Approximate Interpolation Filters for the Fractional Motion Estimation in HEVC Encoders

ABSTRACT

The High Efficiency Video Coding (HEVC) standard includes several modifications that make it more complex than its predecessor, H.264/AVC. Motion Estimation (ME) is one of the most complex HEVC steps, consuming more than 60% of the average encoding time, most of which is spent on its fractional part (Fractional Motion Estimation - FME), in which sub-pixel samples are interpolated and searched over to find motion vectors with higher precision. This paper presents hardware designs for the sub-pixel interpolation unit of the FME step. The designs employ approximate computing techniques by reducing the number of taps in each filter to reduce memory access and hardware cost. The approximate filters were implemented in the HEVC reference to assess their impact on coding performance. The designed architectures were implemented in VHDL and synthesized with different input sizes in order to assess the effect of this parameter on hardware area and performance. The approximate designs reduce the number of adders/subtractors by up to 67.65% and memory bandwidth by up to 75% with a tolerable loss in coding performance (less than 1% using the Bjontegaard Delta bitrate metric).

KEYWORDS

HEVC, motiom estimatiom, FME, approximate

ACM Reference Format:

. 2018. Approximate Interpolation Filters for the Fractional Motion Estimation in HEVC Encoders. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The use of digital video is one of the biggest trends and one of the main content formats currently being used in several applications. This type of media is now present in many aspects of our daily lives, which pushed the industry into making video playback and recording available in several embedded devices, like smart phones and tablets.

CISCO [6] estimates that the sum of all forms of IP (Internet Protocol) video, which includes Internet video, IP Video on Demand (VoD), video files exchanged through file sharing, video-streamed gaming, and video conferencing, will continue to be in the range of 80 to 90% of the total IP traffic. Globally, IP video traffic will account for 82% of the Internet traffic by 2021. Live Internet video is a particular case in terms of growth rate: it already accounts for 3% of the Internet video traffic and will grow 15-fold to reach 13% by 2021.

To meet these demands, videos are constantly being produced with increased spatial resolution, frame sampling rate and pixel bit width, and all these parameters increase the storage/transmission requirements of this media even in its compressed form. The High Efficiency Video Coding (HEVC) standard [14] emerged as an effort to increase the compression of videos with high resolutions, namely High Definition (HD), which comprises the 1280x720 (720p) and 1920x1080 (1080p) resolutions, and beyond.

The HEVC standard is the most recent joint video project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [7]. Released in 2013, it offers more than 50% improvement in video compression over its predecessor [14], the H.264/AVC (Advanced Video Coding) video coding standard, with the same image quality.

Encoding HEVC videos is a computationally intensive task. This characteristic interferes with real-time applications and encoders embedded in battery-powered devices, as high complexity generates high power consumption. One solution to these problems is the implementation of dedicated architectures, which are capable of high throughput at a much lower energy cost than general purpose processors.

Among the HEVC steps, we can highlight the Motion Estimation (ME) as dominant in terms of computational effort, exceeding 60% of the total encoding time [5]. Most of this time is spent on its fractional search, also called Fractional Motion Estimation (FME), during which sub-pixel blocks are produced and matched with the one being encoded.

FME has an important impact in coding efficiency, as disabling it leads to a 14.5% increase in bitrate [5]. However, this step is very computationally expensive, consuming on average 43% of the encoding time [5]. Therefore, efficient implementations of the FME search are required. As the work [13] which proposes strategy for efficient hardware design and coding energy to FME.

Precise and approximate FME interpolation solutions were already proposed in [9] presents one novel data reusing scheme and highly parallel architecture. In [3] presents a novel reconfigurable hardware architecture for interpolation filtering in HEVC that adapts to run-time changes. In [2] presents an energy-aware and high-throughput hardware design for the FME in HEVC standard, the adopted strategy consists in using only the four square-shaped Prediction Unit (PU) sizes rather than using all 24 possible PU sizes in the ME. In [10] presents a hardware design for the FME Interpolation Unit compatible with HEVC, the architecture was designed to consider fixed 16x16 PU size in order to drastically reduce the computational effort and [8] two approximate HEVC fractional interpolation filters are proposed, respectively.

SBCCI'19, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

With that in mind, the aim of this work is to develop a hardware architecture for the FME HEVC step. The solutions presented in this work comprise filters using sum and shifting in place of multiplication and the approximate interpolation filters for the half-pixels and quarter-pixels luminance samples, three types of approximate interpolation filters are proposed, which are then sent to a block-matching search.

The main contribution of this work compared to what is already published in the literature is the design and synthesis of approximate interpolation filters (F1, F2 and F3) for the fractional motion estimation in HEVC. Both F1, F2 and F3 use a 6-tap, 4-tap and 2-tap filters instead of using an 8-tap filter and two 7-tap FIR filters.

This paper is organized as follows: Section 2 presents an overview of the ME process; Section 3 describes the proposed designed architectures hardware; FPGA synthesis results are presented in Section 4; and Section 5 finally concludes this paper.

2 MOTION ESTIMATION

In HEVC, a frame is divided into Coding Tree Units (CTU) [7], whose size is typically 64x64 pixels. To allow for a better compression of diverse regions in this 64x64 area, each CTU can be partitioned into smaller blocks, called Coding Units (CU).

The structure of CUs is square and their size can be 8x8, 16x16, 32x32, or 64x64 (the entire CTU). This is an improvement w.r.t the previous H.264/AVC, containing only a 16x16 macroblock and smaller subdivisions. A larger and more flexible block structure is effective for encoding high-resolution videos. To decide the best CU size in each region of the CTU, the rate-distortion optimization (RDO) algorithm tests all possible configurations, starting at 64x64 and recursively splitting each node into four smaller ones. This forms a quadtree partitioning structure, depicted in Fig. 1.

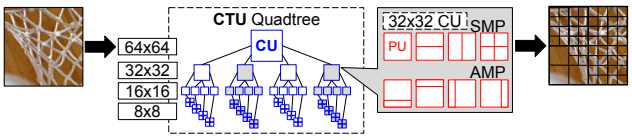


Figure 1: Example of a CTU and its respective CU and PU partitioning decisions. (source: adapted from [4])

During prediction, CUs can assume different dimensions, so a second partitioning structure is necessary. They are called Prediction Units (PUs), and they store the information related to this process, such as prediction mode (inter, intra), motion vectors, reference frame indices etc. Four Symmetric Motion Partitions (SMPs) and four Asymmetric Motion Partitions (AMP) are supported in HEVC. For each PU, the prediction process is executed, which encompasses the inter- and intra-prediction searches. The PU partitions available for a CU depend on its size and its prediction mode [4].

The inter-prediction search contains two major steps: Motion Estimation and Motion Compensation. The Motion Estimation (ME) step is divided into two parts: (i) the Integral Motion Estimation (IME), to identify Motion Vector (MV) based on the original pixels of the image frame, followed by (ii) the Fractional Motion Estimation (FME) to further refine the MV via fractional (interpolated) pixels around the integer pixels. The following section describes the FME, on which this work is focused.

2.1 Fractional Motion Estimation (FME)

HEVC supports MVs with half-pixels and quarter-pixels precision for luminance samples and 1/8 for chrominance ones. The fractional samples can provide a better match between the blocks of the current frame and the frame of reference, increasing compression efficiency.

FME is the most computation-intensive step in the HEVC encoding process, consuming on average 42.9% of the encoding time, from which 15.8% is spent on interpolation and 27.1% on searching [5]. To mitigate this issue, dedicated architectures that accelerate the interpolation and search steps are required.

Given an integer vector from IME as input (MV_{best}), FME is carried out with the following steps:

(i) interpolate half-precision samples in the region pointed out by MV_{best} using an 8-tap interpolation filter
(ii) search the fractional vector of half-precision (MV_H) and update of the best vector ($MV_{best} = MV_{best} << 1 + MV_H$)
(iii) generate quarter-precision samples in the region pointed out by MV_{best} using a 7-tap filter
(iv) search the quarter-precision vector (MV_Q) and update of the best vector ($MV_{best} = MV_{best} << 1 + MV_Q$)

Interpolation (steps (i) and (iii)) is usually performed in three phases: horizontal, vertical, and diagonal samples. These are depicted in Figure 2 as H, V and D. The diagonal samples are interpolated using fractional pixels interpolated in the horizontal phase. This data dependency rules out a fully parallel implementation in VLSI designs.

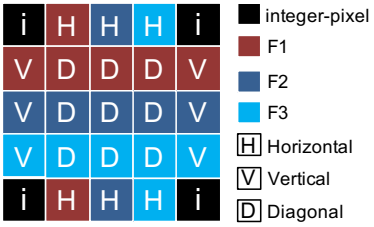


Figure 2: Interpolated area of a 2x2 block.(source: adapted from [11])

In HEVC standard, the luminance samples, half-pixels are interpolated using an 8-tap Finite Impulse Response (FIR) filter, and quarter-pixels with a 7-tap FIR filter. Chrominance pixels are interpolated with a 4-tap filter, according to the desired precision, given by the MV. The coefficients used in each filter are listed in Table 1 and Table 2.

3 DESIGNED ARCHITECTURES

In Figure 3, it is possible to observe the architecture designed for the interpolation responsible for generating the values of the luminance samples in fractional positions. In this way, it is possible to generate one column or one row of fractional samples per cycle. Where N

Table 1: Filter Coefficients for Luma Interpolation

index	-3	-2	-1	0	1	2	3	4
F1	-1	4	-10	58	17	-5	1	
F2	-1	4	-11	40	40	-11	4	-1
F3		1	-5	17	58	-10	4	-1

Table 2: Filter Coefficients for Chroma Interpolation

index	-1	0	1	2
F1	-4	54	10	-2
F2	-4	36	36	-4
F3	-2	54	10	-4

represents the input in pixels and *8* is the size of the edge samples required to calculate samples that are not in the center of the block. This architecture was developed with arbitrary input parallelism so that it was possible to evaluate the impact of the size of the entrance in the area and the performance of the hardware.

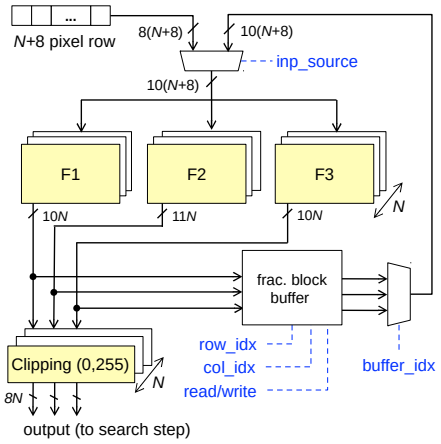


Figure 3: Top-level architecture of the FME interpolation unit for an input size of N integer samples

The hardware that was designed for luma interpolation is purely combinational, so the only registers were introduced at the entrance and exit of the architecture, what consume in a single cycle the input data needed to generate a new interpolated sample, which are capable of generating a sample per clock cycle. The input reference sample is of 8 bit-wide integer pixels, and the input fractional positions are 10 bit-wide. With this, the projected filters are capable of receiving integer and fractional inputs.

Considering 4x4 block size, its data input receives a 1 row of 4 samples from the reference frame, with two lateral edges of 4 samples totaling 12 samples per cycle. In the output, fractional samples are generated from the filters, where each clock cycle generates 15 fractional samples in parallel at the output of the interpolation, with 5 samples of each filter.

In this case, the lines of 12 whole samples side to the sub-pixel samples are delivered in the same order of numbering at the interpolator input. The first 12 lines of resulting samples are of horizontal positions. Lines 13 to 16 are vertical positions and from 17 to 31 are diagonal positions. As shown in the Figure 2.

Then, only the horizontal samples are stored in the buffer, these samples are necessary to generate fractional samples diagonals, as well as on combinational architecture, the buffer has registers on input and output. Considering the size of the 4x4 block, it is necessary to accumulate in the buffer the first 12 horizontal lines. After the buffer is complete he no longer receives samples until the arrival of the next block 4x4.

It is important to note that a 4-pixel padding region is required for each block that is interpolated, because the samples located along the borders must also be interpolated using 4-pixels in each direction.

Filters F1, F2, F3 are designed with approximate architecture, what is introduced in order to reduce the computational complexity of interpolation filters for the FME in HEVC, with a negligible loss of PSNR, by reaching a good trade-off between a good quality of video-coding and energy consumption. The number of taps is dynamically reduced as presented in next.

3.1 Precise FME Filters

Since multiplications represent a higher computational cost with respect to sum, a multiplierless solution is proposed similarly to [1] and [12], in order to shorten the critical path by increasing the throughput.

In this work the filters include optimizations to make multiplications by constant in shift-adds, where the coefficients were represented in powers of 2 and factored. Equations contains a rounding with addition of 32. The sums by constants from the rounding can be simplified with the insertion of +1 in the carry-in of some specific adders, which in this case is 32. The selected combination of shift-adds consisted in the ones with the least use of additions, hence the lowest hardware cost.

All the equations (1)-(6) have been optimized, this optimization allows a reduction in the number of adders when possible, for example in 2-tap filter half-pixels equation (5), only one adder was required for the development of the architecture. In this way, we can get a critical path of at most four adders. In hardware, even if the number of operations are the same, the later the multiplications are carried out the smaller bit width required in the way of adders.

The same adders topologies as presented in this section are applied to the approximate architectures in order to enhance performances. The number of taps is reduced to succeed in this purpose. The proposed approximate interpolation filters are explained next.

3.2 Approximate FME Filters

In this work, three types approximate interpolation filters for the fractional motion estimation (6-tap, 4-tap, 2-tap) are proposed.

The technique used to develop the approximated filters was to remove the coefficients farthest from the center on each side. To maintain the 64 sum, the removed coefficients were added to their

closest remaining neighbors. For instance, to make a 6-tap filter, the coefficients valued -1 on each end were added to the ones valued 4, resulting in two new coefficients whose value is 3. For the 6-tap filter, one coefficient has been zeroed on each side, two coefficients were removed on each side for the 4-tap filter, and three for 2-tap one.

However, with the reduction of the coefficients of the filters, which are shown in Table 3. With this technique the circuit becomes simpler and require less memory accesses than the fractional interpolation HEVC original hardware as is shown in Table 4. In Figure 4 shows the Memory access of implemented filters for each input size (N), to notice the reduction of access to memory.

Table 3: Approximate Filter Coefficients

	index	-3	-2	-1	0	1	2	3	4
6 taps	F2		3	-11	40	40	-11	3	
	F1, F3		3	-10	58	17	-5	1	
4 taps	F2			-8	40	40	-8		
	F1, F3			-7	58	17	-4		
2 taps	F2				32	32			
	F1, F3				51	13			

Table 4: Approximate filters memory access reduction for each input size (N)

N	6 taps	4 taps	2 taps
4	31%	56%	75%
8	23%	44%	61%
16	16%	31%	44%
32	10%	19%	28%

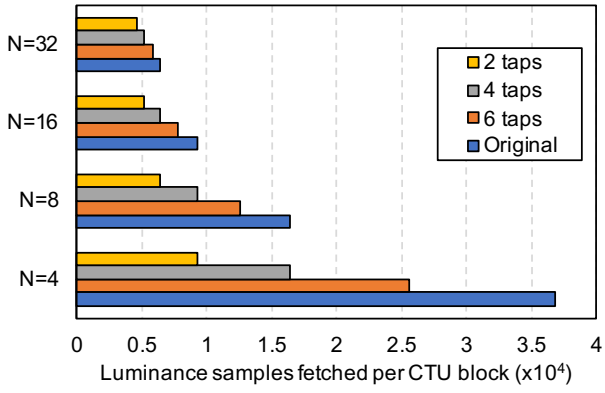
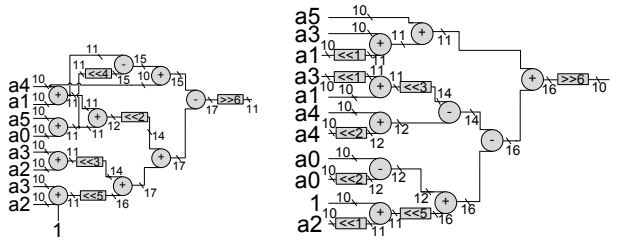


Figure 4: Memory access of implemented filters for each input size (N)

The following is shown approximate fractional interpolation filter equations HEVC proposed for 6-tap, 4-tap and 2-tap are shown in (1) - (6) representing half-pixels and quarter-pixels, according to the coefficients shown in the Table 3. The proposed 6-tap, 4-tap and 2-tap approximate interpolation filters for the FME hardware are shown in Figure 5 - 7.



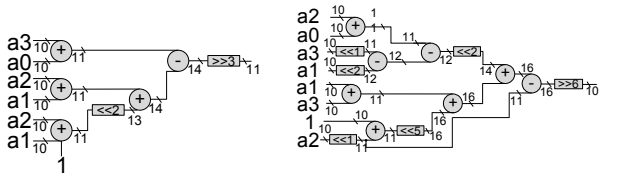
(a) F2 (b) F1

Figure 5: Proposed 6-tap approximate interpolation filter for the FME hardware. (a) F2 and (b) F1 and F3

3.2.1 6-tap Approximate Filters.

$$F2 = 32(a_2 + a_3 + 1) + 4[(a_1 + a_4) + (a_0 + a_5)] + \\ + 8(a_2 + a_3) - 16(a_1 + a_4) - (a_0 + a_5) + a_5 >> 6 \quad (1)$$

$$F1 = 32(2a_2 + 1) + [(4a_0 - a_0) - (4a_4 - a_4)] - \\ - 8(a_1 + 2a_3) + 2a_1 + a_3 + a_5 >> 6 \quad (2)$$



(a) F2 (b) F1

Figure 6: Proposed 4-tap approximate interpolation filter for the FME hardware. (a) F2 and (b) F1 and F3

3.2.2 4-tap Approximate Filters.

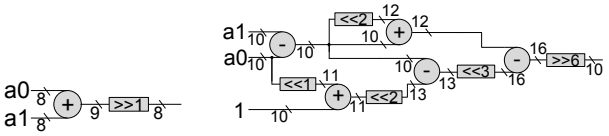
$$F2 = 8[4(a_1 + a_2 + 1) + (a_1 + a_2) - (a_0 + a_3)] >> 6 \quad (3)$$

$$F1 = 32(2a_1 + 1) + 4[(4a_2 - a_0) - (a_1 + a_3)] + \\ + (a_0 + a_1) - 2a_1 >> 6 \quad (4)$$

3.2.3 2-tap Approximate Filters.

$$F2 = 32(a_0 + a_1 + 1) >> 6 \quad (5)$$

$$F1 = 8[4(2a_0 + 1) - (a_0 - a_1)] - \\ - (a_0 - a_1) + 4(a_0 - a_1) >> 6 \quad (6)$$



(a) F2 (b) F1

Figure 7: Proposed 2-tap approximate interpolation filter for the FME hardware. (a) F2 and (b) F1 and F3

Table 5: Experimental setup used to obtain the synthesis and compression results

FPGA tool	Quartus Prime vX.X
device	Cyclone GX
Standard Cell tool	Cadence Genus Synthesis Solution
library	ST 65 nm
optimization effort	medium
input activity	default (0.2)
HEVC software	HM v16.2
GOP structure	Random Access
frame count	50

4 RESULTS AND DISCUSSION

4.1 Synthesis Results

FPGA implementation results are shown in Table 6.

Table 6: FPGA device

	N=4			
	8-tap	6-tap	4-tap	2-tap
Slow 1200mV 85C (MHz)	78.75	74.15	79,.59	91.17
Slow 1200mV 0C (MHz)	88.21	82.78	89.38	101.54
LC combinationals	3458	3384	2314	1622
LC registers	1992	1776	1440	1104
	N=8			
	8-tap	6-tap	4-tap	2-tap
Slow 1200mV 85C (MHz)	63.91	61.48	63.14	75.24
Slow 1200mV 0C (MHz)	70.77	67.39	69.77	83.06
LC combinationals	6883	6697	4864	3556
LC registers	4512	4224	3648	3072

4.2 Compression Efficiency

The proposed approximate interpolation filter for the FME (6-tap, 4-tap and 2-tap) are integrated into FME in HEVC HM v16.2 software encoder, what are coded with four different quantization parameters (QP) and 50 frames using HEVC HM with original HEVC fractional interpolation filters. The resulting rate-distortion performances are shown in Table 7.

In Figure 8 shown the decoded picture comparison of each filter precision for typical $QP = 22$ and $QP = 32$ values.

Table 7: BD-rate results of each approximate filter using Random Access, QPs=(22, 27, 32, 37), and 50 frames

sequence	resolution	BD-rate (6 taps)	BD-rate (4 taps)	BD-rate (2 taps)
PeopleOnStreet	2560x1600	-0.03	0.27	0.44
Traffic	2560x1600	0.07	0.27	0.38
BasketballDrive	1920x1080	0.15	0.18	0.61
BQTerrace	1920x1080	0.00	0.14	0.62
Cactus	1920x1080	0.02	0.18	0.74
Kimono1	1920x1080	-0.03	0.17	0.12
ParkScene	1920x1080	0.11	0.28	0.59
BasketballDrill	832x480	-0.08	0.15	0.79
BQMall	832x480	0.03	0.23	0.96
PartyScene	832x480	0.02	0.17	1.27
RaceHorsesC	832x480	0.09	0.34	1.07
BasketballPass	416x240	0.00	0.36	1.39
BlowingBubbles	416x240	-0.13	0.24	1.53
BQSquare	416x240	0.03	0.28	1.28
RaceHorses	416x240	-0.04	0.47	1.59
	average	**0.02**	**0.25**	**0.89**

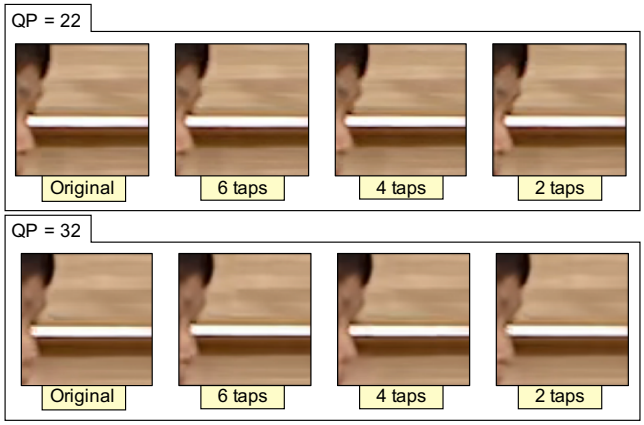


Figure 8: Decoded picture comparison of each filter precision for typical QP values (sequence: BasketballDrive (1920x1080), frame 33, CTU position = (576,320)).

4.3 Comparison with related work

Approximate HEVC fractional interpolation filters and their hardware implementations is proposed in [8], two approximate HEVC fractional interpolation hardware are proposed. Both use one 4-tap and two different 3-tap FIR filters. In this paper they significantly reduce computational complexity of the standard HEVC fractional interpolation by modifying the interpolation filter coefficients at the expense of a negligible PSNR loss and bit rate increase.

The proposed approximate interpolation filters hardware are compared with the HEVC fractional interpolation hardware proposed in the literature [9], [3], [2], [10] and [8]. The comparisons of FPGA implementations are shown in Table 8.

Table 8: Comparison of FPGA implementations with related works

	[9]	[3]	[2]	[10]	[8]	proposed (4-taps)	proposed (2-taps)
FPGA	65 nm	65 nm	28 nm	65 nm	28 nm	60 nm	XX
Slice Count	N/A	2181	13,235	N/A	7,698	XXX	XX
LUT Count	28486	2008	12,031	7,701	3,162	XXX	XX
Max. Freq. (MHz)	120	283	396.8	353.8	259	XXX	XX
Fps	N/A	30@2560x1600	60@3849x2160	60@3849x2160	60@3849x2160	XXXXX	XX
Power Dissipation	N/A	89 mW	N/A	N/A	N/A	XXXX	XX
BR-rate loss	N/A	N/A	4.04	19.36	0.527	XXXX	XX

5 CONCLUSION

ACKNOWLEDGEMENT

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de NÃŋvel Superior - Brasil (CAPES) - Finance Code 001 and also by FAPERGS and CNPq Brazilian research support agencies.

REFERENCES

[1] V. Afonso, H. Maich, L. Agostini, and D. Franco. 2013. Low cost and high throughput FME interpolation for the HEVC emerging video coding standard. In *2013 IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS)*. 1–4. https://doi.org/10.1109/LASCAS.2013.6519017

[2] Vladimir Afonso, Henrique Maich, L Audibert, Bruno Zatt, Marcelo Porto, Luciano Agostini, and Altamiro Susin. 2016. Hardware implementation for the HEVC fractional motion estimation targeting real-time and low-energy. 11 (08 2016), 106–120.

[3] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel. 2015. A Reconfigurable Hardware Architecture for Fractional Pixel Interpolation in High Efficiency Video Coding. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 2 (Feb 2015), 238–251. https://doi.org/10.1109/TCAD.2014.2384517

[4] Mateus Grellert. 2018. *Machine learning mode decision for complexity reduction and scaling in video applications.* Ph.D. Dissertation. Federal University of Rio Grande do Sul, Stanford, CA, USA. Advisor(s) Bruno Zatt and Sergio Bampi.

[5] M. Grellert, S. Bampi, and B. Zatt. 2016. Complexity-scalable HEVC encoding. In *2016 Picture Coding Symposium (PCS)*. 1–5. https://doi.org/10.1109/PCS.2016.7906356

[6] Cisco Visual Networking Index. 2014. Forecast and methodology, 2013–2018. *Cisco white paper* (2014), 154.

[7] ITU-T/ISO/IEC. 2013. TU-T Recommendation H.265 and ISO/IEC 23008-2. (2013).

[8] E. Kalali and I. Hamzaoglu. 2018. Approximate HEVC Fractional Interpolation Filters and Their Hardware Implementations. *IEEE Transactions on Consumer Electronics* 64, 3 (Aug 2018), 285–291. https://doi.org/10.1109/TCE.2018.2867806

[9] C. Lung and C. Shen. 2014. A high-throughput interpolator for fractional motion estimation in high efficient video coding (HEVC) systems. In *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 268–271. https://doi.org/10.1109/APCCAS.2014.7032771

[10] H. Maich, V. Afonso, D. Franco, B. Zatt, M. Porto, and L. Agostini. 2013. High throughput hardware design for the HEVC Fractional Motion Estimation Interpolation Unit. In *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*. 161–164. https://doi.org/10.1109/ICECS.2013.6815379

[11] Guilherme Paim. 2015. *Arquitetura Multipadrão para a Estimaĉão de Movimento Fracionària para os padrões HEVC e H.264/AVC*. Master's thesis. Federal University of Pelotas. Advisor(s) Bruno Zatt and Denis Teixeira Franco.

[12] W. Penny, G. Paim, M. Porto, L. Agostini, and B. Zatt. 2015. Real-time architecture for HEVC motion compensation sample interpolator for UHD videos. In *2015 28th Symposium on Integrated Circuits and Systems Design (SBCCI)*. 1–6.

[13] I. Seidel, V. Rodrigues Filho, L. Agostini, and J. L. GÃijntzel. 2018. Coding- and Energy-Efficient FME Hardware Design. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. https://doi.org/10.1109/ISCAS.2018.8351114

[14] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (Dec 2012), 1649–1668. https://doi.org/10.1109/TCSVT.2012.2221191