform) and it is capable to calculate both half-pixels and quarter-pixels in a single step. It allows the FME to use a single search process for all sub-pixels [5].

As previously cited, the most efficient video coding standard currently available is the H.264/AVC. Regarding this standard, many works can be found in the literature including works related with the development of hardware architectures for FME, such as [6–10]. We did not found works in the literature directly related to the HEVC standard, and then was not possible to make comparisons between our results and related works.

3. Proposed Interpolation Unit Architecture

The hardware architecture designed in this work was based in two documents elaborated by JCT-VC [5][11]. In those documents, it is possible to find all processes used to generate the sub-pixel positions, according to the HEVC specification. Fig. 1 represents the integer samples (shaded blocks) and fractional sample positions (white blocks) for quarter-pixel interpolation in HEVC. This figure and the equations (1-6) which are used to obtain the fractional positions are important to understand the interpolation process.

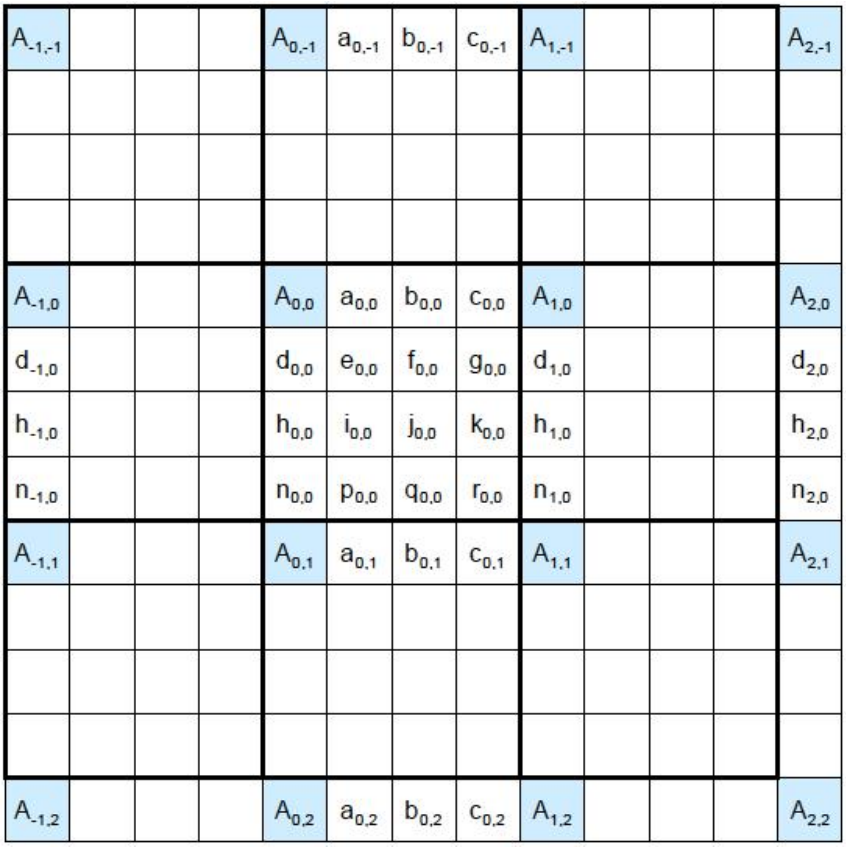

Fig. 1 – Integer samples and fractional samples used in the HEVC quarter-pixel interpolation.

Based on the algorithms given in [11] it was possible to find similarities in the equations of some fractional positions. Equation (1) is used to calculate horizontally aligned sub-pixels between integer positions ($a_{0,0}$, $b_{0,0}$ and $c_{0,0}$), and equation (2) for vertically aligned sub-pixels between integer positions ($d_{0,0}$, $h_{0,0}$ and $n_{0,0}$). The variables C_2-C_7 are coefficients that change according to sub-pixel positions, and their values are showed in tab. 1.

To calculate sub-pixels that are not horizontally or vertically aligned between integer positions, it is necessary to calculate the intermediate values $d1_{i,0}$, $h1_{i,0}$ and $n1_{i,0}$ before, using equation (3). Afterwards, the final prediction is realized. For the sub-pixels $e_{0,0}$, $f_{0,0}$ and $g_{0,0}$ equation (4) is used. The sub-pixels $i_{0,0}$, $j_{0,0}$ and $k_{0,0}$ come from equation (5) and the sub-pixels $p_{0,0}$, $q_{0,0}$ and $r_{0,0}$ are produced by equation (6).

$$(-A_{-3,0} + C_2A_{-2,0} + C_3A_{-1,0} + C_4A_{0,0} + C_5A_{1,0} + C_6A_{2,0} + C_7A_{3,0} - A_{4,0} + 32) >> 6 \quad (1)$$

$$(-A_{0,-3} + C_2A_{0,-2} + C_3A_{0,-1} + C_4A_{0,0} + C_5A_{0,1} + C_6A_{0,2} + C_7A_{0,3} - A_{0,4} + 32) >> 6 \quad (2)$$

$$-A_{i,-3} + C_2A_{i,-2} + C_3A_{i,-1} + C_4A_{i,0} + C_5A_{i,1} + C_6A_{i,2} + C_7A_{i,3} - A_{i,4} \quad (3)$$

$$(-d1_{-3,0} + C_2d1_{-2,0} + C_3d1_{-1,0} + C_4d1_{0,0} + C_5d1_{1,0} + C_6d1_{2,0} + C_7d1_{3,0} - d1_{4,0} + 2048) >> 12 \quad (4)$$

$$(-h1_{-3,0} + C_2h1_{-2,0} + C_3h1_{-1,0} + C_4h1_{0,0} + C_5h1_{1,0} + C_6h1_{2,0} + C_7h1_{3,0} - h1_{4,0} + 2048) >> 12 \quad (5)$$

$$(-n1_{-3,0} + C_2n1_{-2,0} + C_3n1_{-1,0} + C_4n1_{0,0} + C_5n1_{1,0} + C_6n1_{2,0} + C_7n1_{3,0} - n1_{4,0} + 2048) >> 12 \quad (6)$$

Tab. 1 – Coefficients of filters to generate sub-pixel positions

Type	Positions	Coefficients (C_2…C_7)
L(left)	$a_{i,j}$, $d_{i,j}$, $e_{i,j}$, $i_{i,j}$, $p_{i,j}$	{4, –10, 57, 19, –7, 3}
M (middle)	$b_{i,j}$, $h_{i,j}$, $f_{i,j}$, $j_{i,j}$, $q_{i,j}$	{4, –11, 40, 40, –11, 4}
R (right)	$c_{i,j}$, $n_{i,j}$, $g_{i,j}$, $k_{i,j}$, $r_{i,j}$	{3, –7, 19, 57, –10, 4}

In the specification of the designed architecture, blocks of 8x8 size were considered and three different filters were developed, one type for each coefficients group, according to tab. 1. Each one of the filters has three output values: one to implement equations (1-2); one to implement equation (3) and one to implement equations (4-6). The multiplications presented in equations (1-6) were converted in shifts-add to decrease the hardware use and the power consumption and to increase the architecture processing rate. In addition to the filters used in the operative part, register banks were described to implement the barrel shifting operation needed to implement the multiplications. The filters receive an entire line or column coming from these register banks and they can calculate an entire line or column of sub-pixel positions in one clock cycle. Fig. 2 shows the simplified block diagram of the designed architecture.

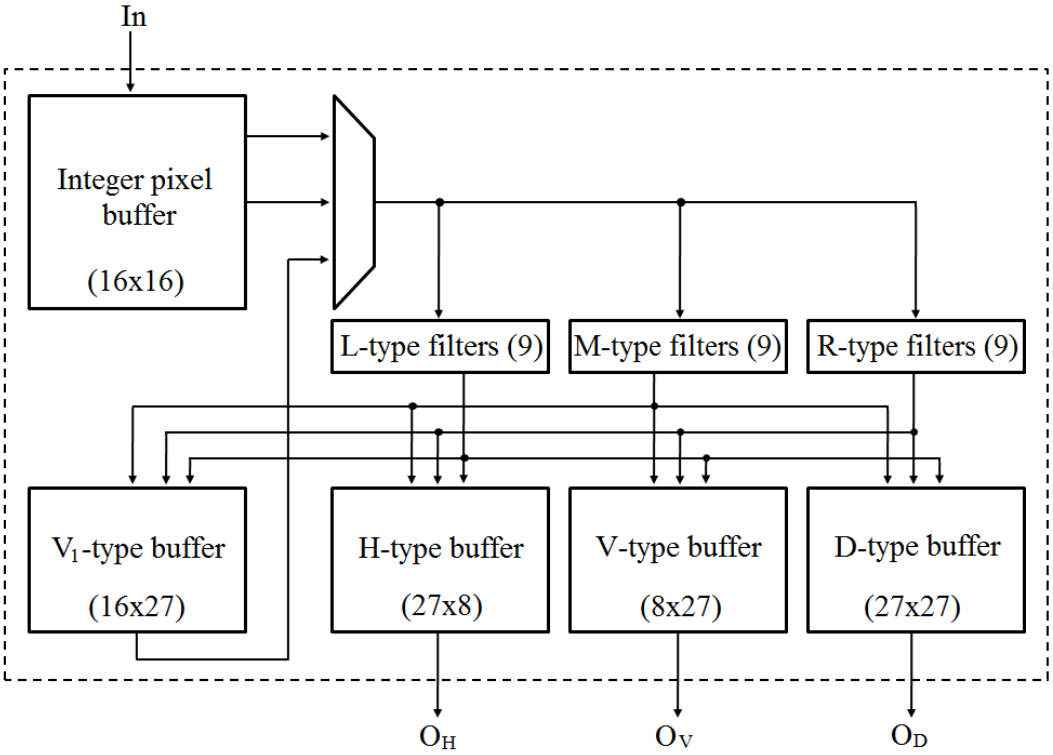

Fig. 2 – Simplified block diagram of the designed architecture.

4. Synthesis Results

The proposed design was fully described in VHDL and synthesized to a Xilinx Virtex4 XC4VLX15 FPGA device, using the Xilinx ISE 10.1 synthesis tool.

As shown in fig. 2, this design needs a total of 27 interpolation filters (9 of each type) in order to simplify the control unit and achieve a high efficiency. Also, register banks are used to store and shift samples.

When synthesized to the XC4VLX15 device, the architecture achieved a maximum frequency of 187 MHz. Since our design needs 52 clock cycles to process an entire block, this frequency allows it to process 111 Full HD (1920x1080) frames per second or 28 QFHD (3840x2160) frames per second.

The operative module of the architecture, composed by the filters, consumed 4615 slices (75% of the target device resources), 3309 slice flip-flops (26%) and 8333 4-input LUTs (67%). The buffers were mapped as registers banks and the amount of hardware elements used by them is shown in tab. 2.

Tab. 2 – Hardware elements used by the buffers.

Buffer	**8-bit REG + 4:1 MUX**	**10-bit REG + 2:1 MUX**	**11-bit REG + 2:1 MUX**	**16-bit REG + 2:1 MUX**
Integer Samples	256	-	-	-
V_1 Type	-	-	-	432
V Type	-	216	-	-
H Type	-	216	-	-
D Type	-	-	729	-

The use of hardware resources may seem high, but it is important to note that this is not an implementation optimized for FPGA devices, and therefore DSP blocks and dedicated memory were not used, since this is a multiplier free implementation.

5. Conclusions and Future Works

In this paper, a high performance hardware architecture for the HEVC sub-pixel interpolation unit was presented.

When synthesized to a Xilinx Virtex4 FPGA, our architecture achieves a very high processing rate. It can process very high definition videos like QFHD in real time, and it can also process Full HD videos when running at lower frequencies, which is very important when power and energy consumption restrictions are being considered.

As future works, it is planned to design a fully functional Fractional Motion Estimation architecture that uses a fast search algorithm for the HEVC video coding standard.

6. References

[1] I. G. Richardson, H.264 and MPEG-4 Video Compression, Wiley, 2003.

[2] 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.

[3] G. Sullivan, and T. Wiegand, Draft requirements for next-generation video coding project, International Telecommunication Union, 2009.

[4] A. Puri, et al., Video Coding Using the H.264/MPEG-4 AVC Compression Standard. Elsevier Signal Processing: Image Communication, 2004.

[5] K. McCann, B. Bross, S. Sekiguchi, and W. Han, High Efficiency Video Coding (HEVC) Test Model 3 Encoder Description, 2011.

[6] M. M. Corrêa, and M. T. Schoenknecht, "A H.264/AVC quarter-pixel motion estimation refinement architecture targeting high resolution videos," *VII Southern Conference on Programmable Logic*, 2011.

[7] M. M. Corrêa, M. T. Schoenknecht, and R. S. Dornelles, "A high-throughput hardware architecture for the H.264/AVC half-pixel motion estimation targeting high-definition videos," *International Journal of Reconfigurable Computing*, 2011.

[8] 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.

[9] 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.

[10] 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.

[11] HM3. “HEVC Test Model Reference Software.” Jan, 2012; http://hevc.info/HM-doc/