Memory-Aware, Low-Power and High-Throughput AV1 FME Interpolation Architecture

William Kolodziejski, Robson Domanski, Marcelo Porto, Bruno Zatt, Luciano Agostini
Federal University of Pelotas (UFPel) - Pelotas, Brasil
Video Technology Research Group (ViTech), Group of Architectures and Integrated Circuits (GACI)
{wkolodziejski, radomanski, porto, zatt, agostini}@inf.ufpel.edu.br

Abstract—**The AV1 implement several complex tools, such as the Fractional Motion Estimation (FME), which defines 90 different interpolation filters. This paper presents an optimized approximate architecture for the AV1 FME interpolation filters, reaching real-time interpolation for UHD 8K@30fps in a memory-aware, low-area, and low-power design. The architecture was synthesized for a 40nm TSMC standard-cells technology allowing a memory bandwidth reduction of 59.5% in comparison with state-of-the-art solutions. It reaches power gains of 83.4% when compared to a precise architecture and 7.1% when compared to a previous work of our group. The area reduction is of 84.6% and 41.8% when compared to the precise version and to our previous work, respectively. The approximation leads to a small coding efficiency degradation of 1.58% in BD-BR.**

Index Terms—**approximate computing, interpolation, fractional motion estimation, video coding, AV1**

I. Introduction

THE advent of the COVID-19 pandemic has pushed up even more the use of digital videos, which was already in great increase over the last years. One clue of this assumption is the decision of Netflix, Amazon Prime, and YouTube to reduce the video quality to guarantee the quality of service [1]. Also, [2] estimated that by 2022, digital video content would be responsible for the traffic of about 77.49 EiB/month (1 EiB is 2^{60} bytes), meaning approximately 82% [3] of the global Internet traffic, which may be underestimated since no pandemic was in sight in that moment. Under this scenario, video compression is being used like it never was before, and this implies in the necessity to improve the current video encoders, in order to reduce the power dissipation and time consumed during the video compression process. One of these video encoders is the AV1, released in 2018 by AOM.

One of the most used AV1 tools is the Fractional Motion Estimation (FME), counting with 90 Finite Impulse Response (FIR) filters [4], that requires samples fetched from the memory. Since the calculation is repeated multiple times during the whole coding process a large memory bandwidth is required.

This paper presents a dedicated memory-aware, low-power and high-throughput hardware design for the AV1 FME interpolation able to process up to UHD 8K@30fps, supporting all filters and block sizes. The method used to develop this architecture consists on the proposal of an optimized combinational multiplierless multi-filter solution. The solution approximates the filters coefficients to hardware-friendly values and reduces the number of taps whereas sustaining the compliance with the AV1 FME specification.

II. Background and Related Works

The inter-frame prediction step of the AV1 is composed of the Motion Estimation (ME) and Motion Compensation (MC) steps. The ME finds, in a list of previously encoded frames, the most similar block to the current one and then generates a Motion Vector (MV) indicating the frame and the position of this block. Then, the MC uses the generated MV to reconstruct the predicted block. This predicted block is required because the differences between the original block and the predicted one (*a.k.a. residues*) must be considered to preserve the video quality. The ME can be further divided into Integer Motion Estimation (IME) and Fractional Motion Estimation (FME). In many cases, the block found by the IME does not satisfactorily match the current block and this can be improved using the FME. This tool is responsible to refine the selected MV using fractional (sub-pixel) positions. The use of fractional positions allows for a better matching in the prediction process, reducing the residues and increasing the encoding efficiency.

To perform the interpolation, the AV1 FME uses a total of 90 FIR filters, organized in six families of 15 filters each (some are divided by number of taps): (i) **Regular** - Lagrange-based filter with 6-tap or 4-tap (15 filters in each); (ii) **Smooth** - Hamming Window-based filter with 6-tap or 4-tap (15 filters in each); (iii) **Sharp** - Direct Cosine Transform-based filter with 8-tap; (iv) **Bilinear** - 2-tap filter used in fast operations. Among the current video codecs, the AV1 uses the largest number of FIR filters in the FME interpolation process. The VVC, for example, uses 78 filters [5]. The AV1 FME filters have from 2 up to 8-taps, achieving an accuracy of 1/8 and 1/16 samples, for luminance and chrominance, respectively, allowing for half, quarter, eighth, and sixteenth precision [4].

The interpolation is divided in two steps: (i) horizontal and (ii) vertical filtering. This division exploits the potential statistical discrepancy between vertical and horizontal directions, improving prediction quality [4]. The 4-tap filters are used for 4×4 block sizes and the 8-tap for larger block sizes.

Due to the fact that the AV1 codec was recently released, limited set of works are found in the literature targeting the encoder, specially employing approximate computing and memory optimizations. A work, targeting the AV1 FME interpolation filters is presented in [6], but with no employment of approximate computing nor memory optimizations and without support for all the 90 defined filters. Two previous works of our group focused in the AV1 interpolation filters. The work presented in [7] is focused on the Motion Compensation decoder step where the filters are applied to decode the frames and then no approximation is allowed. The second work of our group [8] applies approximate computing at the FME, changing the filters coefficients to hardware-friendly ones, but without reduction in memory bandwidth.

III. Proposed Memory-Aware Approximations

In order to simplify the AV1 FME interpolation filter, the first hardware-friendly approximation explored in this paper, was to transform the required multiplications defined in the FIR filters into shifts or shift-adds operations, using an approximation of the original taps. Since there are 90 different filters with up to eight taps, this simplification has a high impact in terms of the required hardware resources.

The proposed approximation considered the fact that the two central taps are the most important ones to the interpolation result and defined their values as two shift-adds, improving the precision. On the other hand, since the most peripheral taps are less important, their values were defined using only one shift. This means that the two central taps use a combination of four powers of two to be generated whereas the other taps are generated with only one power of two. A similar solution was explored in our previous work [8].

Another approximation proposed in this paper is to reduce the number of taps to a maximum of four in all FME filters, mainly to reduce the memory bandwidth. Our previous work [8] only applied approximate computing at the filter coefficients values, mainly focusing in reduce power dissipation. The decision to reduce the taps to four was based in experiments comparing a solution with six taps, which showed limited impacts in coding efficiency. Therefore, further reducing the taps provided important gains in memory bandwidth reduction.

The solution presented in this paper brings some imprecision to the results, as will be discussed in the next section, but, on the other hand, it allowed an important memory bandwidth reduction, together with an important reduction in power dissipation consumption and used area.

Table I shows a comparison with some of the 90 AV1 FME interpolation filters, highlighting the differences between the precise and approximated tap values. Using the Sharp filter as example, the original taps $\{-4, 12, -24, 80, 80, -24, 12, -4\}$ were converted to $\{-8, 84, 84, -32\}$. Regular, Smooth, and Bilinear filters also suffered similar approximations. The coefficient values for each filter were manually set to the power of two closest to the original values, whereas respecting the hardware constraints imposed by the available shifters.

Other important observation from Table I is that the sum of all imprecise taps of each approximated filter must be as similar as possible to the sum of all taps of the precise filter. This sum is 128 and allows a final division using a simple shift operation whereas guaranteeing that the filters

TABLE I
Example of Filters Coefficients

Filter	TAP	Version	Coefficients
Sharp	8	Precise	{-4, 12, -24, 80, 80, -24, 12, -4}
	4	Approximated	{-8, 84, 84, -32}
Bilinear	2	Precise	{56, 72}
		Approximated	{44, 84}
Regular	6	Precise	{2, -14, 76, 76, -14, 2}
	4	Approximated	{-32, 84, 84, -8}
	4	Precise	{-12, 76, 76, -12}
		Approximated	{-32, 84, 84, -8}
Smooth	6	Precise	{-2, 14, 52, 52, 14, -2}
	4	Approximated	{32, 44, 44, 8}
	4	Precise	{12, 52, 52, 12}
		Approximated	{32, 44, 44, 8}

gains are equal to 1 even with the approximations. In a very few approximations (two cases), the sum of the taps is equal to 124, due to the constraints we used in the hardware design, but in all other the sums are exactly 128.

IV. Evaluation of Coding Efficiency Impacts

The proposed approximations were evaluated in comparison to the original filters using 9 video sequences from Mozilla and Netflix data set [9]. The evaluation considered four $CQs = \{20, 32, 43, 55\}$ enabling all AV1 encoding tools. CQ stands for *Constant Quality* and ranges from 0 (no quality loss) to 63 (maximum quality loss). The experiments were done using the libaom, the AV1 reference software, version 2.0.0 [10].

The coding efficiency evaluation was done using the Bjøntegaard Delta Bit Rate (BD-BR) [11] metric. This metric indicates, for the same objective video quality, the percentage difference of bit-rate required to represent a video. The closest to zero, the more similar to the original video. Negative values mean that less bits were required to represent the same result. The experiments were done following the Video Codec Testing and Quality Measurement [9].

Table II presents the results of coding efficiency of the proposed approximations in comparison to the original filters, grouping the videos by resolution. The first observation is that the higher the video resolution, the lower tends to be the approximation impact on BD-BR. In some cases (negative values) the approximations can even increase the coding efficiency. This was not an expected result, but may be justified by the fact that every encoder decision affects the other tools and heuristics, which can lead to hard-to-predict results.

The proposed approximations caused a small degradation in coding efficiency and the results are as better as higher is the video resolution. The worst results were reached for the Full HD sequences and the average results for UHD 4K videos are close to zero. Taking in account that Full HD is one of the most common resolutions, the 2.74% degradation in BD-BR is expressive, but still acceptable, especially for scenarios where power and area are more important than quality, as the case of mobile devices. When compared to our previous work [8], which reached an average BD-BR of 0.54%, the novel

TABLE II
CODING EFFICIENCY RESULTS

Sequence	Resolution	Frames	BD-BR (%)
Arena of Valor	Full HD	60	0.67
Market Place			2.46
Square and Time-lapse			2.62
Tunnel Flag			5.21
Foreman	UHD 4K	60	0.82
Coastguard			-0.14
Cactus			1.28
Bar Scene			-1.90
Boxing			2.06
Average	Full HD	-	**2.74**
	UHD 4K		**0.42**
	Overall		**1.58**

overall BD-BR results are slightly higher, but with important improvements in memory bandwidth.

V. Memory-Aware Approximate Architecture

Figure 1 presents the proposed architecture, called Basic Multiplierless Multi-filter (BMM), at filter level. This set of shifters, adders and multiplexers is able to compute all the 90 AV1 approximated FME interpolation filters (one at a time). Each A in Fig. 1 is an input sample to be multiplied by a filter tap. Since the maximum number of supported taps in this architecture is four, there are four inputs. The shifts and adds are responsible to generate each approximate filtered value and the multiplexers are responsible to select which shifter should be used for each filter.

The first BMM operation over the inputs is a shift-left of n bits. These operations are equivalent to multiplications by 2^n. Since 90 different filters are supported by this architecture, some values for n are re-used to represent different tap values. The $A0$ and $A3$ input samples are processed only with a single shift-left, implying in a higher imprecision level, as previously discussed. On the other hand, the central input samples $A1$ and $A2$ are submitted to a lower level of imprecision, since two shift-add operations are available to calculate both taps, increasing their precision.

The output of multiplexers connected to $A0$ and $A3$ inputs have an additional controlled complement-of-two operation (C2 in Figure 1) to correct the results signal. The signal inversion is used to generate negative taps and it is not applied if a positive tap is required. Then, all outputs are added, and the last step is a 7-bit shift-right to divide by 128 the previous result, generating the interpolated sample.

Figure 2 presents the top-level Approximate Multiplierless 4-tap Filter (AM4F) architecture. Both H-BMM and V-BMM are the very same filter presented in Figure 1, where the difference in their names is only to highlight either is a horizontal (H-BMM) or vertical (V-BMM) instance of the Basic Multiplierless Multi-filter. Then, the H-BMM instances are connected to the V-BMM instances through a shift-register-chain of 4 positions. These positions correspond to the 4 BMM inputs (As in Figure 1).

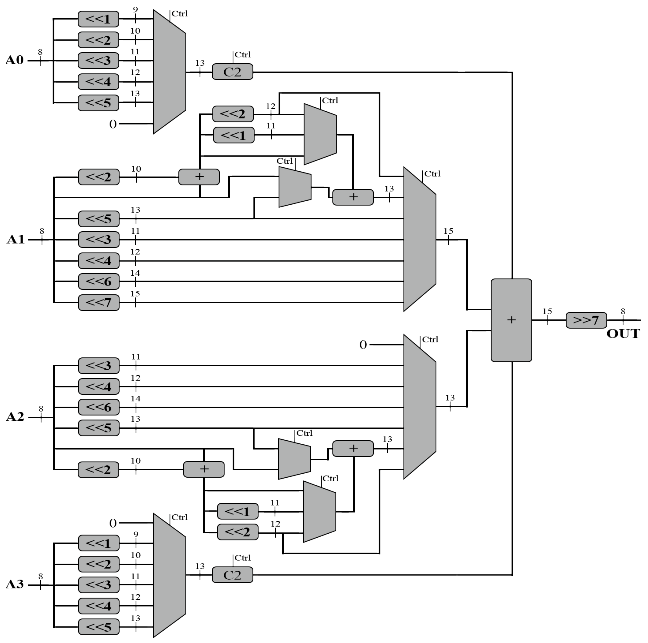

Fig. 1. Basic Multiplier-less Multi-filter (BMM).

The AM4F was designed to process 4×4 blocks, which is the smallest block size supported by AV1. Since any bigger block can be build up from 4×4 sub blocks, only by correctly splitting the input matrix and overlapping the samples, the architecture supports all AV1 block sizes, ranging from 4×4 to 128×128. However, both division of the input matrix and combination of the output blocks are not processed by this architecture, being necessary a different hardware.

The precise AV1 interpolation requires a matrix with 11×11 samples to interpolate a 4×4 block. Since always an extra 7 samples (vertical and horizontal) border is required to interpolate any smaller block size, reducing the number of filter taps, also leads to a reduction in these borders. Therefore, since four taps where removed, 4 rows and 4 columns can also be removed from this matrix, implying in a 7×7 matrix and in a reduction of 59.5% in data fetched from the memory.

The architecture was designed to read one line of this 7×7 matrix at each clock cycle and, then, shift the registers until

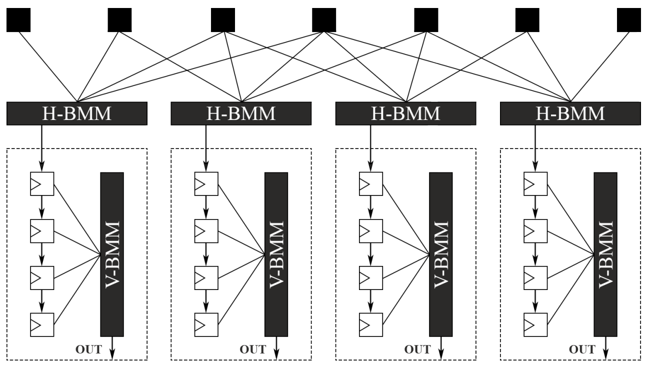

Fig. 2. AM4F: Approximate FME Interpolation Architecture.

filling the chain (these shifts correspond to the combinational distribution of the sample inputs, as done in the H-BMM). The AM4F takes four clock cycles to start processing the V-BMM instances and three clocks to finish the processing, taking seven clocks to generate an interpolated 4×4 block.

Both the memory usage and memory bandwidth are highly dependent of the whole AV1 hardware implementation, and vary according to the video. Independently of the memory technology, the 59.5% reduction is related to the number of samples fetched from the memory on the FME interpolation process, when compared to a standard-defined precise FME.

VI. Synthesis Results and Comparisons

Three architectural versions of the AVI FME interpolation were synthesized to allow comparisons: the approximate architecture designed in this paper, a precise version without simplifications and the previous design of our group, proposed in [8]. The precise version uses multipliers and explore the same idea of a multi-filter structure. All architectures follow the same architectural template (presented in Figure 2) and were described in VHDL and synthesized for a 40nm TSMC standard-cells technology with 1.1V [12] using Cadence RTL Compiler tool [13]. The power results were generated using the default tool switching activity (20%). The gate count was calculated based on 2-input NANDs size ($0.9408\mu m^2$). The frequency was defined as 686MHz since this is the required frequency to process UHD 8K@30fps for the precise version.

Table III presents the synthesis results along with a comparison with related works. As discussed before, only three related works were found in the literature reporting hardware designs for the AV1 interpolation, i.e., [7], [6], and [8]. A fair comparison with works [7] and [6] is not possible, since the work in [7] is focused on the motion compensation (decoder) and the work in [6] supports only the Regular filters family (15 filters). Even in a more complex scenario, our architecture reached an expressive lower area and power dissipation, even with the required higher operation frequency. Our previous work in [8] is the unique published work that can be fairly compared with this work.

The comparison with the precise version showed that our architecture uses 84.6% less area and dissipates 83.4% less power than the precise version. When compared to [8], the gains were of 41.8% in area and 7.1% in power. At this point it is important to highlight that the power results considered only the static and dynamic power of the designed hardware and did not consider the power reduction caused by the reduction in the memory bandwidth.

TABLE III
Synthesis results and comparison with related works

Work	Tech	Tool	Freq. (MHZ)	Gates (K)	Power (mW)
[7]	TSMC 40nm	90 MC Filters	279.9	141.7	81.3
[6]	STMicro 65nm	15 Regular Filters	344.8	270.4	130.7
Precise	TSMC 40nm	90 FME Filters	686.0	275.7	149.7
[8]			686.0	72.6	26.7
AM4F			**686.0**	**42.2**	**24.8**

VII. Conclusion

This paper presented an optimized approximate architecture for the AV1 FME interpolation filters that can process up to UHD 8K@30fps. This architecture is a memory-aware, low-area, low-power, and high-throughput hardware design exploring approximate computing which supports all 90 AV1 FME interpolation filters.

The original AV1 FME filter taps were approximated to hardware-friendly values and the maximum number of filter taps was limited to four. Besides important improvements in area, and power, the proposed approximations allowed a memory bandwidth reduction of 59.5% in comparison to the state-of-the-art solutions. When considering a precise version, our architecture is also able to reduce the area in 84.6% and reach power savings of 83.4% (without consider the power reduction generated by the memory bandwidth reduction), at a cost of 1.58% in BD-BR.

Only a previous published work of our group allows a fair comparison. In this case, the gains were 41.8% in area and 7.1% in power (without consider the memory bandwidth reduction), with a degradation of 1.04% in BD-BR.

References

[1] T. I. Express. (2020, mar) COVID-19 impact: Streaming services to dial down quality as internet speeds fall. [Online]. Available: https://indianexpress.com/article/technology/tech-news-technology/coronavirus-internet-speeds-slow-netflix-hotstar-amazon-prime-youtube-reduce-streaming-quality-6331237/

[2] Statista. (2019, oct) Global mobile data traffic 2017-2022. [Online]. Available: https://www.statista.com/statistics/271405/globalmobiledata-traffic-forecast.

[3] Cisco. (2018, oct) Cisco visual networking index: Forecast and methodology, 2016–2021 - cisco. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[4] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, and J. Bankoski, "A technical overview of AV1," *Proceedings of the IEEE*, pp. 1–28, 2021.

[5] JVET. (2021, abr) VVC codec library. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

[6] D. Freitas, R. da Silva, I. Siqueira, C. M. Diniz, R. A. L. Reis, and M. Grellert, "Hardware architecture for the regular interpolation filter of the AV1 video coding standard," *Eusipco*, 2020.

[7] R. Domanski, J. Goebel, W. Penny, M. Porto, D. Palomino, B. Zatt, and L. Agostini, "High-throughput multifilter interpolation architecture for AV1 motion compensation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 5, pp. 883–887, 2019.

[8] R. Domanski, W. Kolodziejski, M. Porto, G. Correa,, B. Zatt, and L. Agostini, "Low-power and high-throughput approximated architecture for AV1 FME interpolation," *ISCAS*, 2021.

[9] N. W. Group. (2020, aug) Video codec testing and quality measurement. [Online]. Available: https://tools.ietf.org/html/draft-ietf-netvc-testing-09

[10] AOMedia. (2020, may) AV1 codec library. [Online]. Available: https://aomedia.googlesource.com/aom/+/refs/tags/v2.0.0.

[11] G. Bjontegaard, "Improvements of the BD-PSNR model. itu-t sc16/q6," in *35th VCEG Meeting, Berlin, Germany, Doc. VCEG-AI11*, 2008.

[12] TSMC. (2018, aug) 40nm technology. [Online]. Available: http://www.tsmc.com/english/dedicatedFoundry/technology/40nm.htm.

[13] Cadence. (2018, aug) Encounter RTL compiler. [Online]. Available: http://www.cadence.com.