Approximate Hardware Architecture for the Interpolation Filters of Versatile Video Coding

Giovane G. Silva[1], Ícaro G. Siqueira[1], Mateus Grellert[2], Cláudio M. Diniz[3]

[1] Graduate Program on Electronic Engineering and Computing, Catholic University of Pelotas (UCPel), Pelotas, Brazil
[2] Embedded Computing Lab, Federal University of Santa Catarina (UFSC), Florianópolis, Brazil
[3] Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
e-mail: cmdiniz@ieee.org

***Abstract*— The new Versatile Video Coding (VVC) standard was recently developed to improve compression efficiency of previous video coding standards and to support new applications. The compression efficiency gain was achieved in the standardization process at the cost of an increase in the computational complexity of the encoder algorithms, which leads to the need to develop hardware accelerators and to apply approximate computing techniques to reach the performance and power dissipation required for systems that encode video. This work proposes the implementation of an approximate hardware architecture for interpolation filters defined in the VVC standard targeting fractional motion estimation requirements of real-time processing of high resolution videos scenario. The architecture includes four filter cores in parallel, each one generating 15 fractional per clock cycle, so it calculates 60 fractional pixels in parallel. Each filter core is based on approximating the original 8-tap and 7-tap interpolation filters defined in the VVC standard to 6-tap interpolation filters, and by applying Multiple Constant Multiplication (MCM) algorithm to optimize filter datapaths. The architecture is able to process up to 2560×1600 pixels videos at 30 fps with power dissipation of 23.9 mW when operating at a frequency of 522 MHz, with an average compression efficiency degradation of only 0.41% compared to default VVC video encoder software configuration.**

***Index Terms*— Video coding; Versatile Video Coding; Interpolation Filter; Hardware; Architecture.**

I. Introduction

Digital video is widespread into many electronic devices, enabling a diversity of applications such as video on demand, digital television, video surveillance, etc. There is a growing demand for digital video, which is explained by the increased number of devices: a forecast by Cisco points out that by 2023 the number of devices connected to Internet Protocol (IP) networks will be more than three times the global population [1]. The huge demand for digital video and the raise of video resolutions and frame rates pushes the internet data traffic related to video transmission. By 2023, 66% of flat-panel TVs will support Ultra-High-Definition (UHD) or 4K resolution (3840×2160 *pixels*). It results in an increase of video traffic over the Internet. Today video traffic share is about 80% of total Internet traffic and it continues to grow for the next years [2].

Given these demands, and the market need for applications with even higher visual quality, videos are constantly produced with higher spatial resolution, higher bit depth and higher frame sampling rate, increasing storage/transmission requirements. A video with a resolution of 1920×1080 *pixels* with 30 frames per second (fps), when *pixels* are represented with 24 *bits*, produces a bit rate of 186.6 *MB* per second. To store 1 hour of this video would require 671.8 GB of storage space. UHD 4K videos increase bit rate and storage space requirements by $4\times$ compared to HD video. Thus, it becomes unfeasible to use such raw video representation, with motivates the need for video compression.

The Versatile Video Coding (VVC) [3, 4] standard was recently developed by the International Telecommunication Union (ITU) Video Coding Experts Group (VCEG) and International Organization for Standardization (ISO) Motion Picture Experts Group (MPEG) to increase compression efficiency compared to previous VCEG/MPEG standard High Efficiency Video Coding (HEVC) [5], and to be versatile to support different video applications, e.g. high dynamic range, screen content, multiview, and 360-degree videos. As reported by [4], VVC offers bit rate savings of about 50% compared to HEVC for equal subjective quality. However, this comes with an impact on the computational complexity required to encode videos. The processing time of the VVC encoder software is 10.2 times higher than HEVC encoder (on average for different videos) when Single Instruction Multiple Data (SIMD) instructions are enabled, and this cost increases by 15.9 times when SIMD instructions are disabled [6].

Motion Estimation (ME) stands out as one of the most computing-intensive parts in modern encoders. This step is commonly composed of an integer motion estimation (IME) and fractional motion estimation (FME), each requiring several block-matching operations to be performed. Particularly FME is even more concerning, as it requires an interpolation of the fractional pixels prior to its block-matching. To interpolate these samples, the HEVC standard uses 3 different FIR filters with 8-taps to generate 1/2 and 1/4 pixels. VVC increases this complexity, as it introduces a precision of 1/16 pixels for the motion vectors in Affine mode [7]. Therefore, VVC fractional interpolation filter is at least $17\times$ more complex than HEVC fractional interpolation filter.

The high computational complexity of the VVC standard also brings restrictions regarding power consumption on mobile devices. In order to deal with these restrictions, a common and efficient solution is to implement hardware accelerators, since these dedicated hardware architectures are more efficient in terms of power/energy. Recent solutions also rely on approximate computing to further reduce power of in-

Digital Object Identifier 10.29292/jics.v16i2.327

terpolation filters in recent video encoders (as we detail in Section II). The main limitations of previous works on filter hardware architectures for VVC standard is that they do not achieve the required throughput to process FME for UHD videos, and the coding efficiency analysis due to approximation are based on a few videos, which do not represent a real scenario, as we discuss in Section II.

This paper presents the design of an approximate fractional interpolation hardware to reduce the computational complexity and power dissipation of the FME operation on VVC encoders. Our design is based on approximating the number of taps of VVC filters to 6-taps, and by applying Multiple Constant Multiplication (MCM) algorithm to optimize filter datapaths. The compression efficiency loss due to approximation is analyzed with the Bjontegaard Delta Rate (BD-Rate) and PSNR (BD-PSNR) metrics compared to the precise solution. Our architecture includes 4 filter cores in parallel, making it able calculate 60 fractional pixels per clock cycle. Hence, the achieved throughput of the architecture makes it able to process up to 2560×1600 pixels resolution at 30 fps. The throughput achieved by our architecture was not yet achieved by competing solutions that target VVC standard, showing that our contribution bridges the gap between state-of-the-art encoding techniques and real-time applications.

This article is organized as follows. In Section II, after an overview of VVC standard, integer and fractional motion estimation algorithm is explained and related works about interpolation filter architectures are reviewed. In Section III, the proposed approximate VVC fractional interpolation hardware architecture is discussed. Compression efficiency and synthesis results are presented in Section IV. Finally, Section V presents the conclusions.

II. Background and Related Work

A. Versatile Video Coding Overview

Video coding standards specify video decoder and the format of coded video data (called bitstream). The structure of VVC is based on a hybrid video coding, which combines prediction and transforms to reduce redundancy of the input video signal, followed by quantization of prediction residual. A compatible VVC video encoder is composed by the following steps: Prediction, Transforms, Quantization, Inverse Transforms and Rescaling, In-loop Filtering, and Entropy Coding. All those steps are executed in a block-based manner, so the input video frames are first partitioned into blocks of samples so-called coding tree units, a set of coding tree blocks containing luminance and usually subsampled chrominance samples. Fig. 1 shows a simplified diagram of the VVC video encoder.

Prediction is divided into two parts: (1) Intra-frame prediction, which explores spatial redundancy and generates a predicted block based on neighboring samples from the same frame, and (2) Inter-frame prediction, which explores temporal redundancy by generating predicted samples from samples in previously encoded frames. Inter-frame prediction is composed by ME and Motion Compensation (MC). ME searches in reference frames for blocks similar to the original block to be encoded and generates a Motion Vector (MV) indicating the displacement of the best match (the most similar

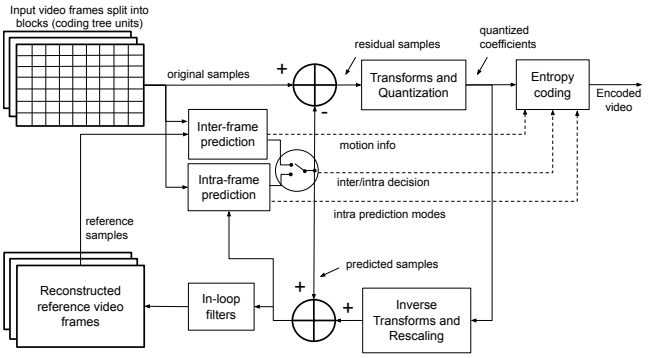

Fig. 1 Video encoder diagram. Source: modified from [8].

block), and MC reconstructs the block based on the obtained MV.

The difference between the original samples and predicted samples are called the residual samples, which are then processed by the transforms and quantization steps. Transforms convert the residual samples into the frequency domain, and Quantization decreases the amount of data of frequency coefficients representation, eliminating information which will likely not be perceived by the human visual system. Quantization is controlled by a Quantization Parameter (QP), which is directly proportional to the strength of the coding loss, and controls the quality of the reconstructed video.

The in-loop filters are responsible for removing coding artifacts that occur in previous steps due to the block-based encoding. Finally, the Entropy Coding step compresses the residual data using binary arithmetic encoder, generating the final coded bitstream of video.

Such structure is very similar to the two previous advanced video coding standards developed by ITU-T VCEG and ISO/IEC MPEG (HEVC and H.264/AVC). Most innovations brought by VVC rely on the new quadtree with nested multitype tree coding block partitioning structure, which supports binary and ternary splits to allow non-square coding units, and the tools included in each video coding step. Reviewing the details of each innovation of VVC is out of the scope of this work and the reader can refer to [4] for more information. We gave an in-depth detail on integer and fractional motion estimation in VVC in the next subsection, which is closely related to the main topic of our work, which are the VVC interpolation filters.

B. Integer and Fractional Motion Estimation in VVC

In modern video encoders, the inter-frame prediction is responsible for reducing the temporal redundancies by analyzing the pixels of the frame to be encoded and of previously encoded ones (called reference frames). The image to be encoded is divided into blocks and, for each block of the current image, the encoder searches for similar blocks in the reference image to find the block that best resembles the block of the current image (which will be encoded). The best match block is called the predicted block. When the best match is found, a MV is generated to indicate the offset of the position of the current block and the position of the selected block in the reference image. Inter-frame prediction is composed by ME and MC steps. In the reconstruction step, the information processed by the ME is used in MC to

generate a displayable frame once again. MC combines the residue blocks with the ones pointed by the MVs (from ME) to build a reconstructed block.

Current video coding standards support fractional-precision motion vectors as well. The purpose of this step is to allow sub-pixel motion representation, which is common when higher frame rates are used. To implement this, the ME process is composed of two steps: IME and FME. In the video encoder, to generate fractional precision motion vectors, it is necessary to define FME, which is composed of two steps: (i) interpolation filter to generate fractional pixels, since the only pixels available in the image are those of full precision; and (ii) search stage to find the fractional block with better resemblance to the original block, to find the direction in which the sub-pixel shift occurred. Like in the FME, MC must realize an interpolation process to obtain fractional samples from integer ones when a fractional MV is used. Hence, interpolation filter is an important component of codec design, because it is used in both the encoder and decoder. Another observation is that MC architectures have smaller throughput requirements than ME ones, because the ME operation is performed over several candidate blocks for each input block, whereas MC only interpolates a single block (the best one found in ME) for each input block. Fig. 2 shows an example of MVs with integer and fractional accuracy, thus requiring IME and FME.

HEVC supports 1/2 and 1/4 pixel MV accuracy, which means the FME generates three fractional pixels between two integer pixels in horizontal and vertical directions, and also nine pixels in the diagonal region, generating 15 interpolated pixels for each integer pixel. This interpolation is computed using 7-tap and 8-tap Finite Impulse Response (FIR) filters. In VVC, the accuracy of a MV is 1/16 of a pixel, so 15 pixels are interpolated horizontally and vertically, plus 225 pixels for the diagonal region, resulting in 255 interpolated pixels for each integer pixel. This represents 17× more interpolated pixels in VVC compared with HEVC. Hence, it is expected at least a 17× increase in computational complexity of VVC interpolation filter compared to the one defined in HEVC.

Fig. 3 illustrates the fractional pixels created between two

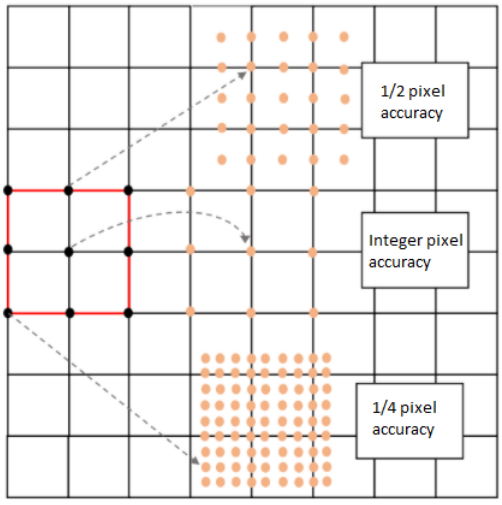

Fig. 2 Integer and Fractional Motion Estimation. Source: [9]

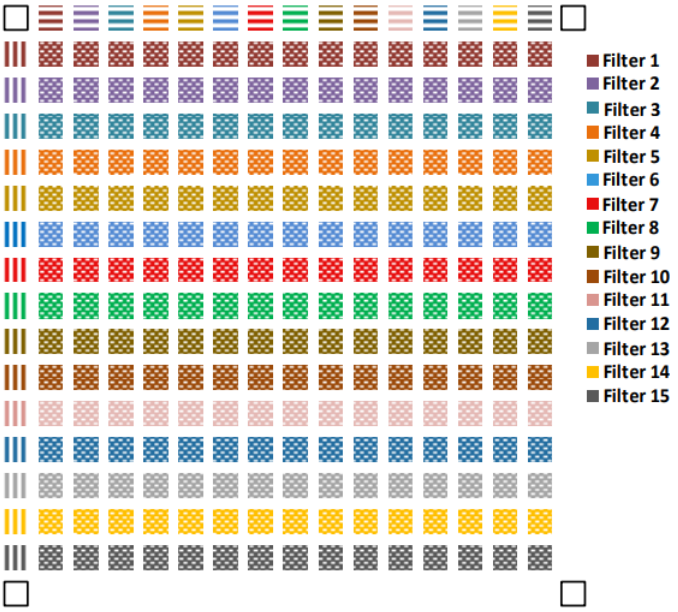

Fig. 3 Interpolated Pixels in VVC. Source: [10]

integer pixels of an image in VVC standard. Each filter is named based on its position in a left-to-right and top-to-bottom convention.

Like HEVC, VVC uses 8-tap FIR filters to generate fractional pixels. Each of the 15 filters has its own coefficient set. The filter coefficients are shown in Table I. An example of how filter 6 (F6) is calculated from the 8 integer input samples A_{-3}, A_{-2}, A_{-1}, A_0, A_1, A_2, A_3, and A_4 is shown in (1).

$$F_6 = (-1A_{-3} + 3A_{-2} - 9A_{-1} + 47A_0 + 31A_1 \\ -10A_2 + 4A_3 - 1A_4) >> 6 \quad (1)$$

C. *Related Work*

Several works have proposed hardware architectures for the HEVC interpolation filters. Diniz et al. [11] propose an architecture containing two modules to address luma and chroma interpolation filters, each one with 12 pixel parallel configurable interpolation datapaths. The configurable datapath is optimized to reduce area of the interpolation dat-

Table I. Coefficients of the interpolation filters defined in VVC standard.

Filters	Coefficients							
	A_{-3}	A_{-2}	A_{-1}	A_0	A_1	A_2	A_3	A_4
1	0	1	-3	63	4	-2	1	0
2	-1	2	-5	62	8	-3	1	0
3	-1	3	-8	60	13	-4	1	0
4	-1	4	-10	58	17	-5	1	0
5	-1	4	-11	52	26	-8	3	-1
6	-1	3	-9	47	31	-10	4	-1
7	-1	4	-11	45	34	-10	4	-1
8	-1	4	-11	40	40	-11	4	-1
9	-1	4	-10	34	45	-11	4	-1
10	-1	4	-10	31	47	-9	3	-1
11	-1	3	-8	26	52	-11	4	-1
12	0	1	-5	17	58	-10	4	-1
13	0	1	-4	13	60	-8	3	-1
14	0	1	-3	8	62	-5	2	-1
15	0	1	-2	4	63	-3	1	0

apath, by relying on the symmetries of the HEVC interpolation filters. This work was extended to reduce power of HEVC interpolation by employing dynamic reconfigurability in field-programmable gate arrays [12], and adder compressors [13]. The works [11–13] provide precise solutions that do not affect BD-Rate. Afonso et al. [14] propose a hardware architecture for FME in HEVC, thus including a interpolation unit and comparison units for the search phase. The main contribution to reduce FME complexity is to adopt only squared-shaped prediction units instead of supporting all the 24 possible prediction unit shapes. Such choice is based on a software analysis on HEVC reference software which reveal that limiting prediction unit to square-shaped sizes reduces almost 59% the encoding time (on average of various videos) at the cost of around 4% BD-Rate increase.

Approximate architectures were also proposed to further extend the area and power savings of dedicated HEVC interpolation filter hardware. Penny et al. [15] propose a configurable hardware that supports the 8-tap HEVC interpolation filters and 6-tap approximate interpolation filters by removing the leftmost and rightmost taps of the original filter. The approach increases BD-Rate in 0.527% compared to the original HEVC interpolation filters. Kalali et al. [16] propose approximate 4-tap and 3-tap interpolation filters for HEVC encoding. The work employs HCub multiplierless multiple constant multiplication algorithm [17] to reduce the number and size of adders of the proposed filter hardware. The approach results in up to 1.14% BD-Rate increase compared to the original HEVC interpolation filters. Silva et al. [18] propose an architectural template for approximate HEVC interpolation filter supporting 6-tap, 4-tap and 2-tap filters, increasing BD-Rate in 0.02%, 0.25% and 0.89%, respectively, compared to the original HEVC interpolation filters. The approximate filters were designed by removing the leftmost and rightmost filter coefficients, and the removed coefficients were added to their closest remaining neighbors to keep the sum of coefficients in 64.

All those works [11–16, 18] target HEVC interpolation filtering, which is 17× less complex than VVC interpolation filtering, as we motivated in the introduction. Performance, power, and energy of HEVC and VVC interpolation filters cannot be directly compared. Although the approximate computing approaches proposed in the context of HEVC are applicable to VVC, the coding efficiency results are also very different from those obtained with HEVC, because they are sensible to video content, video format and encoding decisions.

Since the VVC standard was finished in July, 2020, a few works proposing interpolation filter hardware architectures for VVC can be found in the literature. Azgin et al. [19] propose a reconfigurable hardware architecture for VVC interpolation filters targeting MC, that needs to interpolate only one fractional pixel for each integer pixel. Mert et al. [10] propose a hardware architecture focusing on FME. This design implements 15 fractional interpolation filter datapaths in parallel. It uses the Hcub MCM algorithm [17] to implement multiplications by constant using shifters and adders. Mahdavi and Hamzaoglu [20] propose a VVC interpolation filter hardware with a memory-based multiple constant multiplication approach. The results are shown only for Field Programmable Gate Array (FPGA) platform, which is not suitable for low power systems. The works in [10, 19, 20] provide precise results that do not affect BD-Rate. The aforementioned works have a limited throughput when considering FME, that are supported by [10, 20], but allow real-time encoding for up to 1920×1080 video resolution.

The only work that introduces an approximate VVC fractional interpolation hardware employs 4-tap filters instead of the original 8-tap ones, which leads to a power reduction of up to 40% [21]. However the work targets only FPGA platform, which is less suitable for low power systems that will benefit of such approximate computing techniques. Moreover, it only evaluates the compression efficiency loss of the approximation for two videos (Kimono and Tennis) which are not in the Common Test Conditions (CTC) of VVC standard [22]. Although they show a low impact in compression efficiency for these two videos, in up to 0.52% increase of BD-Rate compared to the default interpolation filters included in the standard, the analysis was conducted for only 10 frames of those two videos in Low Delay P configuration. It is not possible to conclude if this result will keep for other video sequences, resolutions and configurations. Our approach rely on a more comprehensive analysis with 14 video sequences using the more generic Random Access Configuration (which includes bi-prediction frames) and using 32 frames of each video sequence. In addition, the architecture has also a limited throughput for FME, supporting 1920×1080 video resolution at 47 fps.

Our work addresses the limitations of previous works by providing an architecture with higher throughput for the VVC interpolation filters thus supporting higher resolution video encoding in real-time, and an analysis of video coding efficiency drops (using BD-Rate metric) with 14 videos from the CTC document of VVC standard [22]. This is a more realistic analysis on the video coding efficiency drops due to approximation in the context of VVC compared to the one presented in [21] that evaluates only two videos that are not included in VVC's CTC. In addition, it show results for Application Specific Integrated Circuit (ASIC), using a standard-cells implementation in 65 nm CMOS technology, which is a more suitable fabrication technology for low power systems than FPGA. A summary of characteristics of related works is shown in Table II, comparing those works with our work with respect to if the work adopt approximation techniques to the filters, what is the target video coding standard, which technology fabric (FPGA, ASIC, or both), what is the design target, and the number of videos used in the BD-Rate analysis if some approximation is performed.

III. Proposed Architecture

Our proposed architecture is based on the design of a new set of 6-tap interpolation filters. The proposed filters with only 6-tap replace the original 8-tap filters defined in the VVC standard, but only for the FME stage of video encoder. It is important to notice that video encoder is not standardized by VVC, but only the video decoder and the input video decoder syntax. Hence, the design of video encoder is

Table II. Summary of related works

	Approx.	Standard	Techn.	Design	Videos
[11]	No	HEVC	ASIC	FME + MC	N.A.
[12]	No	HEVC	FPGA	FME + MC	N.A.
[13]	No	HEVC	ASIC	FME + MC	N.A.
[14]	No	HEVC	Both	FME + MC	24
[15]	Yes	HEVC	Both	FME	24
[16]	Yes	HEVC	Both	FME	14
[18]	Yes	HEVC	FPGA	FME + MC	15
[19]	No	VVC	Both	MC	N.A.
[10]	No	VVC	Both	FME + MC	N.A.
[20]	No	VVC	FPGA	FME	N.A.
[21]	No	VVC	FPGA	FME	2
Our	Yes	VVC	ASIC	FME	14

free for optimizations provided that it generates a compatible bitstream for the standardized VVC decoder. This way, our work approximates only the interpolation filters in FME stage, and the interpolation filters used in the MC stage are kept as defined in the VVC standard, to make this stage in the encoder fully compatible with MC used in the video decoder, thus making our approach fully compatible with the VVC standard. Our approach benefits on the approximation in only the FME module because it needs to generate at most 255 fractional pixels for each integer pixel, since it chooses what is the best fractional offset to be encoded, while MC needs to generate only one fractional pixel for each integer pixel.

Another principle to design such approximate filters is to reduce the number of taps to 6-tap but keeping the sum of the coefficients of each filter in 64, in order to keep the filter response of the approximate version similar to the precise version, and to keep the shift operation by 6 at the end of the calculation, as shown in (1). Our goal is not to design a completely different (and more simple) interpolation filter to replace the original one, but to make a small approximation on the interpolation filter in order to not degrade the coding efficiency that is provided by VVC. To maintain the sum of filter coefficients in 64 and reduce the number of taps from 8 to 6, the leftmost and rightmost coefficients of original 7-tap and 8-tap filters 2 to 14 were added to the leftmost and rightmost coefficients of the new 6-tap filters, as follows: on the left side, each coefficient multiplied by the input sample A_{-3} is added with each coefficient multiplied by the input sample A_{-2}. On the right side, each coefficient multiplied by the input sample A_4 was added with each coefficient multiplied by the input sample A_3. In this way, it was possible to reduce the number of taps, without changing the total sum of the coefficients in each filter. Table III shows the proposed approximate filters for use in FME stage of VVC video encoders.

The architecture proposed in this work is presented in Fig. 4. The proposed architecture was developed to receive 9 integer samples as input, which are needed to generate a line of a 4×4 block. The input samples are delivered to four filter cores in parallel. Each filter core receives 6 integer samples as input and generates 15 interpolated pixels in parallel at the output according to the filters shown in Table III. Hence, the whole architecture outputs 60 fractional pixels in each clock cycle. This parallelism was employed to reach the high

Table III. Coefficients of the Proposed Approximate Interpolation Filters

Filters	Coefficients							
	A_{-3}	A_{-2}	A_{-1}	A_0	A_1	A_2	A_3	A_4
1	0	1	-3	63	4	-2	1	0
2	0	1	-5	62	8	-3	1	0
3	0	2	-8	60	13	-4	1	0
4	0	3	-10	58	17	-5	1	0
5	0	3	-11	52	26	-8	2	0
6	0	2	-9	47	31	-10	3	0
7	0	3	-11	45	34	-10	3	0
8	0	3	-11	40	40	-11	3	0
9	0	3	-10	34	45	-11	3	0
10	0	3	-10	31	47	-9	2	0
11	0	2	-8	26	52	-11	3	0
12	0	1	-5	17	58	-10	3	0
13	0	1	-4	13	60	-8	2	0
14	0	1	-3	8	62	-5	1	0
15	0	1	-2	4	63	-3	1	0

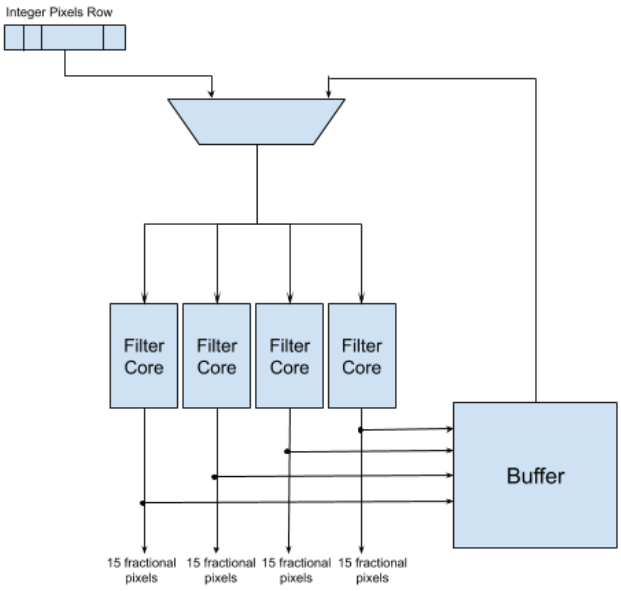

Fig. 4 Approximate Hardware Architecture for VVC Interpolation Filter

throughput needed by the VVC interpolation filter to process high resolutions videos in real-time.

A buffer stores the fractional pixels that are used as input to calculate other fractional pixels. The input multiplexer selects the input pixels or the pixels from buffer depending on which set of pixels must be interpolated. To support two passes of the 8-bit depth reference integer input pixels, the filter datapaths were designed with 10 bits. The output fractional pixels bit depth is based on the number of fractional samples required for a 4×4 block size, which in the proposed solution, has outputs of 12 bits.

Fig. 5 details the internal architecture of the Filter Core, which was instantiated 4 times in the top level architecture to generate 60 pixels in parallel. This module implements all the 15 filters defined in Table III using the MCM technique, that replaces the multiplication of input values of multiple constant coefficients by add and shift operations, optimizing the number of adders and critical path. Since the same input pixel is multiplied by multiple constants when looking at all the 15 filters, the datapath can be optimized by applying an

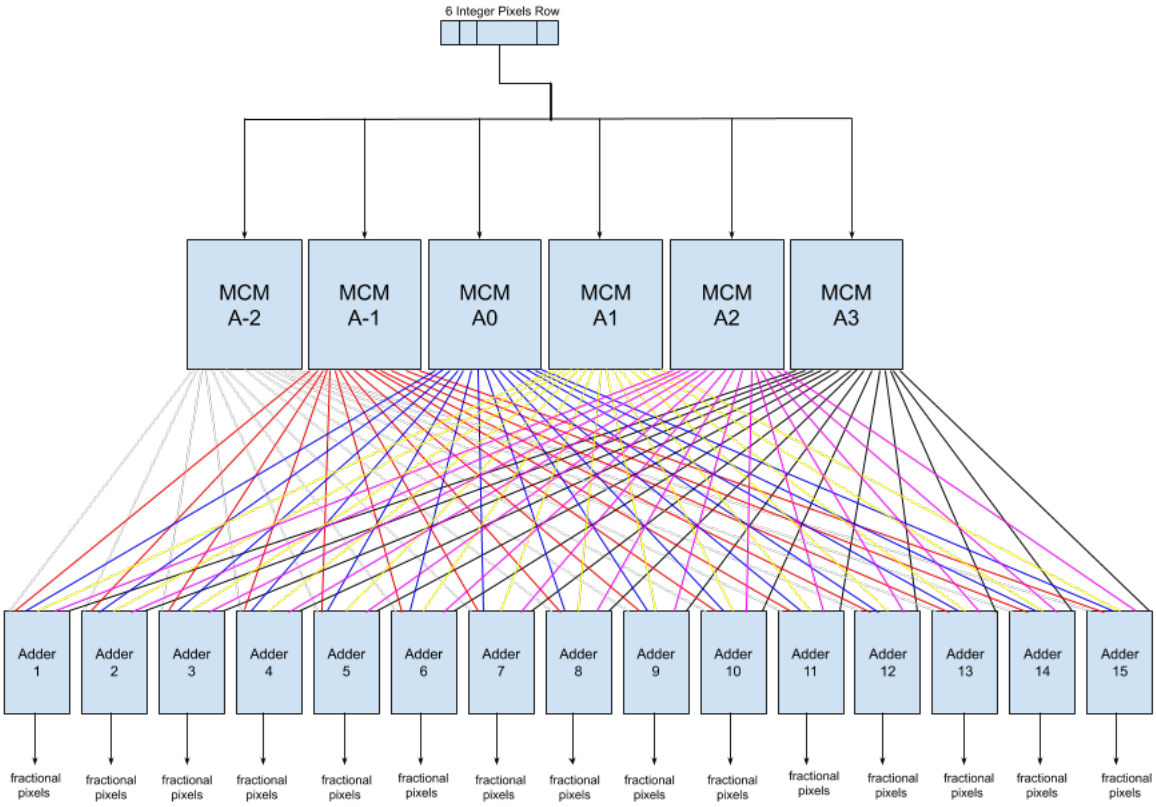

Fig. 5 Filter Core Architecture

efficient MCM algorithm. The MCM algorithm used in this work is Hcub, first proposed in [17]. To generate 15 filters in parallel, our architecture includes only 6 MCM modules in parallel, because this is the number of integer pixels delivered in the input. MCMs were generated with the Spiral software [23]. Spiral has a website interface that generates a C and Verilog files of a optimized MCM module based on the user-defined inputs: constants (the filter coefficients), the number of fractional bits (which is zero for our case, because the coefficients are integer), the MCM algorithm (we have employed HCub [17]), the depth limit (which specifies the maximum allowed adder tree depth, and we kept as unlimited), and a secondary opimization phase. The 6 MCM module outputs are delivered to 15 adder modules that sum the corresponding outputs indicated by colored arrows to generate the final fractional interpolated pixels. All this process is conducted in one clock cycle. Hence, the throughput of our architecture is 60 pixels per cycle.

Table IV shows a comparison in the number of arithmetic operators when employing our approximate filters and MCM, compared with precise version defined in VVC standard. First, by using MCM there is a complete elimination of the 160 multipliers needed to implement precise VVC filters, with an increase of only 8% in the number of adders. The replacing of multipliers by adders and shifters makes a substantial reduction of circuit area, given the area of the multiplier is much higher compared to the area of adders and subtractors. The approximate technique further reduces the number of arithmetic operators compared to precise filters.

Table IV. Comparison of Precise and Approximate Filters

	Adders	Shifters	Multipliers
Precise	95	—	160
Precise (MCM)	123	96	0
Approximate	75	—	120
Approximate (MCM)	103	96	0

IV. Results and Discussion

To measure the compression efficiency, the BD-Rate and Bjontegaard-Delta PSNR (BD-PSNR) metrics are used. BD-Rate represents an average difference in the bit rate between a reference encoder and a test encoder for equivalent image quality. When the test encoder is more efficient than the reference, the resulting BD-Rate is negative [24]. Similarly, BD-PSNR represents the difference in quality (calculated with PSNR, in dB) when the equivalent bit rate is considered. In our work, we consider as reference encoder the default configuration of VTM (VVC Test Model) version 10.1rc1 [25]. The test configuration was made by modifying the VTM software to implement our approximate interpolation filters in software only for the FME. We have used Random Access (RA) configuration with the first 32 frames of 14 video sequences of Class B (1920x1080 pixels), Class C (832x480 pixels) and Class D (416x240 pixels). Thirteen of these video sequences are recommended by the CTC of VVC [22] while Kimono video sequence is included in the CTC of HEVC but is not included in CTC of VVC. Each video sequence was encoded with four Quantization Parameters (22, 27, 32, 37) to enable BD-Rate and BD-PSNR calculation. The results are shown in Table V.

The most part of BD-Rate values of Table V are pos-

Table V.: Results for BD-Rate and BD-PSNR (using the CTC configuration)

Sequence Name	Class	BD-Rate (%)	BD-PSNR (dB)
Kimono	Class B	0.12	-0.0067
MarketPlace	Class B	-0.01	0.0001
Cactus	Class B	0.19	-0.0037
BasketballDrive	Class B	0.19	-0.0031
BQTerrace	Class B	0.46	-0.0084
RitualDance	Class B	0.28	-0.0134
BasketballDrill	Class C	-0.02	0.0008
BQMall	Class C	0.51	-0.0196
PartyScene	Class C	0.35	-0.0162
RaceHorses	Class C	0.81	-0.0505
BasketballPass	Class D	0.54	-0.0263
BQSquare	Class D	0.58	-0.0268
BlowingBubbles	Class D	0.69	-0.0265
RaceHorses	Class D	1.07	-0.0505
Average		0.41	-0.0165

itive and small, showing a negative impact in compression efficiency of the approximate interpolation filter compared to the precise (default) VVC interpolation filters. This is expected as we approximate filters by reducing filter taps. However, the impact in compression efficiency can be considered small, since most of BD-Rate values are less than 1%, while only RaceHorses (Class D) video sequence presents a BD-Rate value slightly higher than 1%. Our average value is 0.41% which is very negligible compression efficiency drop.

Table V also shows negative BD-Rate values. It means that the compression efficiency was improved compared with the default version. It also happens because of the complicated algorithms involved in the rate distortion optimization of advanced video encoders, that makes the encoder to find a better solution when they skip a local minimum rate-distortion point. It happens in our case for BasketballDrill (Class C) and MarketPlace (Class B) video sequences. In the video sequences that the BD-PSNR is positive, it also means that for the same bit rate the proposed approximate filters have obtained a better quality than precise filters.

The proposed architecture was described in VHDL and synthesized with the Cadence Genus Synthesis Solution tool using the ST Microelectronics 65 nm standard cells at 1.35 V. The synthesis results were obtained for each supported resolution which translates to a target frequency, as presented in Table VI. The gate count (in kgates) is the number of equivalent 2-input NAND gates. The needed throughput of each video resolution and frame rate is calculated as the number of interpolated samples to be calculated for second considering that in VVC for each integer sample we have to interpolate 255 fractional samples in the worst case. Since our architecture calculates 60 fractional pixels per cycle it is possible to determine target operating frequencies and set it as a constraint in the Genus tool. Our architecture supports video encoding in real-time of up to 2560×1600 pixels video at 30 fps dissipating 23.98 mW of total power, when operating at 522 MHz. On the other side, it can process low resolution videos (416×240 pixels) by synthesizing the architecture to 12 MHz target frequency and dissipating only 5.86 mW of total power.

In Table VII, we can see that our solution consumes more power and area resources compared to the implementation of the approximate hardware in HEVC [15] and the precise hardware in VVC [10]. We also computed the energy/pixel consumption, which represents how much energy is spent in a second to process each input pixel. Such increase in power and area is justified because the proposed architecture was designed aiming at a higher performance to support the processing requirements of the FME accelerators for UHD videos, while related works [10, 15] support real-time encoding of HD videos. With the increase in frequency, our architecture is able to achieve a higher performance compared to other solutions. This higher throughput also increases energy consumption, as our architecture presents a 25% overhead to process each pixel when compared to [15]. However, we can note that the solution in [15] targets HEVC encoder which is nearly $17\times$ less complex than VVC encoder. Our solution also provides a more comprehensive analysis on compression efficiency drop of interpolation filter approximation in VVC compared with [21], as it provides the analysis only for two videos. It was not possible to compare synthesis results with [21] because they do not provide ASIC implementation results.

V. Conclusion

This article presents a dedicated hardware architecture for an approximate interpolation filter based on VVC standard. The architecture supports 15 filters of 6-taps, implemented using the Hcub MCM algorithm. With these techniques the architecture is able to process up to 2560×1600 pixels videos at 30 fps with power dissipation of 23.9 mW when operating at a frequency of 522 MHz, with an average compression efficiency degradation of only 0.41% compared to default VVC video encoder software configuration.

Acknowledgements

The authors would like to thank CNPq, CAPES and FAPERGS for the financial support to this work.

References

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," Tech. Rep., Mar. 2020. [Online]. Available: www.cisco.com

[2] ——, "Cisco visual networking index: Forecast and trends, 2017–2022," Tech. Rep., Nov. 2018. [Online]. Available: www.cisco.com

[3] ITU-T and ISO/IEC, "Versatile Video Coding," *ITU-T Recommendation H.266 and ISO/IEC 23090-3"*, 2020.

[4] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *Proceedings of the IEEE*, pp. 1–31, 2021.

[5] ITU-T and ISO/IEC, "High Efficiency Video Coding," *ITU-T Recommendation H.265 and ISO/IEC 23008-2*, 2013.

[6] Í. Siqueira, G. Correa, and M. Grellert, "Rate-distortion and complexity comparison of hevc and vvc video encoders," in *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*. IEEE, 2020, pp. 1–4.

[7] D. Li, Z. Zhang, K. Qiu, Y. Pan, Y. Li, H. R. Hu, and L. Yu, "Affine deformation model based intra block copy for intra frame coding," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

Table VI. Synthesis Results

Supported resolution and frame rate	Area (μm^2)	Gate Count (k)	Total Power (mW)	Frequency (MHz)
2560×1600 @ 30 fps	142.838	68.67	23.98	522.47
1920×1080 @ 50 fps	115.207	55.39	19.87	440.72
1920×1080 @ 30 fps	62.053	29.83	8.42	264.4
832×480 @ 50 fps	60.025	28.86	6.81	84.87
832×480 @ 30 fps	60.025	28.86	6.35	50.92
416×240 @ 50 fps	60.025	28.86	5.92	21.22
416×240 @ 30 fps	60.025	28.86	5.86	12.73

Table VII. Comparison With Related Works

Related Work	[15]	[10]	Our
Standard	HEVC	VVC	VVC
Technology (nm)	90	90	65
Frequency (MHz)	300	435	522
Gate Count (k)	12.8	37.6	68.7
Total Power (mW)	15.8	–	23.9
Supported Resolution	1920x1080 @ 49 fps	1920x1080 @ 88 fps	2560x1600 @ 30 fps
Energy/pixel	0.155 nJ	–	0.194 nJ

[8] C. M. Diniz, B. Abreu, M. Grellert, F. M. Sampaio, D. Palomino, F. L. L. Ramos, B. Zatt, and S. Bampi, "Joint algorithm-architecture design of video coding modules," *VLSI Architectures for Future Video Coding*, p. 41, 2019.

[9] B. Bing, *Next-generation video coding and streaming*. John Wiley & Sons, 2015.

[10] A. CanMert, E. Kalali, and I. Hamzaoglu, "A low power versatile video coding (VVC) fractional interpolation hardware," in *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)*. IEEE, 2018, pp. 43–47.

[11] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel, "High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for hevc," in *2013 IEEE International Conference on Image Processing*, 2013, pp. 2091–2095.

[12] ——, "A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 238–251, 2015.

[13] C. Diniz, M. Fonseca, E. da Costa, and S. Bampi, "Evaluating the use of adder compressors for power-efficient hevc interpolation filter architecture," *Analog Integrated Circuits and Signal Processing*, vol. 89, 2016.

[14] V. Afonso, H. Maich, L. Audibert, B. Zatt, M. Porto, L. Agostini, and A. Susin, "Hardware implementation for the hevc fractional motion estimation targeting real-time and low-energy," *Journal of Integrated Circuits and Systems*, vol. 11, no. 2, pp. 106–120, 2016.

[15] W. Penny, M. Ucker, I. Machado, L. Agostini, D. Palomino, M. Porto, and B. Zatt, "Power-efficient and memory-aware approximate hardware design for HEVC FME interpolator," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 237–240.

[16] E. Kalali and I. Hamzaoglu, "Approximate hevc fractional interpolation filters and their hardware implementations," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 3, pp. 285–291, 2018.

[17] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 2, pp. 11–es, 2007.

[18] R. da Silva, Í. Siqueira, and M. Grellert, "Approximate interpolation filters for the fractional motion estimation in HEVC encoders and their VLSI design," in *Proceedings of the 32nd Symposium on Integrated Circuits and Systems Design*, 2019, pp. 1–6.

[19] H. Azgin, A. C. Mert, E. Kalali, and I. Hamzaoglu, "A reconfigurable fractional interpolation hardware for VVC motion compensation," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 99–103.

[20] H. Mahdavi and I. Hamzaoglu, "A VVC fractional interpolation hardware using memory based constant multiplication," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 2021, pp. 1–5.

[21] H. Azgin, E. Kalali, and I. Hamzaoglu, "An approximate versatile video coding fractional interpolation hardware," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2020, pp. 1–4.

[22] F. Bossen, "JVET common test conditions and software reference configurations for sdr video," in *Document JVET-N1010 14h JVET Meeting, Geneva, CH*, 2019.

[23] J. M. Moura, J. Johnson, R. Johnson, D. Padua, V. Prasanna, M. Püschel, and M. Veloso. (2020) Spiral multiplier block generator. http://spiral.ece.cmu.edu/mcm/gen.html.

[24] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.

[25] VTM. (2020) VVC test model (VTM) v. 10.1rc1. https://jvet.hhi.fraunhofer.de/.