Journal of
Low Power Electronics and Applications

MDPI

Article

Optimized VLSI Architecture of HEVC Fractional Pixel Interpolators with Approximate Computing

Stefania Preatto, Andrea Giannini, Luca Valente *, Guido Masera and Maurizio Martina

Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy; stefania.preatto@studenti.polito.it (S.P.); andrea.giannini@studenti.polito.it (A.G.); guido.masera@polito.it (G.M.); maurizio.martina@polito.it (M.M.)
* Correspondence: s257373@studenti.polito.it

Received: 15 June 2020; Accepted: 11 August 2020; Published: 17 August 2020

Abstract: High Efficiency Video Coding (HEVC) is the latest video standard developed by the Joint Video Exploration Team. HEVC is able to offer better compression results than preceding standards but it suffers from a high computational complexity. In particular, one of the most time consuming blocks in HEVC is the fractional-sample interpolation filter, which is used in both the encoding and the decoding processes. Integrating different state-of-the-art techniques, this paper presents an architecture for interpolation filters, able to trade quality for energy and power efficiency by exploiting approximate interpolation filters and by halving the amount of required memory with respect to state-of-the-art implementations.

Keywords: VLSI architecture; approximate computing; HEVC fractional pixel interpolators

1. Introduction

Nowadays, the ubiquitous presence of cameras in daily lives requires high video compression efficiency with respect to storage size, bitrate and energy consumption while retaining an acceptable visual quality. The most recent video compression standard developed by the Joint Video Exploration Team is the High Efficiency Video Coding (HEVC) standard, also known as H.265, which is able to offer a doubled compression ratio over the preceding standard, the H.264 or the Advanced Video Coding (AVC) standard, while retaining a comparable visual quality. While the overall functional structure of the two standards is similar, HEVC can provide better results (higher coding efficency, lower bitrates) than H.264/AVC by exploiting a more complex partitioning scheme with many more prediction and transform possibilities [1–3]. These algorithmic techniques enable a significant decrease in bitrate at the cost of an increase in computational complexity and external memory bandwidth. In this regard, several works were proposed to cope with the strong limitations introduced by the I/O schemes and the Memory access Bandwidth on the algorithm's performance [4–6]. However, this work, like many others [7–13], focuses exclusively on the optimization of the computational kernel in order to maximize its throughput: Different works in the literature, including [14], show that a relevant portion of HEVC complexity is due to the motion estimation. Indeed, HEVC features a two-step motion compensation process, which first works on different search window sizes and then exploits an interpolation step for fractional pixel search refinement. This second step relies on a separable 2D interpolation filter. In the last years several architectures for HEVC interpolation filters have been proposed in the literature, e.g., [7–13]. Most of the published architectures, including [7,9–12], concentrate on the standard HEVC interpolation filters, hereinafter referred to as legacy filters. However, the work in [15] showed that algorithmic-level approximate computing can be exploited to achieve energy efficiency in HEVC decoding and, to the best of our knowledge, [13] is the first paper showing an architecture for interpolation filters where energy/quality trade-offs can be exploited. In particular, in [13] multiply units are dynamically multiplexed, thus allowing to reduce the filter order at the runtime. Even though this solution consumes the lowest amount of energy per interpolated pixel among state of the art solutions [8–10,12], further optimizations can be conceived to reduce the area overhead and to increase the throughput.

2. Contribution

In this work, we stem from the architecture in [13] and we coherently integrate it, for the first time, with other state-of-the-art techniques [10,15–17] and an alternative scheduling algorithm. The result obtained is a new optimized interpolation filter architecture, where important optimizations are introduced: (i) The amount of memory is drastically reduced, (ii) multipliers are substituted with adders by extending the optimized structure presented in [10] for legacy filters to the case of lower order filters. Moreover, we find an appropriate internal architecture for the adders that are involved in the filtering operation, to further increase the throughput of the system, and limit drawbacks in terms of area overhead and power dissipation. The paper is organized as follows: Section 3.1 aims to give an overview of legacy and approximate lower order HEVC filters. Then, in Section 3.2, the used set of interpolation filters for encoding and decoding in conjunction with the proposed architecture will be presented. In Section 3.4 precise and approximate adders solutions are applied to the suggested architectures. Lastly, Section 4 will present the obtained results and discuss them.

3. Materials and Methods

3.1. Interpolation Filters

The standard HEVC interpolation filters for subpixel motion estimation and compensation, show different structures for Luma and Chroma channels [18] and feature several differences with respect to the AVC ones, as it is more precisely presented in [19]. Namely, an eight-tap and a seven-tap DCT-based filters are used for the HEVC fractional pixel interpolation: Pixels at half-sample position ($\alpha = 1/2$) are originated using the eight-tap filter in contrast to the six-tap one used by H.264/AVC. Moreover, the quarter-sample pixels ($\alpha = 1/4$) are evaluated using the seven-tap filter without any average operation between two neighbouring sub-samples. This allows reducing the error due to intermediate rounding and removing the two-stage interpolation process of the H.264/AVC algorithm. Moreover, HEVC uses a single separable interpolation process for all fractional position pixels. Overall, the error due to cascaded rounding operations in H.264/AVC can be reduced from 33/128 to only 1/128 in HEVC for some of the interpolated pixels [19]. The Luma filter coefficients are reported in Table 1. Chroma interpolation (Table 2) is similar: Since the Chroma signals are smoother than the Luma ones, four four-tap filters are used, thus there is no need to use longer filters for high frequency Chroma components in contrast to H.264/AVC, which uses a two-tap bilinear filter.

Table 1. Luma filter coefficients.

N_{tap}	$\alpha = 1/4$	$\alpha = 1/2$
Legacy	$-1, 4, -10, 58, 17, -5, 1$	$-1, 4, -11, 40, 40, -11, 4, -1$
7	$-1, 4, -10, 58, 17, -5, 1$	$-1, 4, -11, 40, 40, -11, 3$
5	$1, -6, 20, 54, -5$	$2, -9, 40, 40, -9$
3	$-4, 20, 48$	$-9, 41, 32$
1	64	64

In addition, it is possible to present a computational complexity-wise optimization of the legacy filters. It has already been shown that approximation techniques can drastically reduce the energy consumptions [15]. The current work focuses on the same type of approximate filters proposed in [15] for HEVC encoding, which are reported in Tables 1 and 2, respectively. Figures 1 and 2 show the amplitude response of the adopted half-pel approximate filters with respect to the legacy ones used in the HEVC standard. As it can be observed, the response of the 5-tap Luma approximate filter is slightly different from the original 8-tap one, thus creating some artifacts. This effect is even more evident for the 3-tap filter which may cause losses in texture details. A similar behavior can be observed for the chrominance filter. Since the focus in [15] is on the decoder side, this current work aims to complete the analysis on the encoder side as well.

Table 2. Chroma filter coefficients.

N_{tap}	$\alpha = 1/8$	$\alpha = 2/8$	$\alpha = 3/8$	$\alpha = 4/8$
Legacy	$-2, 58, 10, -2$	$-4, 54, 16, -2$	$-6, 46, 28, -4$	$-4, 36, 36, -4$
3	$-3, 62, 5$	$-5, 58, 11$	$-7, 51, 20$	$-6, 42, 28$
2	$57, 7$	$50, 14$	$41, 23$	$32, 32$
1	64	64	64	64

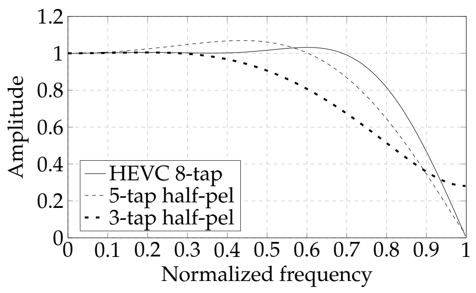

Figure 1. Half-pel approximate Luma interpolation filter amplitude response comparison.

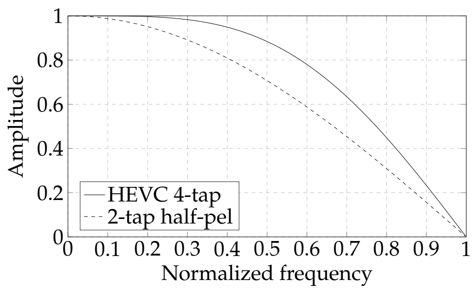

Figure 2. Half-pel approximate Chroma interpolation filter amplitude response comparison.

The approximate filters have been implemented into the HM 16.15 software model [18], and PSNR analysis has been performed to clearly assess the impact of the approximate filters on the rate-distorion (RD) performance of the HEVC system. The combined PSNR (i.e., PSNR_{YUV}) is calculated as the weighted sum of the PSNR per frame of its individual components (PSNR_{Y}, PSNR_{U}, PSNR_{V}), as suggested in [1]. A complete analysis of the HM system is carried out by applying approximate interpolation filters for several different configurations with the proposed architecture: In addition to the entirely correct or approximate solution, two hybrid conditions are added (legacy encoder and approximate decoder and vice-versa). Some significant experimental results are depicted in Figures 3 and 4, where the RD-curves obtained for the BasketballDrive test sequence with the Random Access and Low-Delay profiles (as recommended by the Common-Test-Conditions [20]) are shown.

Figures 3 and 4 highlight that the solution with both approximate encoder and decoder is the best one to obtain an acceptable PSNR, especially for higher bitrates. It is also possible to notice that the solution with an approximate encoder performs better than the one with an approximate decoder, with a reasonable PSNR for low bitrates.

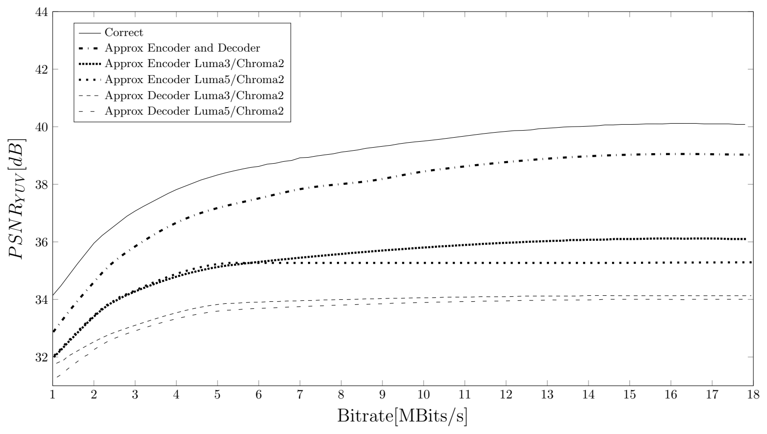

Figure 3. PSNR comparison between ideal processing and different approximate-computing options (BasketballDrive [20], 1920 × 1080, Random Access).

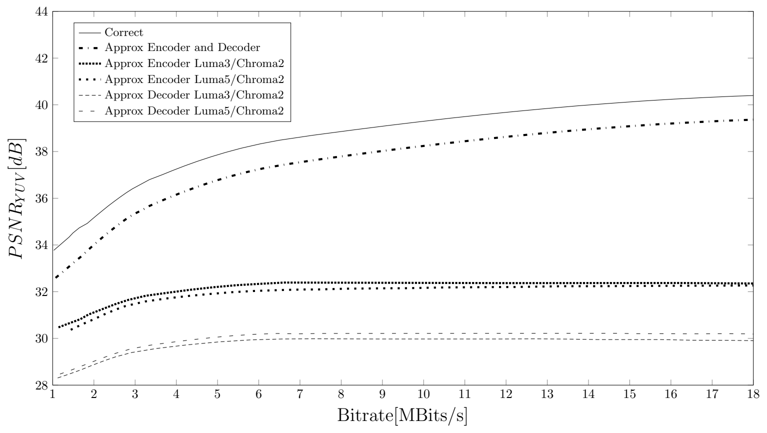

Figure 4. PSNR comparison between ideal processing and different approximate-computing options (BasketballDrive [20], 1920 × 1080, Low Delay).

3.2. Proposed Architecture

Based on the profiling results available in [14], it is clear that fractional-pixel interpolation used by motion estimation and by motion compensation is one of the most CPU time expensive blocks of HEVC (up to 43% of the time is spent in motion compensation on a ARM ISA, and up to 49% for a ×86 ISA [14]). In this Section, a hardware architecture for the DCT-based interpolation filter (DCT-IF) is derived by exploiting multiplier-less solutions, hardware reconfigurability and approximate computing in order to reduce the energy consumption, while ensuring a certain energy-quality tradeoff. Each Luma and Chroma prediction block interpolation process is performed using two separable 1D filters for the horizontal and the vertical direction respectively (horizontal first in HEVC). According to the HM reference software [18], there are two options to perform a 2D separable filter computation:

- Parallel Interpolation filter (Figure 5)
- Folded Interpolation filter (Figure 6)

The first one is a straightforward hardware implementation of the algorithm: It relies on two different filtering units and an intermediate buffer (Figure 5).

Consecutive pixel rows of a prediction block can be provided to the horizontal 1D filter in consecutive clock cycles, then the output can be temporarily stored in a buffer for the following vertical filtering process. In this case, with 16-bit samples (as required by the standard), for the legacy 8-tap filter, the buffer size is roughly

$$(N_{tap,\max} - 1 + W_{\max}) \cdot H_{\max} = 64 \cdot (64 + 8 - 1) = 71 \cdot 64\,\text{samples} = 9.09\,\text{kB} \tag{1}$$

where W_{max} is the maximum width of the prediction block and H_{max} its maximum height. This solution allows for a very high throughput, since both the filters can work in parallel with new data after the first prediction block has been horizontally interpolated. The key weakness of this option is the large amount of memory required to store the intermediate partly interpolated samples.

On the other hand, the folded structure does not reduce the size of the internal buffer but is able to save some hardware at the expense of throughput (Figure 6).

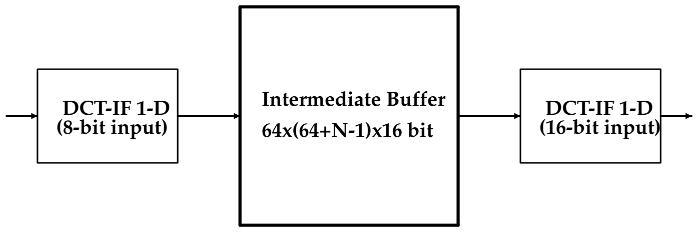

Figure 5. Parallel interpolation filter architecture with intermediate block buffer.

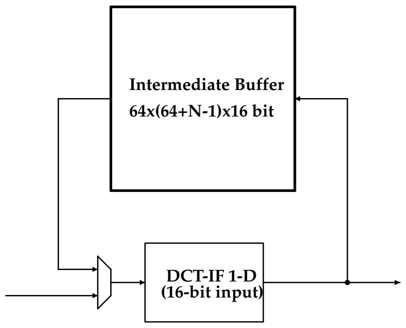

Figure 6. Folded interpolation filter architecture with intermediate block buffer.

One advantage of the parallel interpolation scheme derives from the fact that a 1D filtering operation needs just a number of samples equal to the filter number of taps before starting the filtering process. In addition, in the parallel scheme, there is no need to wait for one entire prediction block to be partly sub-sampled before starting a new 1D filtering process. As these features allow reducing the latency and the memory cost, the parallel option has been selected as the starting point for our optimization. Figure 7 shows the scheduling of the implemented filter. Each location represents a pixel, the ones highlighted in the upper part of the Figure refer to the input pixels provided to the first 1D filter, the locations in the middle of the figure refer to the 1D filtered samples outputs of the first filter and the shaded pixels in the bottom part represent the output of the two-dimensional interpolation.

Let N_{tap} be the number of coefficients of the DCT-IF filter: As soon as $N_{tap} - 1$ columns have been stored in an input buffer, the first 1D filter starts computing one pixel per cycle. The second filter waits only for the availability of $N_{tap} - 1$ partly interpolated samples, then it can start interpolating in parallel to the first stage filter. The throughput is the same as the previous solution, considering that when the vertical filter reaches the last column sample, it should wait for $N_{tap} - 1$ cycles, because of the data dependencies between the two filter stages. With this scheduling, it is possible to move the buffer to the input of the system, as shown in Figure 8.

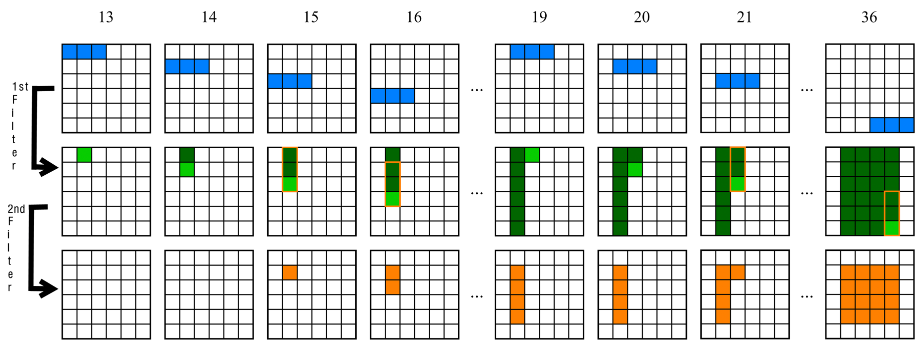

Figure 7. Filter alternative scheduling example with time stamp in clock cycle count with $N_{tap} - 1 = 3$.

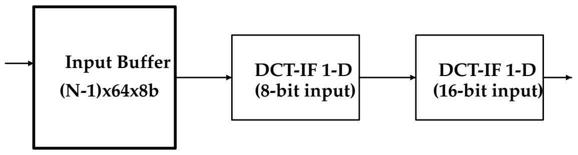

Figure 8. Alternative scheduling filter architecture.

Thus, the proposed scheduling algorithm greatly reduces the required memory buffer to just:

$$(N_{tap,\max} - 1) \cdot W_{\max} = 7 \times 71 \,\text{samples} = 497\,\text{B} \tag{2}$$

thanks to the fact that now the samples are 8 bit wide, which means about a 18× factor reduction with respect to the initial size of 9.09 kB. This enables an on-chip implementation, which is important for the case of multiple filter instances required in high resolution video sequences.

3.3. One-Dimensional DCT-IF Architecture

The single 1D DCT-IF architecture can be easily implemented and pipelined as a direct form FIR architecture using a spatial delay line and a set of multiply and accumulate blocks to get the final result:

$$y[n] = \sum_{i=0}^{N} B_i \cdot x_i[n-i], \quad i = 0, \ldots, N \tag{3}$$

where B_i are the filter coefficients that depend on the filter applied in the interpolation process. In order to reduce the amount of energy per computation required by the 1D architecture, a commonly adopted operand substitution method relies on replacing all the multiplications with additions and shifting operations. This is a particularly suitable technique with filter architectures since the multiplier coefficients are known at design time. However, keeping a certain order of coarse-grained reconfigurability in a multiplier-less approach is not easily achievable as with direct form FIR filters. Several methods have been proposed in order to find a reconfigurable multiplier-less 1D filter architecture ([7,9,11] and others). Among those, the one implemented here is a slightly modified version of the filters introduced by Diniz et al. in [10]. The Luma legacy datapath is depicted in Figure 9, with the Luma and Chroma legacy multiplier-less solutions in Figures 10 and 11.

The element composing the 2D interpolation filter above are the following:

- Shift Register Bank (SRB): This represents the input buffer. As soon as it receives a pixel row in input it sends it to the RtU and the content of the corresponding Shift Register is shifted.
- Address Counter (CNT): This is a programmable counter that points to a SRB shift register. It fills the lines used to start the filtering process.
- Routing Unit (RtU): This redirects the output of the memory bank toward the inputs of the filter.
- DCT-IF: This represents the Luma and Chroma legacy multiplier-less architecture described below.

- Rounding Unit (Round): This applies an half-up rounding at the output of the second filter, when required.
- Clipping Unit (Clip): This manages the arithmetic saturation.

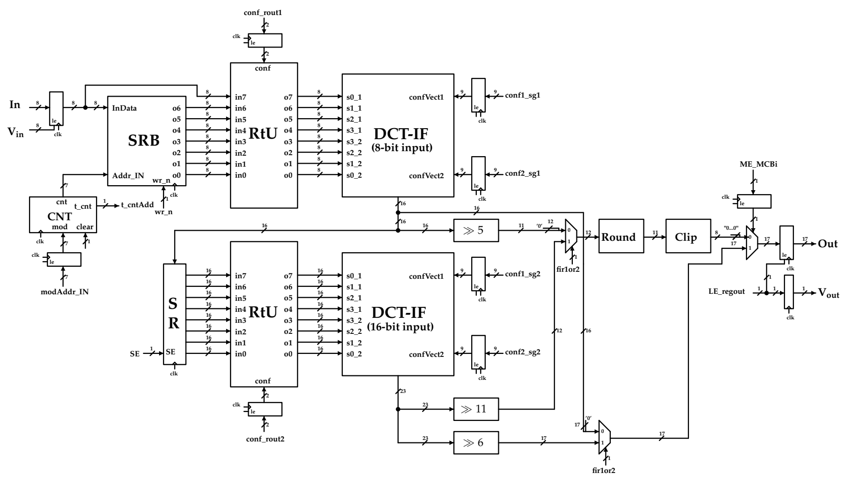

Figure 9. Datapath 2D DCT-based interpolation filter (DCT-IF) Luma legacy.

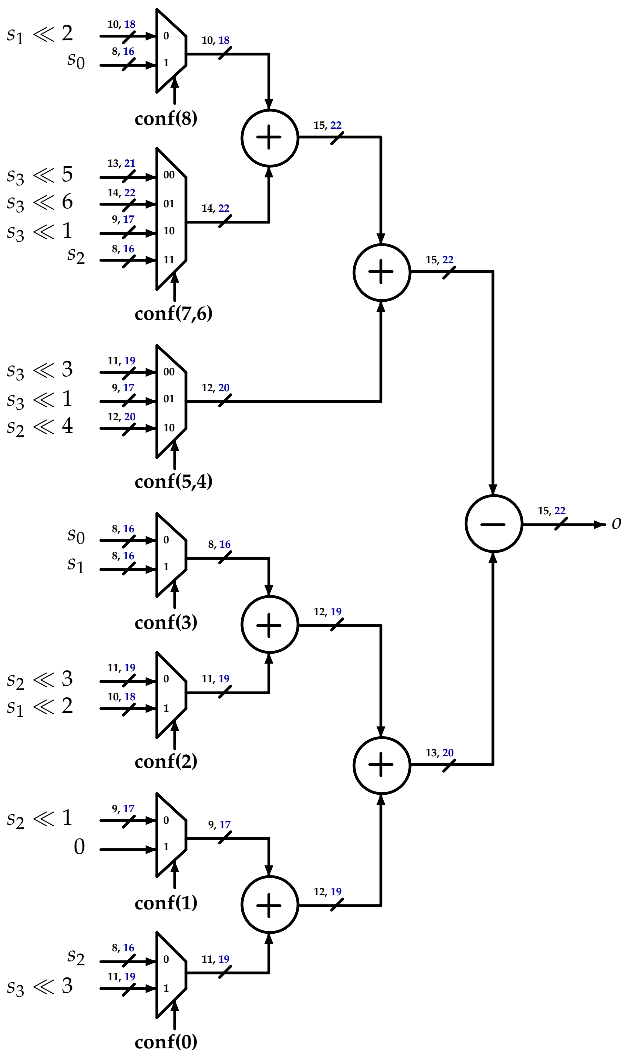

Figure 10. Reconfigurable legacy Luma filter.

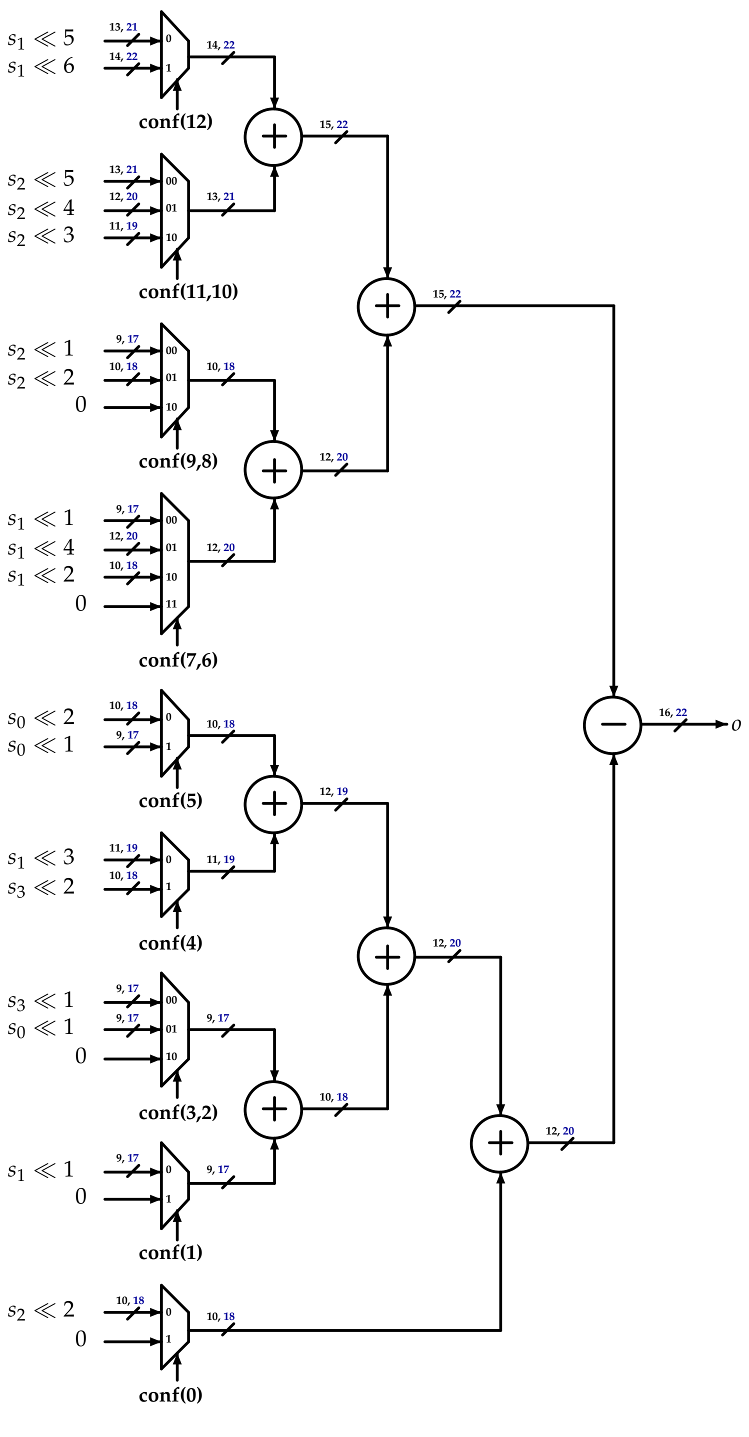

Figure 11. Reconfigurable legacy Chroma filter.

The 3-tap and 5-tap configurations were considered for the Luma datapath, the 2-tap one for the Chroma datapath (Tables 1 and 2). Higher order Luma filters were not considered because the energy benefit results reported in [13] do not encourage such a choice, since it gave no energy consumption reduction with respect to the legacy implementation. Figures 12 and 13 report the 5-tap and the 3-tap reconfigurable Luma DCT-IFs implementations respectively, while in Figure 14 the reconfigurable Chroma 2-tap architecture is shown. Table 3 gives the add and shift replacements implemented in our architecture with the respective multiplier coefficients.

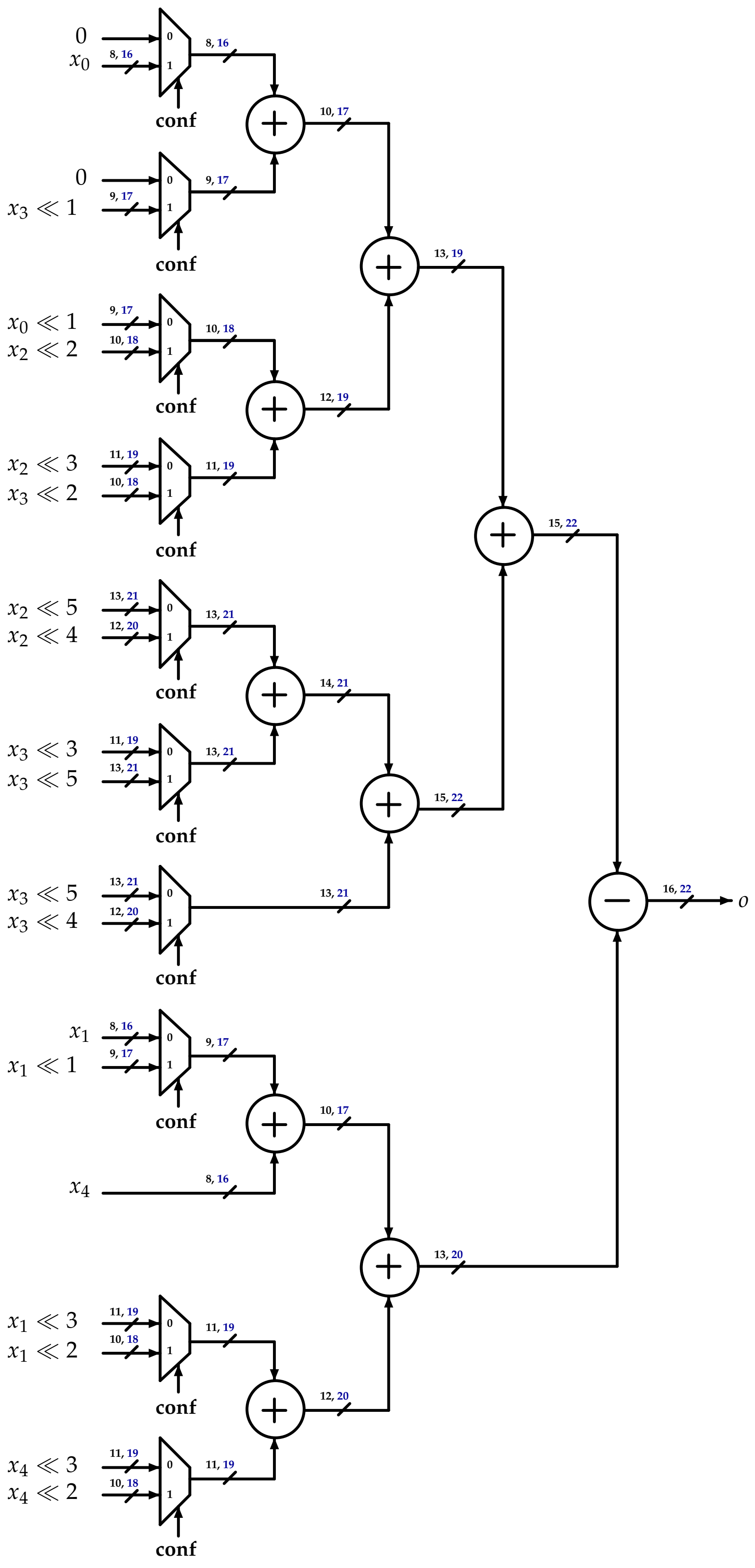

Figure 12. Reconfigurable approximate Luma 5-tap filter.

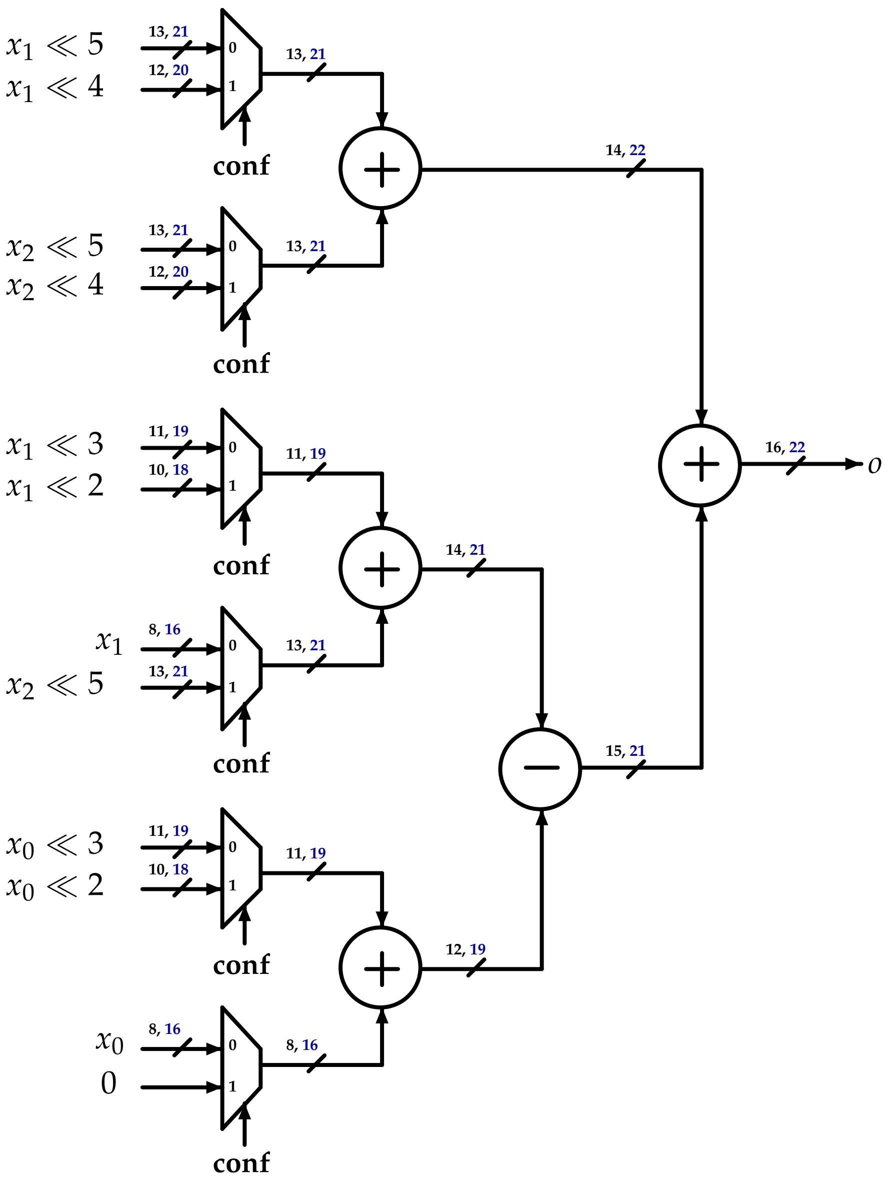

Figure 13. Reconfigurable approximate Luma 3-tap filter.

Table 3. Luma and Chroma approximate coefficient multiplications replaced by add/shift.

Shift—Coeff	1	2	4	5	6	7	9	14	20	23	32	40	41	48	50	54	57
x	+			+		−	+			−			+				+
$x \ll 1$		+			+			−							+	+	
$x \ll 2$			+	+	+				+							+	
$x \ll 3$						+	+			−		+	+				−
$x \ll 4$								+	+					+	+	+	
$x \ll 5$										+	+	+	+	+	+	+	
$x \ll 6$																	+

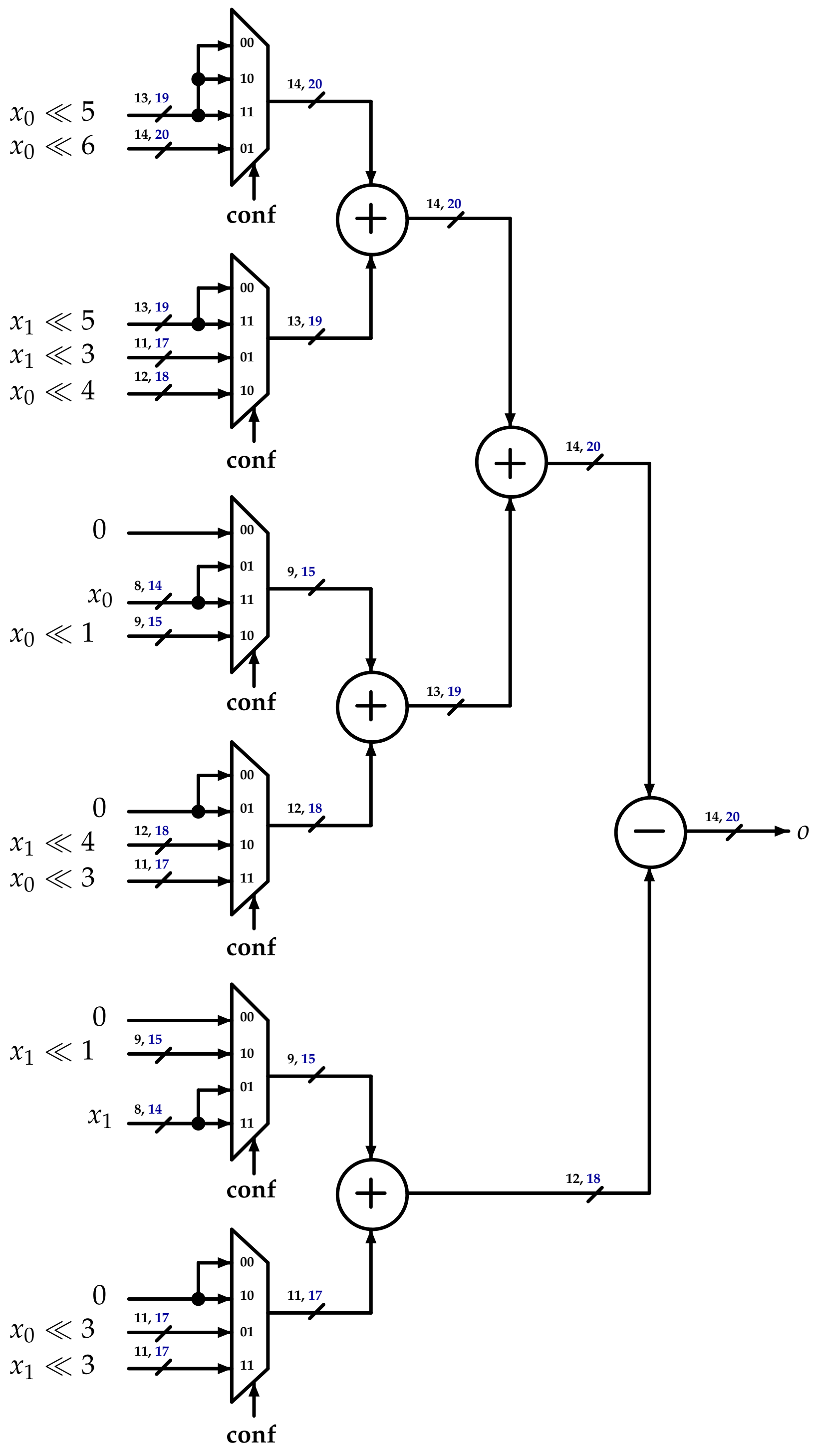

Figure 14. Reconfigurable approximate Chroma 2-tap filter.

The Luma datapath of the proposed architecture is shown in Figure 15. This structure is able to perform both the approximate and the legacy filters to better exploit energy-quality scalability. The datapath is composed by different parallel filter branches, each one related to a specific reconfigurable DCT-IF implementation. Depending on the input requirements, multiplexers select which branch output should be considered for the first and for the second stage. As it will be shown in Section 4, it is important to block the switching of the inputs of the unused filters to reduce the total activity and consequently the power consumption. Rather than the use of demultiplexers to the RtUs' inputs, this blocking behaviour is directly embedded in the RtUs by means of AND ports (not showed in the Figure).

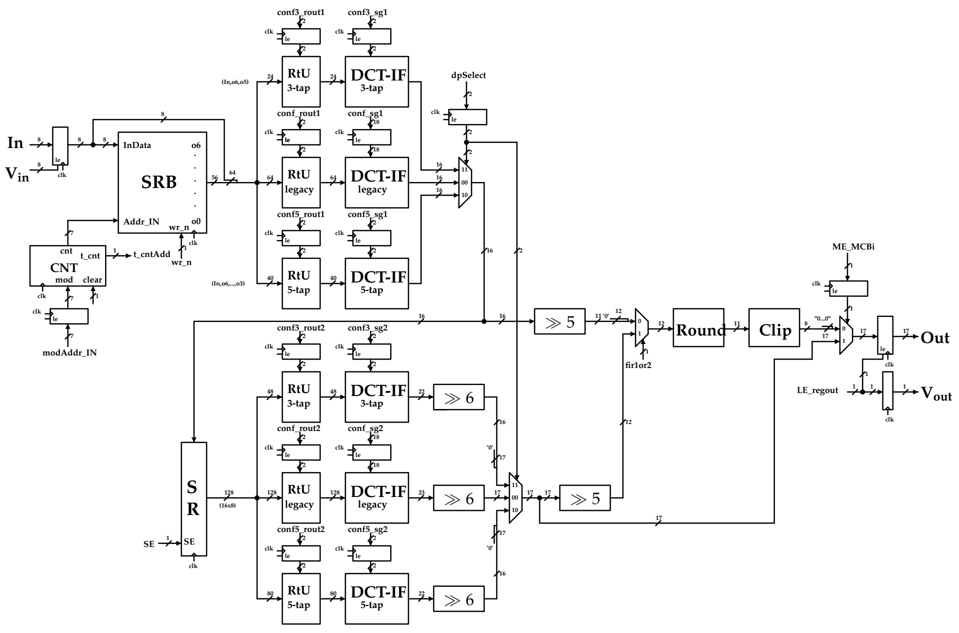

Figure 15. Datapath 2D DCT-IF Luma approximate.

3.4. *Optimized Adder Architectures*

Additional improvements can be applied to the proposed architecture at different levels of the design, in order to further increase the throughput of the system. Indeed, two different approaches are proposed: An exact solution, regarding the adoption of parallel and prefix adders is described here below, while an approximate alternative, with Generic Accuracy Configurable Adders on the second stage interpolation filters is described in the following Section.

Parallel Prefix Adders (i.e., PPAs) are able to speed-up the carry computation, which is the bottleneck in the critical path evaluation [21]. In the proposed work we combined two different topologies:

- Han–Carlson (H.C.): This achieves a good trade-off between complexity, fan-out and perfomance by combining outer Brent–Kung layers and inner Kogge–Stone layers.
- The topology in [16], which uses outer Brent–Kung layers and inner Ladner-Fischer layers. This solution is able to shorten the critical path delay with respect to the tree of prefix operators.

The [16] topology is applied to the Chroma Legacy architecture as it shows the best improvements in performance, at the cost of a negligible area overhead and power dissipation. On the other hand, the Han–Carlson one is applied to the Luma Approximate because it guarantees the highest precision.

3.5. *Generic Accuracy Configurable Adders*

Generic Accuracy Configurable (GeAr) adders [22] support both an exact mode and an approximate mode, so allowing a dynamic tuning of the accuracy.

In this work we implement a particular version of the GeAr adder proposed in [17], which exploits a *Complementary Modules* scheme to limit the magnitude of the generated errors. Figure 16 is used to illustrate the concept. Two different types of adders, introducing errors of opposed polarity ($+\varepsilon$ and $-\varepsilon$), are used together with an Error Detection (ED) mechanism: When an error at the first adder is detected ($ED = 1$), it will select the adder, the output of the adder with negative error (A_{2b}) is selected as the final sum. In this way the total error is always kept between 0 and $|\varepsilon|$.

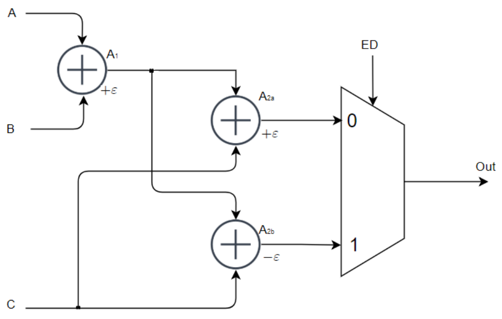

Figure 16. Scheme principle of complementary module.

As reported in [17], by breaking the carry-chain, a GeAr adder supports a generic model for block-based adders: It exploits multiple sub-adder units of equal length and allows the implementation of an error correction unit. So, given two N-bits operands to be added, a GeAr computes the sum through k L-bits ($L \leq N$) sub-adders, that perform the sum operation in parallel. Let R be the number of resultant bits contributing to the final sum, and P the number of previous bits used for the carry prediction for each sub-adder: The first one computes the precise sum over L = R + P bits, while all the other sub-adders are R-bit blocks. The carry-in is generated by a Carry Generator Unit, implemented as a P-bit Carry Look-Ahead adder.

The work in [17] also shows that it is possible to define two different types of GeAr: Standard GeAr and Complementary GeAr (CGeAr) and they differ between each other by just the input carry c_{in}. For the GeAr, it is always set to 0, while for the CGeAr is always set to 1, therefore, CGeAr and GeAr introduce errors of opposed polarity. This allows implementing *Complementary Modules* as circuits able to switch between a GeAr and a CGeAr. The final result is an approximate adder with an adaptive behaviour, with just the addition of two XOR gates instead of the any other user-driven EC logic [17].

The implementation adopted in this work, and shown in Figure 17, introduces three distinct sub-blocks ($k = 3$) and two 1-bit Carry Look-Ahead Adder ($P = 1$).

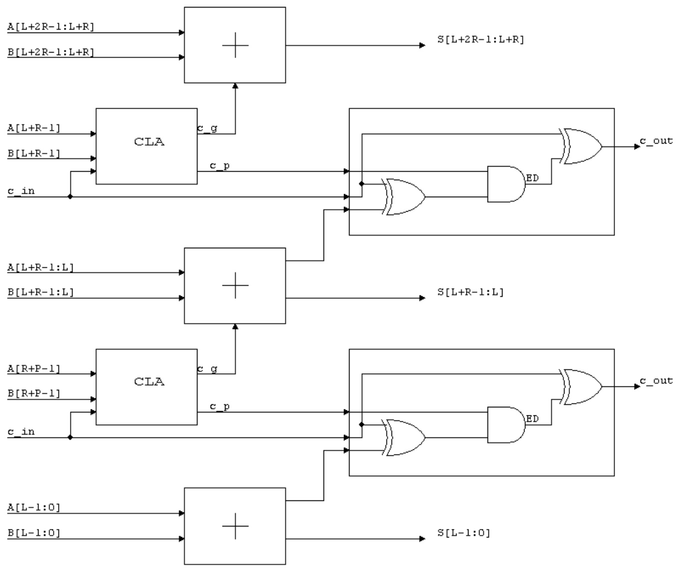

Figure 17. Architecture of GeAr and CGeAr k = 3, P = 1.

The logic circuit that computes the c_{in} for the jth sub-adder of the i^{th+1} adder uses the error detection signal of the previous GeAr block, according to this equation:

$$cin_{i+1,j} = cin_{i,j} \oplus ED_{i,j} \tag{4}$$

The ED signal of the jth sub-adder can be obtained as follows:

$$ED_j = cp_j \cdot (cin_j \oplus cout_{j-1}) \tag{5}$$

where cg and cp are the outputs of the jth CLA and, given $P = 1$, are equal to:

$$cg = A_j \cdot B_j + c_{in} \cdot cp$$
$$cp = A_j \oplus B_j$$

where A_j and B_j are the inputs of the jth CLA (in this case $A[R]$, $A[L + R - 1]$, $B[R]$ and $B[L + R - 1]$). The choice $P = 1$, simplifies the equations to obtain cg and cp. This reduces the occupied area overhead and the critical path delay, but generates a higher number of errors.

In the Luma Legacy architecture the majority of adders are chosen with an adaptive approximate configuration to earn in speed, area and energy efficiency. In order to assess the impact of this approach in terms of PSNR degradation on the entire HEVC system [18] we inserted in the model an error contribution ε' with the same probability distribution as the hardware interpolation filters architecture.

The probability density function that characterizes the interpolation process was derived by evaluating the difference between exact and approximated values and deriving the corresponding histogram. As reported in Figure 18 and Table 4 the error distribution is well modeled as it is composed by the superposition of three normal density functions with similar standard deviation and different mean. Finally, a random noise is generated following the three Gaussian statistics. This error is inserted in the HM software and the PSNR is evaluated for different sequences presented in [20], obtaining the results depicted in Figures 19–30. In this set of Figures, each couple of adjacent pictures refers to a given video sequence. Moreover, left pictures show the results obtained with the Random Access, while right pictures present PSNR results for the case of Low Delay access. These results show that the PSNR degratation introduced by the approximate encoder and decoder in the HEVC system is marginal as the maximum difference between the two PSNR is always between 0.4 dB and 1.8 dB, and no significant trade-off has been made. Indeed, the approximated adders were chosen to explore the maximum achievable throughput without significantly downgrading the performance. Thus, the main impact on the PSNR has to be attributed to the choice of N_{tap} [15].

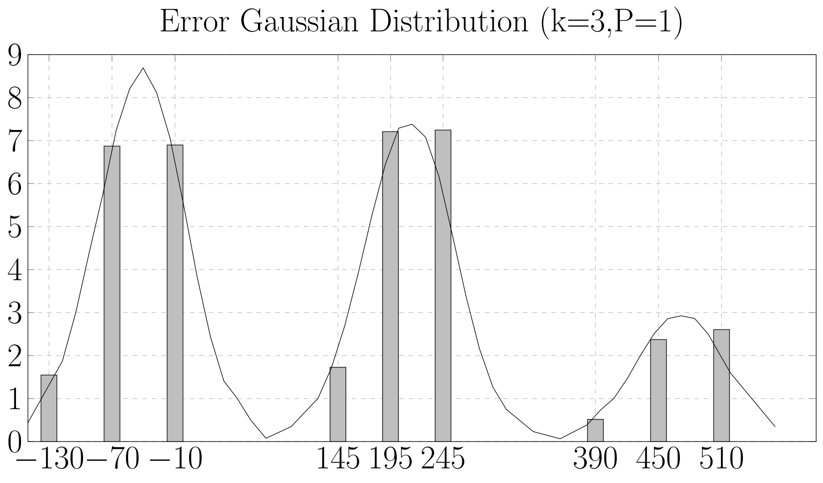

Figure 18. Probability density functions for error distribution k = 3, P = 1.

Table 4. Mean and standard deviation for Gaussian distributions k = 3, P = 0.

	Gaussian 1	Gaussian 2	Gaussian 3
μ	–41.09	214.23	472.73
σ	41.98	42.51	41.63

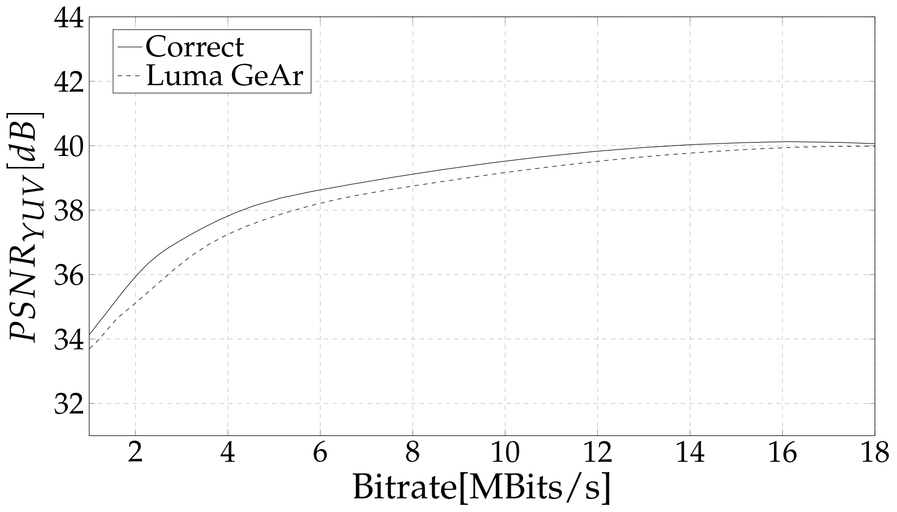

Figure 19. PSNR degradation with GeAr (BasketballDrive [20], 1920 × 1080, Random Access).

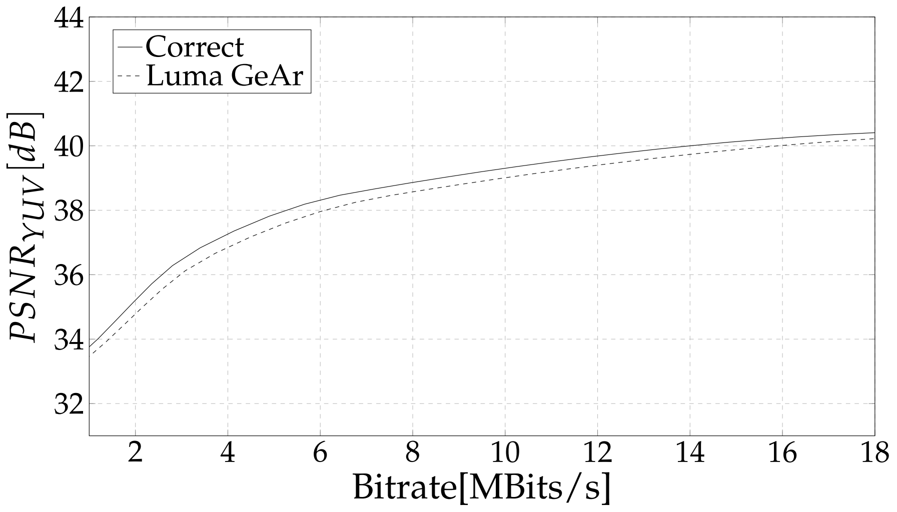

Figure 20. PSNR degradation with GeAr (BasketballDrive [20], 1920 × 1080, Low-Delay).

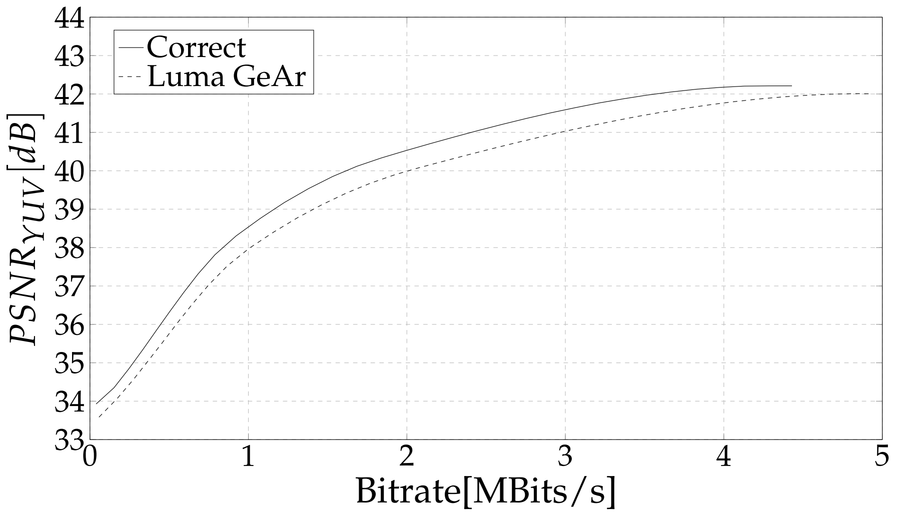

Figure 21. PSNR degradation with GeAr (Kimono, [20], 1920 × 1080, Random Access).

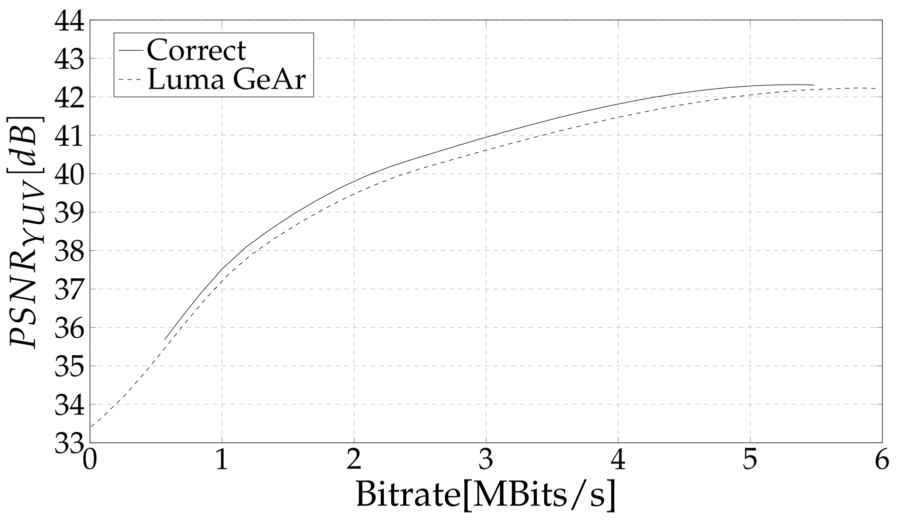

Figure 22. PSNR degradation with GeAr (Kimono [20], 1920 × 1080, Low-Delay).

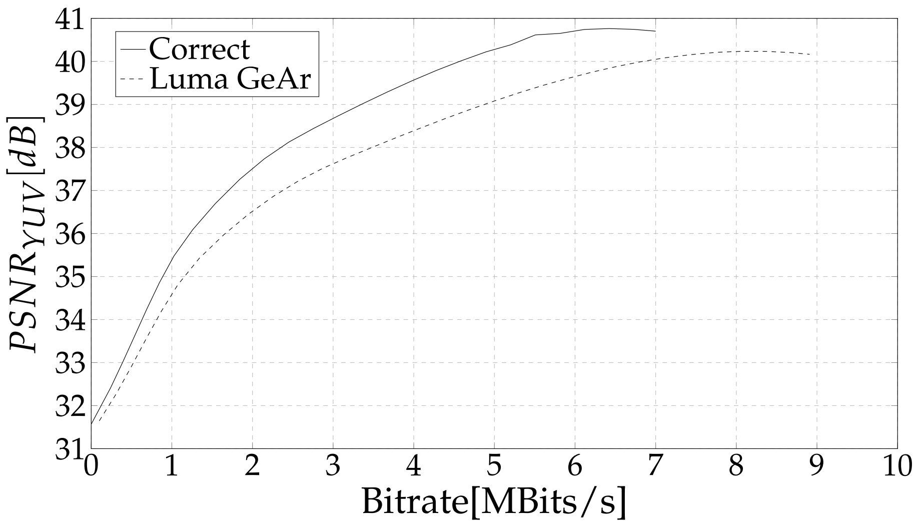

Figure 23. PSNR degradation with GeAr (ParkScene, [20], 1920 × 1080, Random Access).

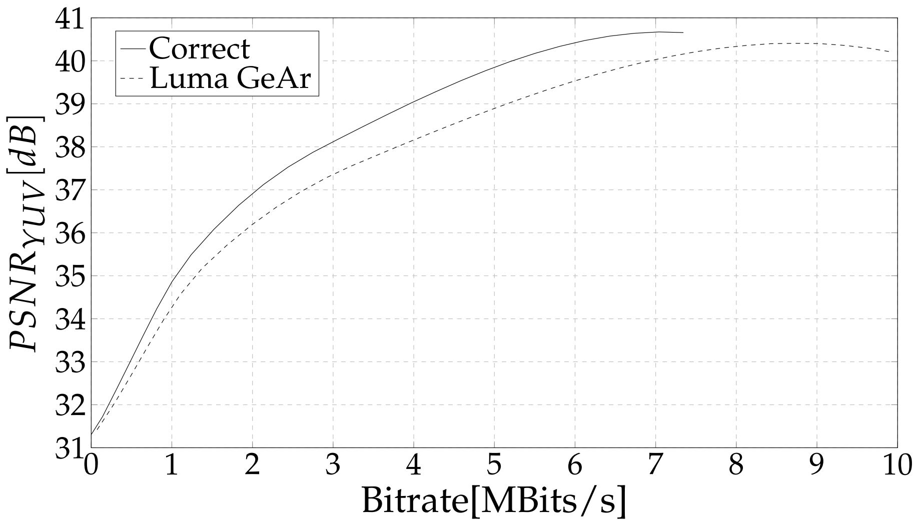

Figure 24. PSNR degradation with GeAr (ParkScene, [20], 1920 × 1080, Low-Delay).

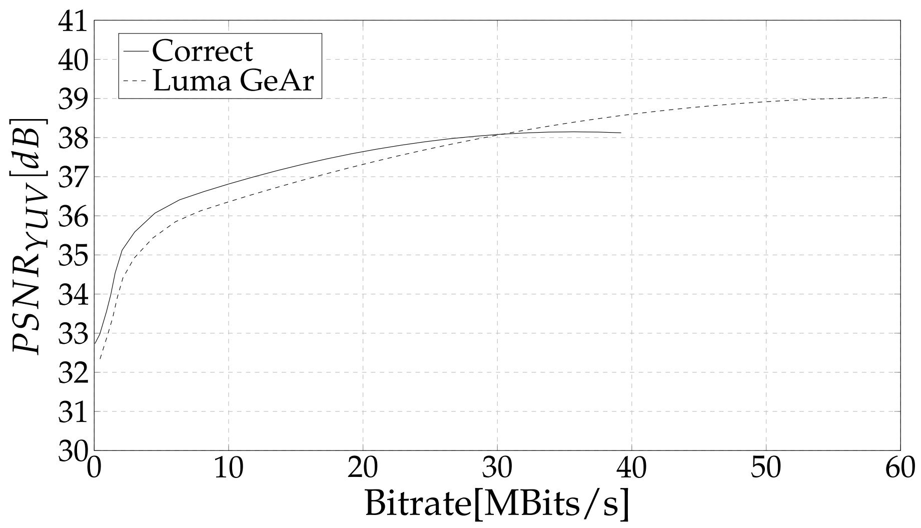

Figure 25. PSNR degradation with GeAr (BQTerrace, [20], 1920 × 1080, Random Access).

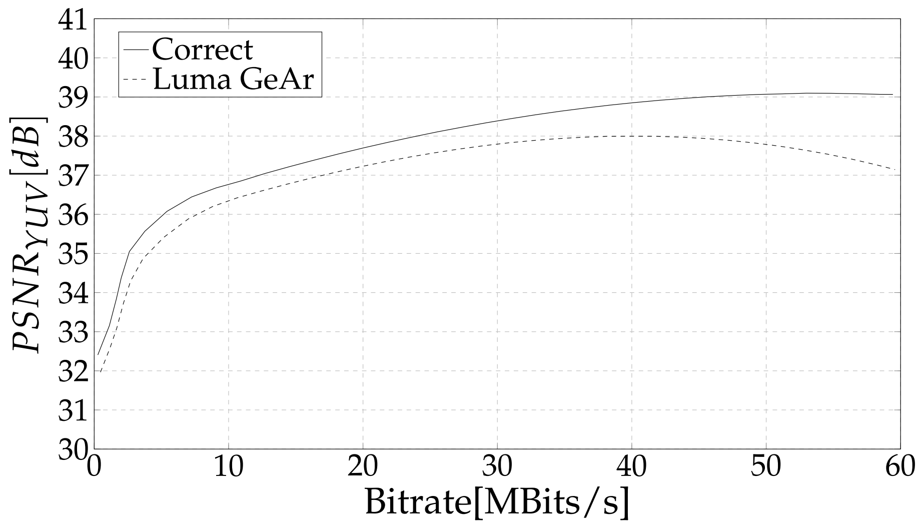

Figure 26. PSNR degradation with GeAr (BQTerrace, [20], 1920 × 1080, Low-Delay).

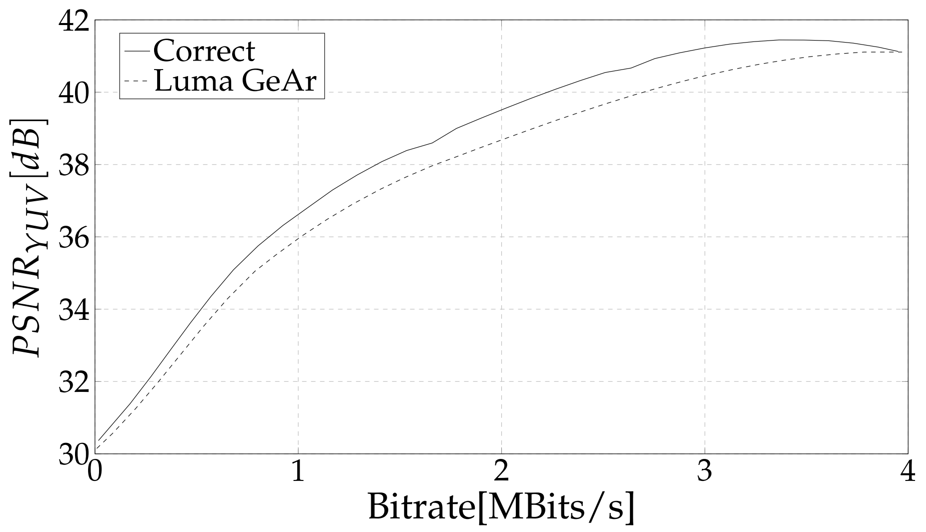

Figure 27. PSNR degradation with GeAr (BasketballDrill [20], 832 × 480, Random Access).

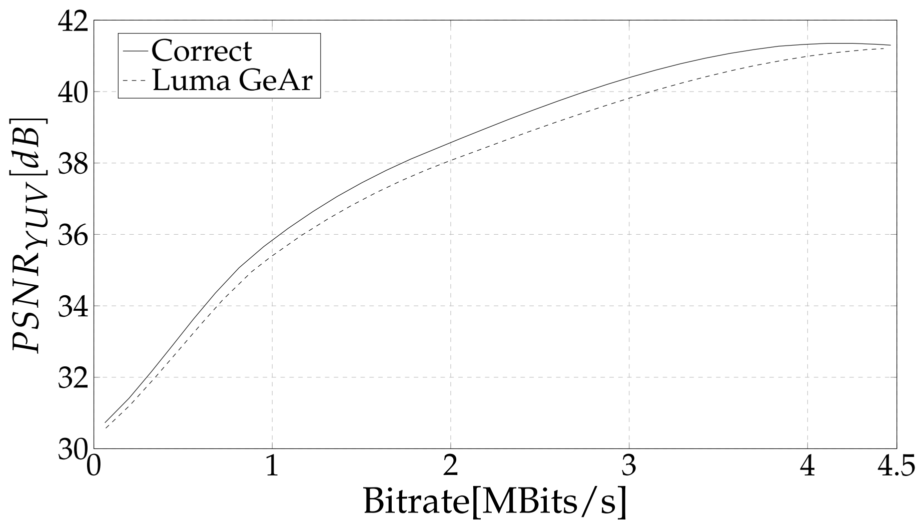

Figure 28. PSNR degradation with GeAr (BasketballDrill [20], 832 × 480, Low-Delay).

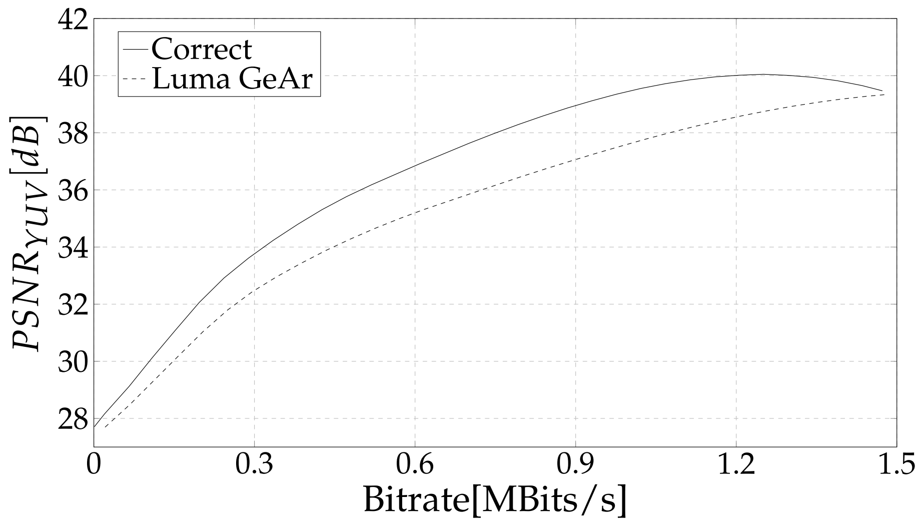

Figure 29. PSNR degradation with GeAr (RaceHorses, [20], 416 × 240, Random Access).

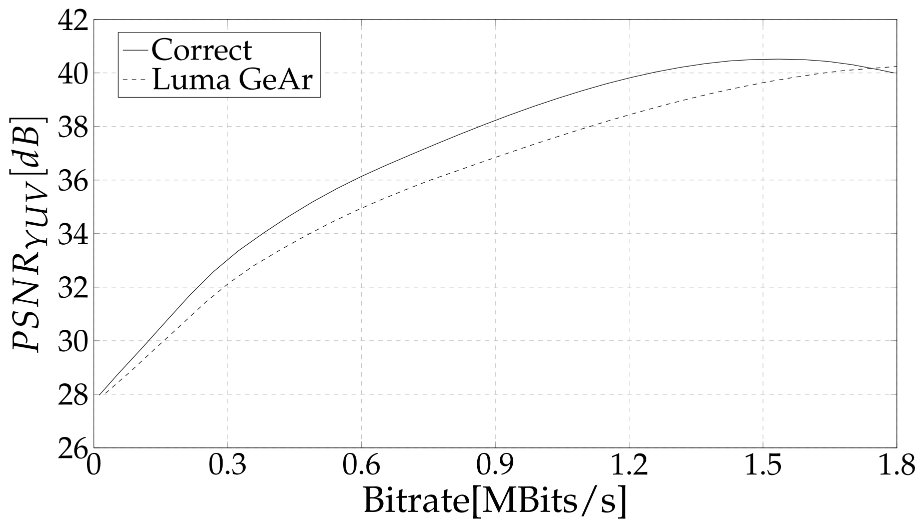

Figure 30. PSNR degradation with GeAr (RaceHorses [20], 416 × 240, Low-Delay).

4. Results and Discussion

The described architecture was modeled in VHDL: The Power estimation has been performed with Synospsys®, while the place and route has been sythetized using Cadence® Innovus, with the UMC 65 nm standard cell technology [23], at 1.2 V, typical process (TT) for the lowest clock frequency achievable by Chroma and Luma filters, which is $f_{\max} = 435$ MHz.

Figures 31 and 32 report the throughput results in numbers of pixels per cycle given the prediction block dimensions used by the HEVC standard for both legacy and approximate filters. From these Tables it is possible to notice that, given the prediction block dimensions, reducing the order filter increases the throughput (up to +79% for the Luma architecture and up to +61.1% for the Chroma one) and lowers the energy consumptions. Moreover, the best gains in throughput are achieved with H equal to 4, for the Luma case, and equal to 2, for the Chroma one, which are respectively the lowest H values possible, while changing the value of W does not show any advantage. These cases are the ones that have the lowest number of pixels to process and that can simultaneously take the most advantage by the proposed alternative scheduling. From Figures 31 and 32 we can also compute the needed Processing Elements to perform the standard HEVC algorithm: To process UHD resolution video sequences at 60 fps and with 4:2:2 Chroma subsampling, the interpolator has to provide 500 and 250 Mpixels/s respectively for Luma and for chroma. We support a throughput between 0.395 and 0.907 pixels per cycle for the Luma Legacy and between 0.432 and 0.918. Clocked at 435 MHz, for the worst $pel/cycle$, three Luma and two Chroma in parallel meet the required throughput constraints while only two Luma and one Chroma in parallel are needed for the best $pel/cycle$.

Table 5 presents the proposed Luma architectures and the best state-of-the-art implementations [6,11,13], including FPGA implementations that cannot be compared to our design. Firstly, it is possible to notice how, thanks to the proposed alternative scheduling, the presented architecture can achieve much higher frequency, up to 64 MHz more, than the architecture [11]. Secondly, Table 5 allows us to assess the impact of the H.C. adder and of the GeAr adder on the perfomance of the Luma Processing Element: The employment of Han–Carlson adders is responsible for a reduction in power (−4.43% for the 3-tap Luma case) and occupied area (−2.36%) while the GeAr shows a performance improvement of 3.45% with drawbacks in terms of area overhead (+7.86%) and power consumption (+6.42%). Thirdly, it clearly indicates that the Power Consumption reduction is mainly due to the N_{tap} reduction rather than due to the adder choice and not on the Adders' side. Thus, it is possible to observe that the choice of the H.C. and GeAr adders respectively slightly reduces the $f_{max} \cdot pel_{max}/A$ ratio. Therefore, for the Luma case, having the possibility to choice the adder allows us to model the Processing Element according to our needs, but always reducing the ratio between throughput and Area.

Table 5. Luma legacy filter synthesis results with optimized adder and N_{tap} architectures.

	N_{tap}	P [mW]	f_{max} [MHz]	Technology	A [µm²]	$\frac{f_{max} \cdot pel_{max}}{A}$ $[\frac{pel}{s \cdot \mu m^2}]$
Luma Legacy [13]	8	11	213	Artix-7 28 nm FPGA	-	-
Luma Approximated [13]	8	12	200	Artix-7 28 nm FPGA	-	-
	7	11	200	Artix-7 28 nm FPGA	-	-
	5	10	200	Artix-7 28 nm FPGA	-	-
	3	10	200	Artix-7 28 nm FPGA	-	-
Luma Legacy [6]	8	-	76.49	Intel 60 nm FPGA	-	-
Luma Legacy [11]	8	-	384	65 nm	-	-
Luma Legacy	8	9.95 (+0%)	435	65 nm	60.28	6.54×10^6
Luma Legacy GeAr	8	10.589 (+6.42%)	450	65 nm	65.04	6.25×10^6
Luma 5-tap	5	9.062 (–8.92%)	438	65 nm	66.89	6.18×10^6
Luma 5-tap H.C.	5	9.131 (–8.23%)	427	65 nm	65.31	6.05×10^6
Luma 3-tap	3	7.384 (–25.8%)	438	65 nm	66.89	6.35×10^6
Luma 3-tap H.C	3	7.057 (–29.1%)	427	65 nm	65.31	6.20×10^6

		N_{tap}			
W	H	8	7	5	3
8	8	0.566	0.604	0.696	0.821
16	16	0.709	0.740	0.810	0.895
32	32	0.825	0.846	0.892	0.943
64	64	0.903	0.916	0.942	0.970
8	4	0.395	0.432	0.533	0.696
4	8	0.604	0.640	0.727	0.842
16	8	0.549	0.587	0.681	0.810
8	16	0.723	0.753	0.821	0.901
32	16	0.702	0.734	0.805	0.892
16	32	0.830	0.850	0.895	0.945
64	32	0.823	0.844	0.890	0.942
32	64	0.904	0.917	0.943	0.971
16	12	0.646	0.681	0.762	0.865
12	16	0.714	0.744	0.814	0.897
16	4	0.379	0.416	0.516	0.681
4	16	0.753	0.780	0.842	0.914
32	24	0.780	0.805	0.861	0.925
24	32	0.827	0.848	0.893	0.943
32	8	0.541	0.579	0.674	0.805
8	32	0.839	0.859	0.901	0.948
64	48	0.874	0.890	0.924	0.961
48	64	0.903	0.916	0.942	0.970
64	16	0.699	0.730	0.803	0.890
16	64	0.907	0.919	0.945	0.972
AVG + Δ%		0	4.15%	14%	27%
MAX + Δ%		0	9%	16%	79%

Figure 31. Two-dimensional legacy and approximate Luma architecture throughput (pel/cycle).

		N_{tap}	
H	W	4	2
4	4	0.640	0.842
8	8	0.753	0.901
16	16	0.850	0.945
32	32	0.917	0.971
4	2	0.471	0.727
2	4	0.727	0.889
8	4	0.604	0.821
4	8	0.780	0.914
16	8	0.740	0.895
8	16	0.859	0.948
32	16	0.846	0.943
16	32	0.919	0.972
8	6	0.696	0.873
6	8	0.762	0.906
8	2	0.432	0.696
2	8	0.842	0.941
16	12	0.810	0.928
12	16	0.853	0.946
16	4	0.587	0.810
4	16	0.877	0.955
32	24	0.892	0.961
24	32	0.918	0.971
32	8	0.734	0.892
8	32	0.924	0.973
AVG + Δ%		0 %	19.8%
MAX + Δ%		0 %	61.1%

Figure 32. Two-dimensional legacy and approximate Chroma architecture throughput (pel/cycle).

Table 6 shows that the higher speed of the approximate solution can be exploited to reduce the energy consumption: For instance, considering the Luma 64×64 case, a -16.5% and a -35.9% of energy reduction is obtained using the 5-tap or the 3-tap filter respectively.

Table 6. Maximum and minum energy per operation for the approximate Luma architecture (nJ/op).

N_{tap}	8	5	3
$(E/\mathrm{op})_{\max}$	88.85	47.66	24.71
$(E/\mathrm{op})_{\min}$	20.84	17.40	13.36

It is possible to extend the same considerations on the Chroma Processing Elements (Tables 7 and 8). As for the Luma case, the Chroma architecture takes advantage of the increase in performance granted by the reduction of N_{tap}: Given a 32×32 block, a -34.9% of energy reduction is achieved when considering the 2-tap filter instead of the legacy 4-tap one (Table 8). Most importantly, Table 7 shows that, differently from he H.C. and GeAr adders in the Luma case, the Adder presented in [16] shows for the Chroma case a huge improvement in terms of area reduction (-28.37%) and $f_{max} \cdot pel_{max}/A$ ratio ($+40,6\%$) at the minor cost of a performance reduction (-4.39%) and a power consumption increase (+1.58%).

Table 7. Chroma legacy filter synthesis results with optimized adder and N_{tap} architectures.

	N_{tap}	P [mW]	f_{max} [MHz]	Technology	A [µm^2]	$\frac{f_{max} \cdot pel_{max}}{A}$ [$\frac{pel}{s \cdot \mu m^2}$]
Chroma Legacy [13]	4	9	217	Artix-7 28 nm FPGA	-	-
Chroma Approximated [13]	4	9	200	Artix-7 28 nm FPGA	-	-
	3	8	200	Artix-7 28 nm FPGA	-	-
	2	6	200	Artix-7 28 nm FPGA	-	-
Chroma Legacy	4	2.966 (+0%)	501	65 nm	21.99	21.04×10^6
Chroma Legacy Adder [16]	4	3.013 (+1.58%)	479	65 nm	15.75	29.58×10^6
Chroma 2-tap	2	2.157 (–27.3%)	-	65 nm	-	-

Table 8. Maximum and minum energy per operation for the approximate Chroma architecture [nJ/op].

N_{tap}	4	2
$(E/\mathrm{op})_{\max}$	23.79	8.25
$(E/\mathrm{op})_{\min}$	6.01	3.91

5. Conclusions

This paper presented an hardware architecture able to perform the fractional-sample filtering required by both the HEVC encoder and decoder. Section 3.1 introduced a set of approximated filters for both Luma and Chroma components. The optimized multiplier-less two-dimensional filter architecture has been described in Section 3.2 featuring hardware reconfiguration, throughput adaptation, on-chip storage and clock gating, guaranteeing a tunable interpolation system able to offer a trade-off in energy saving versus visual quality. Furthermore, the paper introduces a number of architecture-level optimizations that allow to reach a speed enhancement in both Luma and Chroma proposed structures and characterizes the impact of different adders in terms of area, throughput and power. The implemented architectures are fully standard compliant, addressing the 1D and 2D interpolation processes of all the different Luma and Chroma prediction unit sizes adopted by HEVC.

Author Contributions: Investigation, S.P. and A.G.; resources, M.M. and G.M.; writing–original draft preparation, S.P. and A.G.; writing–review and editing, L.V.; supervision, M.M. and G.M.; project administration, M.M. and G.M. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HEVC	High Efficient Video Coding
AVC	Advanced Video Coding
RD	Rate Distorsion
RCA	Ripple-Carry Adder
PPAs	Parallel Prefix Adders
H.C.	Han–Carlson
L.F.	Ladner-Fischer
EDC	Error Detection and Correction
SAM	Standard Approximate Module
ED	Error Detection
CAM	Complementary Approximate Module
GeAr	Generic Accuracy
CGeAr	Complementary GeAr

References

1. Ohm, J.R.; Sullivan, G.J.; Schwarz, H.; Tan, T.K.; Wiegand, T. Comparison of the Coding Efficiency of Video Coding Standards-Including High Efficiency Video Coding (HEVC). *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1669–1684. [CrossRef]
2. Sayood, K. *Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2005; pp. 571–614.
3. Sullivan, G.J.; Ohm, J.R.; Han, W.J.; Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1649–1668. [CrossRef]
4. Aiyar, M.L.; Kenchappa, R. A high-performance and high-precision sub-pixel motion estimator-interpolator for real-time HDTV(8K) in MPEGH/HEVC coding. In Proceedings of the 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), Pudukkottai, India, 24–26 February 2016; pp. 1–8.
5. Tikekar, M.; Huang, C.; Juvekar, C.; Sze, V.; Chandrakasan, A.P. A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications. *IEEE J. Solid-State Circuits* **2014**, *49*, 61–72. [CrossRef]
6. Da Silva, R.; Siqueira, I.; Grellert, M. Approximate Interpolation Filters for the Fractional Motion Estimation in HEVC Encoders and their VLSI Design. In Proceedings of the 2019 32nd Symposium on Integrated Circuits and Systems Design (SBCCI), Sao Paulo, Brazil, 26–30 August 2019; pp. 1–6.
7. Guo, Z.; Zhou, D.; Goto, S. An optimized MC interpolation architecture for HEVC. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 1117–1120. [CrossRef]
8. Afonso, V.; Maich, H.; Agostini, L.; Franco, D. Low cost and high throughput FME interpolation for the HEVC emerging video coding standard. In Proceedings of the IEEE Latin America Symposium on Circuits and Systems, Cusco, Peru, 27 February–1 March 2013; pp. 1–4.
9. Kalali, E.; Adibelli, Y.; Hamzaoglu, I. A Reconfigurable HEVC sub-pixel interpolation hardware. In Proceedings of the 2013 IEEE Third International Conference on Consumer Electronics, Berlin (ICCE-Berlin), Berlin, Germany, 9–11 September 2013; pp. 125–128. [CrossRef]
10. Diniz, C.M.; Shafique, M.; Bampi, S.; Henkel, J. A Reconfigurable Hardware Architecture for Fractional Pixel Interpolation in High Efficiency Video Coding. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 238–251. [CrossRef]
11. Diefy, A.; Shalaby, A.; Sayed, M.S. Low cost Luma interpolation filter for motion compensation in HEVC. In Proceedings of the 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), Abu Dhabi, UAE, 16–19 October 2016; pp. 1–4. [CrossRef]
12. Ghani, A.; Kalali, E.; Hamzaoglu, I. FPGA implementations of HEVC sub-pixel interpolation using high-level synthesis. In Proceedings of the IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era, Istanbul, Turkey, 12–14 April 2016; pp. 1–4.

13. Sau, C.; Palumbo, F.; Pelcat, M.; Heulot, J.; Nogues, E.; Menard, D.; Meloni, P.; Raffo, L. Challenging the Best HEVC Fractional Pixel FPGA Interpolators With Reconfigurable and Multifrequency Approximate Computing. *IEEE Embed. Syst. Lett.* **2017**, *9*, 65–68. [CrossRef]
14. Bossen, F.; Bross, B.; Suhring, K.; Flynn, D. HEVC Complexity and Implementation Analysis. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1685–1696. [CrossRef]
15. Nogues, E.; Menard, D.; Pelcat, M. Algorithmic-level Approximate Computing Applied to Energy Efficient HEVC Decoding. *IEEE Trans. Emerg. Top. Comput.* **2016**, 1–12. [CrossRef]
16. Esposito, D.; Caro, D.D.; Strollo, A.G.M. Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands. *IEEE Trans. Circuits Syst.* **2016**, *63*, 1200–1209. [CrossRef]
17. Mazahir, S.; Hasan, O.; Shafique, M. Adaptive Approximate Computing in Arithmetic Datapaths. *IEEE Des. Test* **2017**, *35*, 65–74.
18. ITU-T Video Coding Experts Group; ISO/IEC Moving Picture Experts Group. HM16.15. Available online: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.15/ (accessed on 20 June 2020).
19. Ugur, K.; Alshin, A.; Alshina, E.; Bossen, F.; Han, W.J.; Park, J.H.; Lainema, J. Motion Compensated Prediction and Interpolation Filter Design in H.265/HEVC. *IEEE J. Sel. Top. Signal Process.* **2013**, *7*, 946–956. [CrossRef]
20. Bossen, F. Common test conditions and software reference configurations. In Proceedings of the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 Wp 3 and ISO/IEC JTC 1/SC 29/WG 11, 12th Meeting, Geneva, Switzerland, 14–23 January 2013.
21. Macedo, M.; Soares, L.; Silveira, B.; Diniz, C.M.; da Costa, E.A.C. Exploring the Use of Parallel Prefix Adder Topologies into Approximate Adder Circuits. In Proceedings of the IEEE Transactions on Circuits and Systems, Batumi, Georgia, 5–8 December 2017; pp. 298–301.
22. Shafique, M.; Ahmad, W.; Hafiz, R.; Henkel, J. A low latency generic accuracy configurable adder. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015; p. 86.
23. (UMC), U.M.C. 65 Nanometer. Available online: http://www.umc.com/english/pdf/UMC%2065nm.pdf (accessed on September 2018). Now Available online: https://www.umc.com/en/Product/process_technologies/Detail/55_65_90nm (accessed on 10 June 2020).