Power-Efficient and Memory-Aware Approximate Hardware Design for HEVC FME Interpolator

Wagner Penny[1,2], Mariana Ucker[1,2], Italo Machado[1], Luciano Agostini[1], Daniel Palomino[1], Marcelo Porto[1], Bruno Zatt[1]
Video Technology Research Group - Group of Architectures and Integrated Circuits - Computer Science Graduate Program
[1]Federal University of Pelotas – [2]Sul-rio-grandense Federal Institute
Pelotas, Brazil
{wi.penny, mariana.ucker, idmachado, agostini, dpalomino, porto, zatt}@inf.ufpel.edu.br

***Abstract*—Challenges related to mobile video coding systems are of utmost importance for researchers, especially regarding issues posed by the limited battery capacity and computational resources. The current state-of-the-art video coding standard is the High Efficiency Video Coding (HEVC), which implements computationally intensive coding tools, such as the Fractional Motion Estimation (FME). As a way to deal with real-time constraints, hardware acceleration is necessary. Furthermore, the usage of hardware accelerators allied to approximate computing is also an alternative to improve performance and save power/energy. In this work, we present a power-efficient and memory-aware approximate configurable hardware design for the HEVC FME interpolator. The proposed design is capable of real-time interpolation of UHD 4K and 8K (Ultra High Definition - 2160p@60fps and 4320@60fps) videos when synthesized using a 45nm standard-cell library, with a power dissipation ranging from 9.78mW to 50.4mW.**

Keywords—FME; approximate computing; hardware design; HEVC; video coding

I. INTRODUCTION

Over the last years, the number of mobile devices capable of capturing, manipulating, storing, and transmitting digital videos has grown meaningfully. The video broadcasting is responsible for an Internet traffic that exceeded 7 EiB/month in 2016, with an expectation of reaching up to 49 EiB/month in 2021 [1]. In fact, according to Cisco [2], in 2021, 80-90% of all global Internet traffic will be related to video transmission. Besides those features, the need for ever-increasing resolutions and compression ratios, have led to the development of new video coding standards, such as the High Efficiency Video Coding (HEVC) [3]. This standard was released in 2013, becoming the state-of-the-art video coding standard, replacing its predecessor, the H.264. HEVC is capable of reaching around 40% lower bit rate considering the same video quality compared to H.264 [4].

However, these higher compression ratios were obtained by the HEVC at the cost of reaching sizable complexity values, mainly caused by the use of more intensive processing tools, such as the Fractional Motion Estimation (FME), which is responsible for more than 49% of the total HEVC encoding time [5]. By applying an interpolation over the integer samples from the reference frame (brought from main memory), FME generates sub-pixels (fractional) samples, allowing finer grain motion estimation (ME). This interpolation process uses 7- or 8-tap FIR (Finite Impulse Response) filters, as defined by the HEVC standard [3]. To quantify the complexity, when compared with H.264, the HEVC computational effort increase varies from 10% up to 500%, depending on the resolutions and encoder configurations [6]. This increase in computational effort leads also to great energy consumption and intensive memory communication. For instance, the HEVC may require up to three times more memory accesses when compared with H.264 encoder [7].

Common software implementations are not able to deal with such complex tasks (e.g., FME), not being able to give the necessary throughput required by real-time applications. As a way to address such computational costs and deal with real-time constraints, typical systems-on-chip (SoCs) for mobile devices have been applying different alternatives, as the use of dedicated hardware and the usage of approximate computing. Recently, the concept of approximate computing has gained a lot of research attention and has been seen as an alternative to improve performance/power efficiency by compromising (in a tolerable range) the quality of the applications [8].

Generally, video coding is a suitable application to develop hardware implementations, aiming to deal with complex tasks, whereas it is also suitable to apply approximate computing due to the possibility of exploration of the trade-off between a scalable final video quality and another variable (e.g., power dissipation). Still, the use of approximate computing can be allied to the design of specific hardware accelerators, e.g., for FME. Besides providing a reduction on energy consumption, the development of an approximate architecture may favor a reduction in memory communication, since the implemented simplifications can reduce the amount of information needed to carry out the same process.

In this work, we present a power-efficient and memory-aware approximate hardware design for HEVC FME interpolator. Approximate computing was applied by changing the number of filters taps in a configurable way, leading to an efficient hardware implementation and considerable reduction on memory communication required from main memory.

II. BACKGROUND AND RELATED WORKS

The FME is a search based on fractional (sub-pixel) positions, which happens after the Integer Motion Estimation (IME). The process consists of an interpolation among integer luminance samples, generating the fractional samples presented in Fig. 1. These samples are calculated using FIR filters (7- or 8-taps on HEVC), whose coefficients values are also presented in Fig. 1. These filters, as stated in [9], are called *UP*, *MIDDLE* or *DOWN*, according to the samples they calculate. The generation

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance code 001, the Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq), and the Fundação de Amparo à Pesquisa do Rio Grande do Sul - Brasil (FAPERGS).

of sub-pixel samples is done by performing an interpolation over the integer samples, which consists of two steps: the horizontal filtering and the vertical filtering. During the first step, samples *a*, *b*, and *c* are calculated by the usage of neighboring integer samples in a row of the reference matrix. Next, the second step consists of a vertical interpolation, using neighboring integer samples to calculate *d*, *h*, and *n*, or using the previously pre-calculated fractional samples (*a*, *b*, and *c*), to calculate the remaining ones.

Some works in the literature propose hardware designs to deal with very complex tasks of the HEVC encoder, such as inter-frame prediction. Work [9] proposed a hardware architecture for the HEVC FME interpolator reaching high throughput. In a further improvement of their work [10], the authors developed a hardware design for the whole FME, capable of reducing memory communication and computational effort by the use of only four sizes of square-shaped prediction units (PUs), instead of using the 24 sizes supported by HEVC motion estimation. Besides the reduction of memory accesses and computational effort, it does not support other PU sizes and significantly affects the compression efficiency. In [11] is proposed a high-throughput architecture for HEVC FME, also involving all FME steps. Besides reaching complexity reductions, it does not show a complete evaluation of impacts of the adopted complexity-reduction strategy whereas it significantly affects coding efficiency. Work [12] proposes a high-throughput architecture for HEVC FME interpolator. Besides reaching computational effort reduction, its solution supports only 16x16 PUs affecting the coding efficiency.

Many works introduce approximate computing to develop architectures targeting specific accelerators for HEVC inter-prediction blocks. The work [13] has proposed the development of efficient architectures for HEVC luminance interpolation filters. It applies approximate computing by changing the number of adders in the design, resizing the filter coefficients according to a proposed scaling factor modification to reduce hardware implementation complexity. In work [14], a runtime coarse-grained adaptation solution is developed to guarantee energy reduction, in constraint-aware or user-defined situations, while introducing a controllable quality degradation by approximate computing, targeting Motion Compensation (MC) HEVC filters. Although they have obtained expressive gains in performance and area, with small quality degradation, they only act at the interpolation filters, not involving the whole interpolator unit. Moreover, they focus only on the MC tool, at the decoder side, not targeting the FME step.

Some of the related works do not apply approximate computing. Other related works, even though presenting approximate hardware designs, do not evaluate the coding efficiency. Moreover, none of the related works have proposed an approximate hardware design for the HEVC FME interpolator. Our work proposes a power-efficient and memory-aware approximate hardware design for HEVC FME interpolator, being a novel configurable architecture by resizing the number of filter taps. Thus, fewer samples need to be read from main memory during the fractional samples calculation.

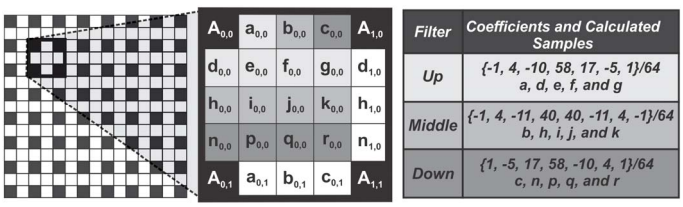

Fig. 1. Fractional-position Samples (lower-case letters) and Integer-position Samples (upper-case letters).

III. Approximate Filters Configurations Evaluation

The HEVC standard supports 25 possible PU sizes and the FME must support 24 of them in order to search all possible positions and find the best matching (FME does not evaluate 4x4 PUs). Although hardware designs are capable of improving performance, it would be impracticable to provide support for all possible PU sizes with a hardware architecture. Work [15] has introduced the concept of the use of a Basic Processing Unit (BPU), showing that any PU could be processed using a fixed BPU size. Eventually, this leads to process overhead, which happens on different levels depending on the BPU size. They demonstrated that the best BPU size for the HEVC is 8x8 (used in this work), taking into account the trade-off among samples to process, line-level parallelism, and memory communication.

The interpolation of an 8x8 block requires reading a bigger reference matrix from the main memory, containing the sample borders. Generally, filters with *n-taps*, where *n* is an even number, require *n-1* samples per border. Considering the worst case, when an 8-tap filter is used, for an 8x8 block is necessary to bring a 15x15 reference (input) matrix (here called *Legacy*, presented in Fig. 2). Thus, changing the maximum number of taps in a filter mainly affects the number of samples needed to be read from external memory. Fig. 2 also presents our main case study, here called *LRTR* (*Left-Right Taps Removed*), when both left and right taps are removed, which leads to the smallest evaluated reference matrix size (13x13). In this case, fewer samples compound the input matrix, which requires less intensive memory communication (*LRTR* avoids the transmission of 56 samples when compared with the *Legacy*, representing 24.89% memory bandwidth saving).

We performed simulations testing four filter configurations. Besides the mentioned *Legacy* and *LRTR* cases, we also tested the removal of the most external left tap (*LTR – Left Tap Removed*) and the most external right tap (*RTR – Right Tap Removed*), using the reference software of the HEVC, the HM 16.18 (HEVC Test Model) [16], according to the Common Test Conditions (CTCs) [17], established by this video coding standard. Our simulations take into account 200 frames from each of the 24 recommended video sequences and recommended Quantization Parameters (QP) – 22, 27, 32, and 37. The evaluation of the coding efficiency for this different set of cases was made using a metric called *Bjontegaard Difference* (BD)

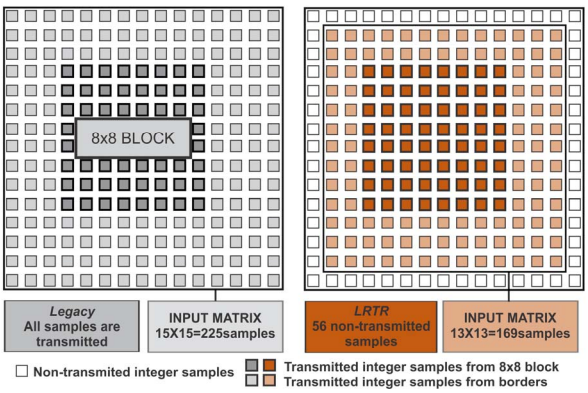

Fig. 2. Reference matrices scenarios.

[18], considering the BD-BR (BD-Bit Rate), which is widely used by the academic and standardization community to fairly measure the coding efficiency/quality losses. *LRTR* case gave the greatest memory communication savings, affecting BD-BR in only 0.527% in average, face off 0.619% and 0.618% for *LTR* and *RTR* cases, respectively. Therefore, *LTR* and *RTR* are not suitable to the developed architecture, since they affect more heavily the coding efficiency and do not reduce memory traffic as much as *LRTR*.

It is worth to notice that the changes implemented on filters design were applied only on FME step, which belongs to the encoder side. Regarding the development of video coding standards, the standardization is mandatory at the decoder side, in order to be compliant with the standard. The works that implement changes on the decoder side, affect the video quality. However, the decoder steps existing inside the encoding flow (e.g., MC), must not be altered, under penalty of making the encoded video not compliant with the standard or inserting encoder-decoder drifting.

IV. Interpolator and Filters Design

The architecture of the interpolator proposed in this work is presented in Fig. 3 (a). This solution contains eight horizontal filters (**H-Filter**) and eight vertical filters (**V-Filter**). The input of the architecture is a row of the reference matrix (which can be configurable having 15 or 13 integer samples, according to what was presented in Fig. 2), read at each clock cycle. Thus, for instance, if the row has 13 samples, the matrix will spend 13 cycles to be read, and so on.

Each horizontal filter has internally three FIR filters: UP (**U**), MIDDLE (**M**), and DOWN (**D**). They calculate the fractional samples detailed in Fig. 1 (*a*, *b*, and *c*), respectively. In addition, there is a simple connection passing directly the integer sample. The outputs of the horizontal filters feed four position shift registers. On each cycle, each position of the shift registers is forwarded. After eight cycles, all positions are completely filled. So, all their positions feed the vertical filters. These ones are similar to horizontal filters. However, since at this time the number of input samples is four times greater, four types of each FIR are needed. Inside these filters, all the other fractional samples are calculated. In addition, integer samples (*A*), and the fractional samples (*a*, *b*, and *c*), pass directly through the filters. The vertical filter output is always a set of 16 samples (previously shown in Fig. 1). Finally, the interpolator output is the number of fractional samples related to an 8x8 block.

The horizontal and vertical filters used in the development of the interpolator can use up to eight of the fifteen samples received by the whole architecture. Internally, they are designed to make the interpolator architecture reconfigurable and can receive two sets of samples as input (eight or six for the **M** filter, seven or five for the **U**/**D** filters).

The architecture of the UP and DOWN filters, used by the vertical and horizontal filters, is the same (joined as UP/DOWN), only the sequence of the input samples need to be inverted, according to what is presented in Fig. 1. Fig. 3 (b) and Fig. 3 (c) present the architectures developed for the UP/DOWN and MIDDLE filters, respectively. As can be seen in Fig. 3 (b) and Fig. 3 (c), the filters has a control variable (Ctrl), which enables the data gating, allowing the implementation of *Legacy* or *LRTR* case. The **U**/**D** and **M** filters architectures were optimized by replacing the multipliers with adders and shifters. They have an output with variable bit depth, depending on the number of samples ("s") received in the input. For this reason, the outputs of the horizontal and vertical filters also have variable bit depth (in the worst case, 10 bits for **U**/**D** and 11 bits for the **M** filter).

Both **U**/**D** and **M** filters have three pipeline stages between the adder/subtractors (omitted in the figures in order to optimize the representation). In the worst case, the system latency is 14 cycles (eight from the shift registers and six from internal filters pipelines). After that, supposing a continuous sample input, every 15 or 13 clock cycles, 1024 samples are delivered, depending on the architecture configuration. Giving a variable throughput of 68.26 or 78.77 samples per cycle.

V. Results and Discussion

The proposed architecture was described in VHDL and mapped to an Altera Stratix V 5SGXEA7H3F35C3 device. It was also synthesized using a 45nm Nangate standard cell library with the Cadence RTL Compiler tool. The operating frequencies were defined as the minimum frequencies required to achieve real-time processing on different UHD video resolutions. The designed approximate interpolator was evaluated by considering the area and power dissipation results and by focusing on performance ranging from UHD 2160p@60fps – (4K) resolution to UHD 4320p@60fps - (8K) resolution.

Tables I and II present the synthesis results, targeting FPGA and ASIC technologies, respectively, regarding the worst cases scenarios, comparing the developed work with related works. Regarding the ASIC design, the gate count was calculated based

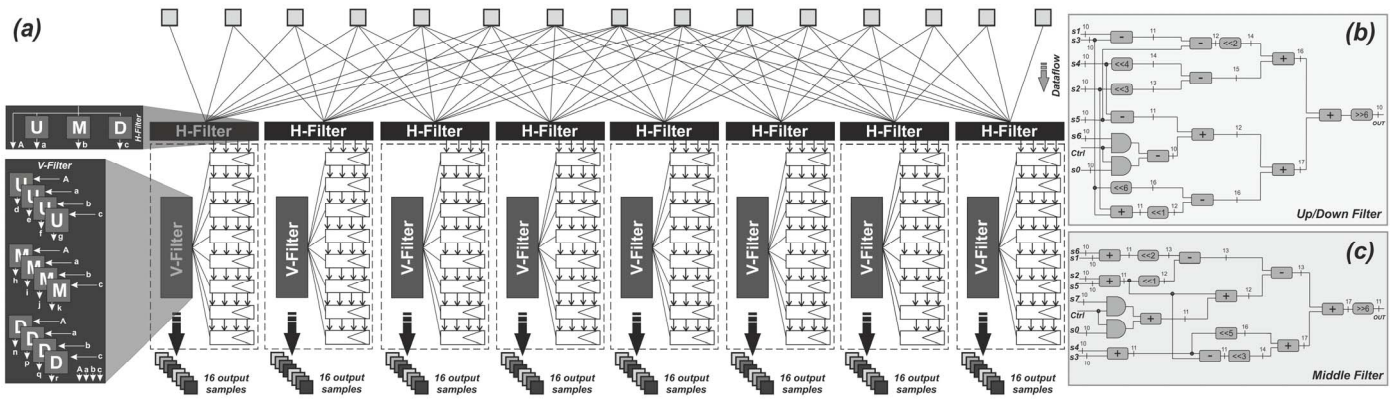

Fig. 3. Configurable HEVC FME Architecture: (a) Interpolator Architecture (b) Up/Down Filter (c) Middle Filter.

TABLE I. SYNTHESIS RESULTS AND COMPARISON FOR FPGA

Parameter	Related Works				This Work
	[9]	[10]	[12]	[14]	
Max. Power (mW)	-	-	-	125	-
Max. Throughput	2160p @60fps	2160p @60fps	2160p @60fps	2160p @60fps	2160p @60fps
Max. Freq. (MHz)	403.06	396.8	353.8	172	259
Logic Cells (ALMS or ALUTS)	4,077 ALUTS	12,031 ALUTS	7,701 ALUTS	1,559 ALUTS	3,162 ALMS
Registers	3,861	13,235		-	7,698
FPGA Device	Stratix III	Stratix V	Stratix III	Artix 7	Stratix V
BD-BR (%)	22.52	4.04	19.36	NA	0.527

TABLE II. SYNTHESIS RESULTS AND COMPARISON FOR ASIC

Parameter	Related Works			This Work@4K		This Work@8K	
	[10]	[11]	[13]	*Legacy*	*LRTR*	*Legacy*	*LRTR*
Max. Power (mW)	15.85	48.3	-	15.06	9.78	50.4	32.36
Max. Throughput	2160p @60fps	2160p @30fps	2160p @60fps	2160p @60fps		4320p @60fps	
Max. Freq. (MHz)	396.8	95	689	116.64		466.56	
Gate Count (K)	148.41	1,183.00	0.99	87.9			
Technology	Nangate 45nm	65nm*	TSMC 65nm	Nangate 45nm			
BD-BR (%)	4.04	2.07	0.883	0.527			

*Library not mentioned

on 2-input NANDs count and the power dissipation was generated with a power supply of 1.1V.

Considering approximate hardware designs, it is possible to highlight that only two works reach enough throughput to give real-time performance: our work and [13]. However, besides reaching the highest frequency among the related works and the smallest area, they designed only the approximate filters architectures, not the whole interpolator. In addition, [13] focus only in MC and does not calculate all fractional samples (as is required by the FME, and significantly increases area and power). Additionally, even though they develop an ASIC solution, power results are not presented. It makes impossible to do a fair comparison. In addition, such work presented worse BR-BR results than our solution. Although bringing forth an interesting approximate hardware for the interpolator, reaching the same throughput of our FPGA design, work [14] focus only on the MC and reaches greater values of power dissipation.

Regarding all related works, [9] and [12] are the ones that developed only the FME interpolator hardware design (as is done by our work). In [10] and [11] whole FME designs are proposed, including the search and comparison step (thus, their area is the greatest among all works). Considering these works, our solution is uniquely capable of reaching enough throughput to deal with UHD videos 8K in a real-time environment, while presenting the best coding efficiency values.

When mapped to FPGA, our design reached 259MHz, which makes the hardware architecture capable of processing 4K videos at 60 fps. Our ASIC solution reached the required performance to deal with 4K and 8K videos, while dissipating a power of 15.06mW and 50.4mW, respectively, in *Legacy* case, and dissipating a power of 9.78mW and 32.36mW, in *LRTR* case. Such reduction in dissipated power happened due to the usage of fewer operators (data gating). In addition, our solution is the sole one, among the related works, capable of saving up to 25% main memory bandwidth, by reducing considerably the number of samples read from memory.

VI. CONCLUSIONS

This paper presented a power-efficient and memory-aware approximate hardware design for the HEVC FME interpolator. Using a fixed block size as a basic processing unit, the whole interpolator was designed with FIR filters having a configurable number of taps. The developed approximate interpolator architecture was synthesized using a 45nm Nangate standard cell and mapped to an Altera Stratix V FPGA. This work was the unique approximate hardware for the HEVC FME interpolator among the related works. The developed architecture, in *LRTR* case, is able to process 2160p@60fps with a power of 9.78mW, and 4320p@60fps with a power dissipation of 32.36mW.

ACKNOWLEDGMENT

We have a special acknowledgment to CNPq, CAPES, and FAPERGS to support this work.

REFERENCES

[1] Statista, “Statistics and facts on mobile internet usage, 2016,” [Online]. Available: https://www.statista.com/topics/779/mobile-internet/. [Accessed: 27-Jun-2018].

[2] Cisco, “Forecast and Methodology, 2016–2021,” [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. [Accessed: 01-Mar-2018].

[3] ISO/IEC-JCT1/SC29/WG11, “High Efficiency Video Coding (HEVC) text specification draft 10.” doc. JCTVC-L1003, 2013.

[4] D. Grois et al., “Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders,” in *2013 PCS*, 2013, pp. 394–397.

[5] H. Li, Y. Zhang, and H. Chao, “An optimally scalable and cost-effective fractional-pixel motion estimation algorithm for HEVC,” in *2013 IEEE ICASSP*, 2013, pp. 1399–1403.

[6] L. Mengzhe, J. Xiuhua, and L. Xiaohua, “Analysis of H.265/HEVC, H.264 and VP9 coding efficiency based on video content complexity,” in *2015 IEEE ICCC*, 2015, pp. 420–424.

[7] M. Shafique and J. Henkel, “Low power design of the next-generation High Efficiency Video Coding,” in *2014 ASP-DAC*, 2014, pp. 274–281.

[8] S. Venkataramani et al., “Computing Approximately, and Efficiently,” in *Proceedings of the 2015 DATE*, pp. 748–751.

[9] V. Afonso et al., “Low cost and high throughput FME interpolation for the HEVC emerging video coding standard,” in *IEEE 4th LASCAS*, 2013.

[10] V. Afonso et al., “Hardware implementation for the HEVC fractional motion estimation targeting real-time and low-energy,” *J. Integr. Circuits Syst.*, vol. 11, no. 2, pp. 106–120, 2016.

[11] G. He et al., “High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding,” *IEEE TVLSI*, vol. 23, no. 12, pp. 3138–3142, 2015.

[12] H. Maich et al., "High throughput hardware design for the HEVC Fractional Motion Estimation Interpolation Unit," *IEEE ICECS*, Abu Dhabi, 2013, pp. 161-164.

[13] A. Diefy et al., “Efficient architectures for HEVC luma interpolation filter,” in *Proceedings of ICM*, 2016, pp. 9–12.

[14] F. Palumbo et al., “Runtime Energy versus Quality Tuning in Motion Compensation Filters for HEVC,” *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 145–152, 2016.

[15] W. Penny et al., “Real-time architecture for HEVC motion compensation sample interpolator for UHD videos,” *SBCCI*, 2015, pp. 1-6.

[16] J. Boyce, “HM16: High Efficiency Video Coding Test Model (HM16) Encoder Discription.” JCTVC-R1002, Sapporo, 2014.

[17] F. Bossen, “Common test conditions and software reference configurations.” JCTVC-L1100, Geneva, 2011.

[18] G. Bjontegaard, “Improvements of the BD-PSNR model.” ITU-T, Berlim, 2008.