Journal of Real-Time Image Processing
https://doi.org/10.1007/s11554-020-01014-6

ORIGINAL RESEARCH PAPER

Fast and energy-efficient approximate motion estimation architecture for real-time 4 K UHD processing

Roger Porto[1,2] · Murilo Perleberg[1] · Vladimir Afonso[1,2] · Bruno Zatt[1] · Nuno Roma[3,4] · Luciano Agostini[1] · Marcelo Porto[1]

Received: 22 March 2020 / Accepted: 27 August 2020

Abstract
Approximate computing techniques exploit the characteristics of error-tolerant applications either to provide faster implementations of their computational structures or to achieve substantial improvements in terms of energy efficiency. In video encoding, the motion estimation (ME) stage, including the Integer ME (IME) and the Fractional ME (FME) steps, is the most computational intensive task and it is highly resilient to controlled losses of accuracy. In accordance, this article proposes the exploitation of approximate computing techniques to implement energy efficient dedicated hardware structures targeting the motion estimation stage of current video encoders. The designed ME architecture supports IME and FME and is able to real-time process 4 K UHD videos (3840×2160 pixels) at 30 frames per second, while dissipating 108.92 mW. When running at its maximum operation frequency, the architecture can process 8 K UHD videos (7680×4320 pixels) at 120 frames per second. The solution described in this article presents the highest throughput and the highest energy efficiency among all state-of-the-art compared works, showing that the use of approximate computing is a promising solution when implementing video encoders in dedicated hardware.

Keywords Approximate computing · Approximate adders · SAD · Motion estimation · Video coding · HEVC · Energy-efficiency

1 Introduction

With the advent of high spatial and temporal resolution video streaming, possibly supporting 3D contents and representations, video coding becomes a key technology in the multimedia applications domain. On the other hand, there is nowadays an important academic and industrial effort to increase video coding efficiency (bitrate versus image quality). The state-of-the-art High Efficiency Video Coding (HEVC) standard [1], the Chinese Audio Video Standard 2 (AVS2) [2], the recently released AV1 encoder from Alliance for Open Media (AOM), [3] and the Versatile Video Coding (VVC) [4], which will be the next generation of ISO and IEC standard, are some examples of this effort.

On the other hand, multimedia applications are increasingly migrating to battery-powered devices, such as smartphones, digital cameras, virtual reality appliances, and others. In this scenario, not only the video coding efficiency must be highly considered, but also the encoding throughput, the power dissipation and the energy consumption must be regarded as fundamental key aspects. In fact, considering the very high computational effort that is already required by current video encoders, dedicated hardware designs have been extensively used to allow energy-efficient real-time processing of video contents and this is the focus of this work.

✉ Roger Porto
recporto@inf.ufpel.edu.br

1 Video Technology Research Group (ViTech), Group of Architectures and Integrated Circuits (GACI), Graduate Program in Computing (PPGC), Federal University of Pelotas (UFPel), Pelotas, Brazil

2 Sul-Rio-Grandense Federal Institute of Science and Technology (IFSul), Pelotas, Brazil

3 Instituto Superior Técnico (IST), Universidade de Lisboa (ULisboa), Lisboa, Portugal

4 Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa (INESC-ID), Lisboa, Portugal

Published online: 10 September 2020

To overcome this demand, approximate computing has been regarded as a highly promising approach to achieve substantially improved energy efficiency [5, 6]. It exploits the characteristics of error-tolerant applications, i.e., applications that are resilient to a minor loss of accuracy or to numerically imprecise partial results. In the particular domain of video encoding, the introduction of a limited amount of numerical imprecision in the implementation of several video coding algorithms often results in almost imperceptible visual artifacts [7], mostly because of the error-tolerant characteristics of the adopted encoding tools.

Motion Estimation (ME) stands out in this context, being one of the most complex and energy demanding operations in a video encoder [8]. ME consists in a best match search of each block of the current frame inside one or more previously processed reference frames. The current frame block is denoted as current block and the blocks evaluated in the reference frames are called candidate blocks. The method that defines how the search is done is denoted as search algorithm. To reduce the complexity, the search is only implemented in a circumscribed region of the reference frames (search area). The used criterion to define the best match among all the considered candidates is denoted as distortion metric and the Sum of Absolute Differences (SAD) is the most used one, especially when dedicated hardware is considered.

The current video coding standards support multiple block sizes and the ME step must be applied for each one of these block sizes. The Full Search (FS) is the optimal search algorithm since all candidate blocks inside the search area are evaluated. The FS requires a prohibitive computational effort and many fast search algorithms [9, 10] were proposed in the literature intending to surpass this FS limitation reaching almost optimal results. Finally, current video encoders also allow the use of a fractional motion estimation (FME) to increase the ME efficiency, since the ME over integer positions (IME) is often not able to capture the best matching between two frames (integer positions). These fractional positions are obtained by interpolating neighboring integer positions [1].

In accordance, the ME complexity is a function of: (1) the used search algorithm, (2) the number of supported reference frames, (3) the number supported block sizes, (4) the support of FME, (5) the size of the search area, and (6) the used distortion metric. As such, ME can be implemented using several different configurations, varying the block sizes, using different formats, or various search algorithms. This enables the exploration of solutions that are non-optimal in terms of coding efficiency but leads to good coding efficiency results and less energy consumption, which is essential for many current applications, notably those supported by battery powered devices. Moreover, the arithmetic operators used in the SAD calculation are also resilient to simplifications. As an example, the carry chain can be easily shortened or truncated, resulting in an operator that is faster and dissipates less energy. Thus, since ME is basically a search procedure for the block of the previously processed frames that is the most similar to the block under consideration, and the choice of a non-optimal matching does not cause any inconsistency in the encoder process [8].

In current literature, there are several architectural solutions for ME that present various simplifications to reduce complexity and allow an efficient hardware design. In [11], an efficient ME design with a joint algorithm and architecture optimization is proposed. A predictive integer ME algorithm selects the most probable search directions and steps through statistical analysis to reduce the number of search points. An optimized fractional ME algorithm is also presented. The work presented in [12] is a VLSI implementation of a FME design in High Efficiency Video Coding for ultrahigh-definition video applications. It proposes a bilinear quarter pixel approximation, together with an appropriate search pattern, to reduce the complexity of the interpolation and of the fractional search procedure [12]. A highly parallel motion estimation architecture for HEVC encoder is presented in [13]. It has 16 processing units operating in parallel to calculate the SAD values of all possible block sizes. Under a different perspective, the ME presented in [14] uses a modified reference data access skip technique to reduce the memory bandwidth in a ME. In [15], a low-power HEVC ME algorithm and architecture for consumer applications are presented. The proposed algorithm and architecture use subsampling, data reuse, pixel truncation, and adaptive search range techniques to reduce the computational power [15]. Finally, in [16], a low-power motion estimation VLSI architecture is proposed based on a novel method of calculating the SAD, by reusing calculations. However, none of these works are focused on reducing the energy consumption, which is the main concern of current VLSI design, especially for mobile application. Moreover, none of these works exploit the use of an approximate operator to improve the energy efficiency of the arithmetic calculation, by exploring the error resilience of the motion estimation process.

In this context, this article proposes a novel energy-efficient motion estimation architecture based on the usage of approximate Lower-Part-OR Adders (LOA) [17], together with an algorithm level approximation that reduces the number of supported block sizes. The designed architecture is able to process 4 K UHD video in real-time at 30fps. The evaluations in terms of encoding efficiency of the approximate computing techniques used in this work were conducted using the reference software of the HEVC encoder. The designed architecture is fully compliant with the HEVC standard, but it can be easily adapted to be also compliant with other current video encoders, like AV1, AVS2, and

VVC. This solution was based on two previous works of our group [18, 19].

The main contributions of this work are described as follows:

- A set of algorithm simplifications in the ME module, by reducing the amount of supported block sizes, based on experimental results considering the coding efficiency and hardware characterization.
- The usage of different approximate adders to implement the ME SAD tree, with a multi-variable analysis including coding efficiency, area, power, and delay.
- An energy-efficient ME architecture supporting integer and fractional motion estimation which employs the two previous contributions.

All these contributions are supported on several analyses that will be presented in the following sections of this article.

2 Approximate computing approaches for motion estimation

There are many possibilities to apply approximate computing techniques to reduce the energy consumption [1, 5]. In the case of motion estimation, it is possible to apply approximate computing both at the algorithm level and at the arithmetic operator level, aiming the energy-efficient hardware design.

2.1 Algorithm level

Algorithm level simplifications have been the most commonly used approaches to alleviate the ME computational demands. In particular, there are several simplification opportunities to reduce the energy consumption. One possible approach is to consider fast search algorithms [9, 10]. This type of search procedures performs fewer comparisons than the FS algorithm. Nevertheless, it still leads to good results.

Another alternative to obtain energy reduction is to use some subsampling technique like pixel subsampling or block subsampling [20, 21]. As in the previous case, the reduction in energy consumption is obtained by reducing the number of samples that are used in the calculations.

Also on algorithm-level approximations, it is possible to perform simplifications in the evaluated block sizes. Video coding standards specify that a large variety of block sizes may be evaluated at ME. For example, the HEVC defines 24 different block sizes to be evaluated, considering symmetrical and asymmetrical block sizes [1]. It is possible to simplify this task by performing motion estimation only for some block sizes, maintaining the compliance with the standard.

These three opportunities to apply approximate computing at algorithmic level can significantly reduce the required computational effort and the resulting energy consumption, but at a cost of some eventual degradations in the encoding efficiency. As such, these solutions must be carefully applied by considering this tradeoff and compromise.

2.2 Approximate adders

Arithmetic operators significantly influence the overall performance of a digital system [22] and they are not only mainly responsible for the delay but are also the main cause of energy consumption in combinational circuits, mostly due to carry propagation chains [23]. Approximate computing approaches are usually used to mitigate this problem, by shortening or truncating the carry chain. The resulting operator becomes faster, and its energy consumption gets reduced, at the cost of the introduction of a certain level of imprecision in the results.

Several approximate adders have been proposed in the literature. Some examples are the Accuracy-Configurable Adder [24], Carry Cut-Back Adder [25], Error-Tolerant Adder [26, 27], Generic Accuracy Configurable Adder [28], Lower-Part-OR Adder [17], among others.

The Accuracy-Configurable Adder (ACA-II) [24] and the Carry Cut-Back Adder (CCB) [25] are segment-based approximate adders. While the first employs three overlapping sub-adders to reduce the carry chain, the CCB cuts the carry propagation chain at lower-significance positions, using the carry propagation only at the high-significance stages.

Error-Tolerant Adders (ETA) are proposed in [26, 27]. The first one is the ETA-I [26], classified as an approximate full adder. This operator splits the addition into two, by only applying approximate computing techniques in the lower part, where all bits are checked from left to right. The ETA-IV [27] is also a segment-based approximate adder with non-overlapping sub-adders. Specialized units generate the carries from one stage to another.

In [28], a Generic Accuracy Configurable Adder (GeAr) is proposed. The GeAr adder can be configured by specifying the number of sub-adder units and, for each sub-adder, the number of prediction bits, the number of sum bits, and the bit width. Once again, overlapped areas are used.

The Lower-Part-OR Adder (LOA) [17] splits the addition into two smaller parts. The upper-part performs a regular and precise addition with the most significant bits. On the other hand, the computation of the least significant bits is simplified, and the carry chain propagation is eliminated by applying bitwise OR to the inputs. An extra AND gate is used in the most significant bits of the imprecise part to

generate a carry-in for the upper-part, in order to decrease the imprecision [17].

3 Experimental evaluation

As it was previously mentioned, several modifications can be done in the encoding tools to reduce their computation complexity and energy consumption. These modifications are mandatory in battery-powered devices, such as smartphones. However, any encoding tool modification may affect the coding efficiency and the resulting image quality. As such, the whole set of modifications proposed in this article shall be carefully evaluated. This evaluation is presented in this section. First, the ME constraints are evaluated in terms of the number of supported block sizes. Then, the performed evaluations to define the approximate operator that will be applied in the SAD computation module are presented.

The evaluations were performed with the HM 16.15 HEVC reference software regarding the Common Test Conditions [29], which recommends a set of 24 video sequences for tests, where each sequence must be encoded using the four Quantization Parameters (QP): 22, 27, 32, and 37. Most of the video sequences have 8-bit samples, but in the case of 10-bit source videos, each sample is converted to an 8-bit value before encoding [29]. This is automatically done by the reference software, which contains routines to handle this conversion. On the whole, 60 frames of each sequence were used in this evaluation, which means that than 2.1 billion of luminance samples were considered. Only luminance samples were considered because the motion estimation is applied only over luminance samples. Furthermore, all evaluations were performed using the Main Profile (low-complexity profile) and considering the Random Access (RA) temporal configuration. The reason we chose RA instead of another temporal configuration is because it is a more realistic configuration, especially for video streaming. Finally, the coding efficiency was evaluated by considering the BD-rate, which is a metric that measures the percentage variation in the bit rate for the same objective image quality [30]. The ASIC synthesis results were reached using the Cadence Encounter RTL Compiler tool targeting TSMC 40 nm standard-cells [31] technology.

3.1 Algorithmic approximation evaluation

The conducted evaluation in terms of algorithmic approximations targeted the future hardware design. This evaluation was firstly presented in [19], a previous work of this research team.

The presented results considered only the four HEVC square-shaped block sizes, since they are the most frequently used and are the most representative sizes considering the average values used for these video sequences. The evaluations showed that these four sizes together are used to encode, on average, 50.06% of the pixels or, in other words, these four sizes are more used than the other 20 available sizes together.

The HM 16.15 allows the use of two search algorithms: Full Search (FS) and Test Zone Search (TZS). Considering the prohibitive computational effort required by FS, the TZS was used in this work. Modifications were also introduced in TZS [32] implementation to reduce even more its complexity. TZS applies a four-step search: (1) Motion Vector Prediction, which defines the start point for the next step; (2) Initial Search, where an expansion is done in a diamond or square pattern; (3) Raster Search, where a sub-sampled full search is applied over the best result of previous step, and (4) Refinement, where a new diamond or square search is done around the best result of the previous step. In this work, only the Zero predictor was maintained in the first step, and the Raster step was deactivated. Therefore, the search area cannot be moved to a distant region of the collocated block position inside the reference frame. To ensure an efficient memory usage, the bidirectional prediction was also deactivated, and the number of reference frames was limited to one frame only. Moreover, the SAD distortion criterion was adopted for both IME and FME steps. Besides the SAD, the Sum of Absolute Transformed Differences (SATD) distortion metric is also supported in TZS. The HEVC reference software default configuration for TZS uses SAD in IME and SATD in FME. Since our focus is a fast and energy-efficient design for motion estimation, the SAD was used in both: IME and FME, and this choice is consistent with the majority of related works. These modifications in the TZS were essential to allow the design of a high throughput and low-power architecture for the complete ME.

All constraints were evaluated by varying different operation points, where each operation point is characterized by a different combination of the four square-shaped block sizes (single block sizes were not evaluated). Therefore, these combinations result in 11 distinct operation points.

The first evaluation considered the coding efficiency and the energy consumption of these 11 operation points, in order to identify which points are the most suitable to be implemented in a fully optimized hardware architecture.

A hardware design was developed for each of these 11 operation points to allow the evaluation of the reached hardware results, as presented in [19]. By varying the operation point of the developed hardware, different coding efficiency compromises can be reached, also impacting in energy consumption and in other hardware results. Since this is a multivariable optimization problem, we first applied a Pareto Efficiency Analysis [33] by considering these two objectives (coding efficiency and energy consumption) of each of the 11 operation points.

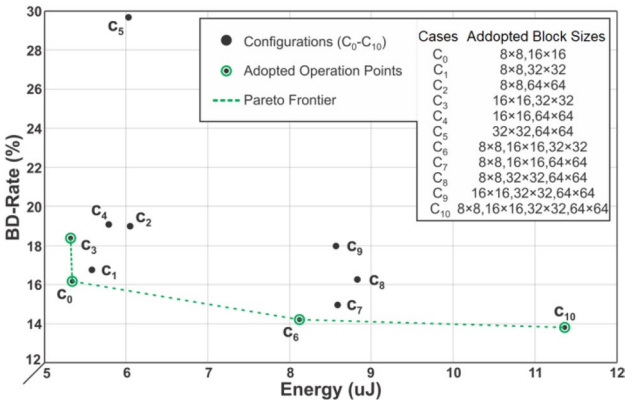


Fig. 1 Pareto Efficiency in the rate-energy space

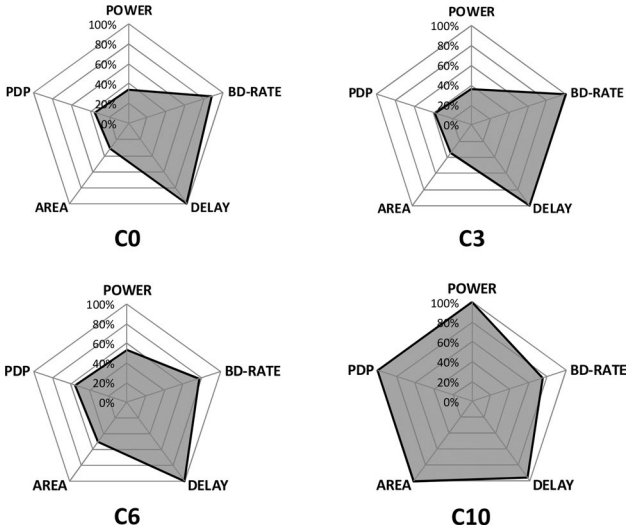


Fig. 2 Multi-variable comparison of Pareto-based optimal operation points

The energy consumption was estimated by considering the processing of one CTU (Coding Tree Unit, basic encoding unit in HEVC [1]). Its evaluation was obtained through the ASIC synthesis of the architecture and by considering the frequency value that is required to process HD 1080p video sequences at 30 frames per second.

When considering this set of 11 operation points, the BD-rate varies from 13.79% (when using all four block sizes) to 29.68% (when only using the 32×32 and 64×64 block sizes). The energy consumption varies from 11.36 μJ (when using all four block sizes) to 5.32 μJ (when only using 16×16 and 32×32 block sizes).

Figure 1 presents the Pareto Frontier of the 11 evaluated operation points. Each of the 11 cases is represented by a black dot, and the legend shows the block sizes adopted in each operation point. The four cases traced by the dotted green line represent the Pareto Frontier, that is, the four optimal operation points, since no other point presents better results in both objectives. As one can observe, the case that uses all block sizes (C_{10}) is the one that reaches the best BD-rate value, but it is also the one that consumes the larger amount of energy. Still, the case that uses only the 16×16 and the 32×32 block sizes (C_3) is the one that reaches the smallest energy consumption, with a considerable impact in BD-rate.

Therefore, the four operation points of the Pareto Frontier (C_0, C_3, C_6, and C_{10}) were used as the starting point of our research. In particular, the four operation points are compared once more, but considering now a multi-variable scenario, including: (1) power dissipation, (2) BD-rate, (3) delay, (4) area, and (5) Power-Delay Product (PDP).

Figure 2 shows the result of this multi-variable comparison in the form of radar charts, where the best solution is the case that presents the smallest gray area. The values in the charts are normalized. As one can observe, the best results that were obtained in almost all evaluated variables are the case C_0, which presents the smallest gray area. Thus, the C_0 operation point was selected to be used in the proposed architecture. This operation point causes an average BD-rate loss of 16.17% and an average BD-PSNR loss of −0.37 dB, considering all evaluated video sequences. This experiment also showed that the operating modes with fewer blocks may lead to a reduction of computational effort without a substantial impact on coding efficiency.

In particular, the choice of the C_0 operation point represents a power of 91.30 mW, an area of 4775 Kgates, a delay of 20.97 ns, a PDP of 1.91 nJ, and a BD-rate of 16.17%. When compared with the C_{10} operation point (which has the largest gray area—see Fig. 2), the operation point C_0 provides a power reduction of 66%, an area reduction of 69%, and a PDP reduction of 65%, with a slight increase in delay (4%) and BD-rate (17%). It is important to note that, since the BD-Rate is a relative metric, these percentages indicate how much the bitrate must be increased in order to maintain the same objective video quality (PSNR). If we consider the loss of quality for the same bitrate, we must observe the effect in BD-PSNR. The BD-PSNR indicates a quality loss of -0.37 dB for the C0 operation point. This means that for the same compression rate, the designed architecture supporting the C0 operation point will have a PSNR 0.37 dB lower than the original PSNR. The BD-PSNR values were obtained from the same evaluation performed to obtain BD-rate values.

3.2 Approximate adders evaluation

A second evaluation task considered the identification of the approximate adder topologies that are the most suitable to be used in the ME architecture for the C_0 operation point presented in the last subsection (supporting 8×8 and 16×16

Table 1 BD-rate results for the considered adder structures

Adder	ACA-II	CCB	ETA-I	ETA-IV	GeAr	LOA
BD-rate (%)	7.5	2.8	2.5	0.2	3.9	0.8

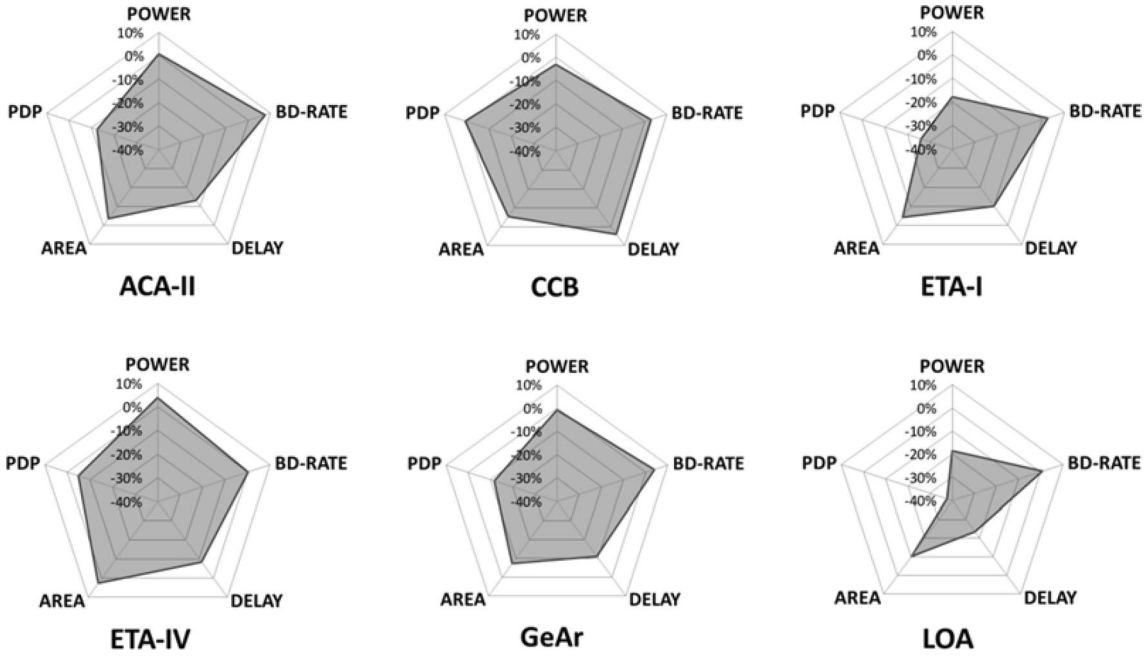


Fig. 3 Multi-variable comparison of approximate adders

block sizes). In this case, the idea is to use the approximate adder in the SAD calculation which is the most intensive operation in ME, intending to provide the highest reduction of the energy consumption and hardware resources. To attain this objective, a set of 8-bit approximate addition functions was described in C++ and evaluated. Each adder was evaluated by using 99,840 luminance samples, extracted from the first frame from a class D-type test video sequence (BasketballPass_416×240_50.yuv). A total of 26 versions of the six previously presented 8-bit operators were evaluated. From these first analyses, the best configurations for these approximate adders were identified: (1) ACA-II with three overlapping sub-adders with four bits each; (2) CCB with two sub-adders with four bits each and one bit at the cut-back definition; (3) ETA-I with three precise and five imprecise bits; (4) ETA-IV with three sub-adders (three, three, and two bits) using two and three bits in the first and second carry generations, respectively; (5) GeAr using two overlapping sub-adders with five bits each; and (6) LOA with three precise and five imprecise bits.

Then, these six approximate adders were embedded in the motion estimation encoding tool of the HM 16.15 HEVC reference software, to evaluate their impact on the coding efficiency (in terms of BD-rate). The approximate operators were only inserted in the first stage of the SAD computation modules, in order to avoid accumulated error effects. This first operation corresponds to the subtraction that is needed to generate the SAD absolute differences. The average results of this evaluation are presented in Table 1 and they were obtained by encoding the 20 test video sequences, using the four QP values recommended in the Common Test Conditions (CTC) [29], and using the Low Delay P Main configuration. According to the values presented in Table 1, the best results were obtained with LOA and ETA-IV.

The next experiment was conducted to obtain a hardware characterization of the approximate adders. The operators were described in VHDL and synthesized to ASIC.

These experimental results were compared with those obtained for the ripple-carry (RCA) precise adders, by considering the increase (or decrease) in terms of (1) power dissipation, (2) delay, (3) area, and (4) Power–Delay Product (PDP). The decision on the RCA adder is justified because it is commonly used as a reference in many works in the literature, such as [17, 23, 26], and [28]. If we use a fast adder, the effects of using approximate adders can be masked in the results. Therefore, we decided to use the RCA adder as the precise operator in our comparisons. The multi-variable comparison of the approximate arithmetic operators is depicted in Fig. 3 in the form of radar charts. The BD-rate results were also included. One can notice that the LOA

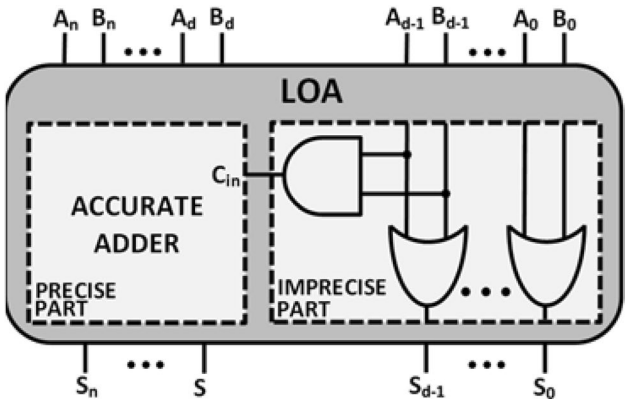


Fig. 4 Lower-part-OR adder structure

operator reached the best performance, since it presented the best result in almost all axes and also the smallest gray area among all evaluated operators. Consequently, the LOA was selected to be used in the energy-efficient ME architecture. The architecture of a LOA operator is presented in Fig. 4.

The choice of LOA results in a consumed power of 105 µW, an area of 65 cells, a delay of 693 ps, a PDP of 72.77 pJ, and a BD-rate of 2.5%. When compared with the CCB adder (which has the largest gray area in Fig. 3), LOA reduces the power in 16%, the area in 5%, the delay in 26%, the PDP in 38%, and the BD-rate in 36%.

4 Energy-efficient approximate HEVC motion estimation architecture

This section presents the developed energy-efficient approximate IME/FME architecture, which implements an approximated solution for the ME encoding tool, according to the constraints and operation points defined in Sect. 3.1. In addition, this architecture also considers the approximated SAD computation, by using the approximate operator defined in Sect. 3.2.

The processing structure is based on the ME architecture proposed in [19]. The same architectural template was used, but the new architecture was simplified since only two block sizes are supported now (C_0 only processes blocks with size 8×8 and 16×16 samples). The original version supports four block sizes: 8×8, 16×16, 32×32, and 64×64 samples. Then, the SAD unities were replaced with the ones using imprecise adders. The architecture adopts the modified TZS algorithm to perform the IME step, while the interpolation filters, defined by the HEVC, were implemented to perform the FME step. This architecture has several dedicated modules to maintain a high throughput, where each module has an instance of the IME and FME units to process one block size.

According to the evaluations presented in Sect. 3, the case that results in the best tradeoff in the multi-variable comparison is the C_0 case, which processes blocks with size 8×8 and 16×16 pixels. In accordance, these two block sizes were adopted in this architecture. Aiming the processing of high-resolution videos, some of these modules were also replicated. This results in four modules to deal with the 8×8 blocks and two modules to deal with the 16×16 blocks. All these modules can work in parallel to encode the video.

The complete ME architecture is represented in Fig. 5, where all TZS and FME modules for the two block sizes adopted in this work are presented. Each module processes a predefined block group. When each FME module finishes its execution to process a candidate block, the global control unit copies the result to a SAD table. This table stores the SAD value and the motion vector related to that SAD and releases the stored values only when all modules finish the CTU processing.

Although memory access is a major concern when designing video encoders in hardware, this problem was

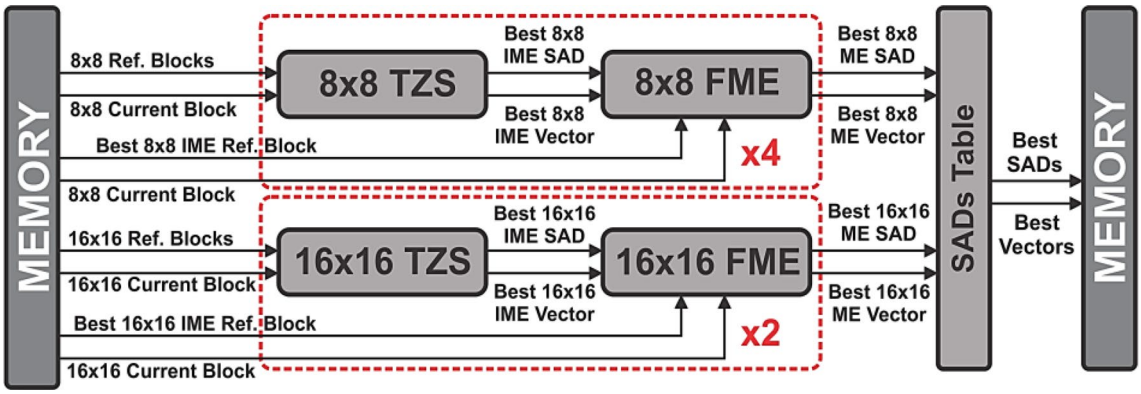


Fig. 5 Complete ME architecture

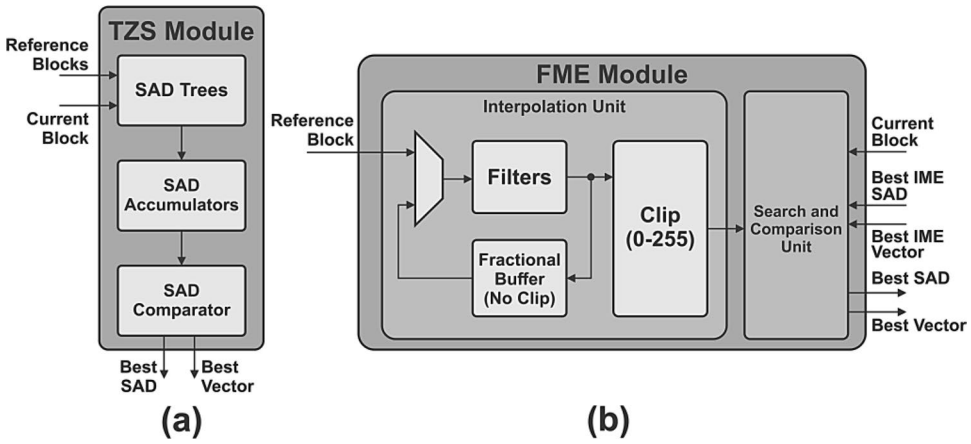


Fig. 6 Architecture modules: **a** TZS module; **b** FME module

not focused in this article. The architecture proposed in this article, when integrated in a complete encoder, can be easily adapted to use an optimized memory system applying the solutions published in the literature, like the use of an internal memory to store the entire search area [34], the use of Level C reuse schemes [35], and the use of an additional memory for the FME interpolated positions [12]. Thus, IME and FME could operate simultaneously with other encoder modules minimizing the number of accesses to the external memory.

In the next subsections, the hardware solution of these two ME steps is explained in detail. Then, Sect. 4.3 explains the approximated SAD computation.

4.1 Modified TZS implementation

The designed hardware unit to perform the IME step is based in the architecture presented in [19] and it was divided into the control and the operative parts. The control part selects the candidate blocks to be compared, according to a modified TZS algorithm. This modified TZS only implements the zero predictor in the prediction step, and the raster step was disabled, which makes the search area to be predefined around the current block, ensuring a regular memory communication. Moreover, all TZS expansions of the first search and refinement steps always receive 16 candidate blocks from memory and the total number of expansions was decreased from 7 (in the original TZS) to 5 (in this modified version). These simplifications are detailed in [19]. The control part also deals with the stop conditions of each TZS step, as it will be better explained below.

The operative part of the IME unit performs the comparison of the 16 candidates selected by the control part in parallel, producing the motion vector corresponding to the best candidate block according to the adopted similarity criterion, in this case, the SAD criterion. The processing is made one line per clock cycle, so this unit is designed to process in parallel one line of all 16 candidate blocks selected by the control unit.

To ensure the real-time processing of UHD videos, the comparison between each candidate block and the current block is made using a specialized Search and Comparison Unit (SCU), to process each of the adopted block sizes. The architecture of each SCU is represented in Fig. 6, along with its three steps. The first step includes the SAD trees, responsible for calculating the SAD value of one line from the current block with one line of his candidate block. The SAD tree architecture, along with the proposed approximate operator, will be better presented in Sect. 4.3.

After the SAD trees processing, the obtained SAD value is accumulated by the SAD accumulators, while the SAD trees process the remaining block lines. The architecture of each SAD accumulator is presented in Fig. 7a. This architecture stores a new value at the start of the processing of a new block and sums the input value with the stored value in the remaining block lines. Hence, when the SAD tree concludes the processing of the last line of the processed block,

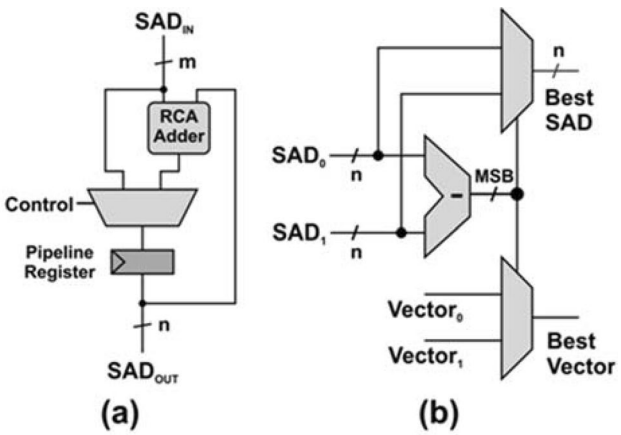


Fig. 7 SCU modules: **a** SAD accumulator; **b** SAD comparator

the value on each SAD accumulator corresponds to the SAD value of the evaluated candidate block.

Finally, at the end of the processing of the last line of the considered block, the SAD comparator step processes the SAD values produced by all accumulators along with their motion vectors. It compares all SAD values, two by two, and when it finishes its processing the comparator outputs the smallest SAD value and the respective motion vector. Due to the introduced pipeline stages, the comparator needs four cycles to process the SAD values from the 16 candidate blocks. Figure 7b presents the SAD comparator of two SAD values.

The control part of the modified TZS is also responsible for the implementation of a stop condition. The original TZS could freely go through all blocks of the SA while the refinement step finds a smaller SAD value in a new iteration. This is a problem when considering real-time processing, since the number of cycles that are needed to obtain the best candidate block will be undefined and depending on the video data. Hence, to ensure real-time throughput, the control part of the IME limits the number of candidate blocks evaluated by each module to 240 candidates. According to previous evaluations presented in [19], 93.33% of the TZS executions use less than 240 candidate blocks.

The 8×8 TZS module requires 148 clock cycles to process the 240 candidates and this module is the bottleneck of the complete architecture, defining the reached throughput, as will be discussed in Sect. 5.

4.2 Fractional motion estimation implementation

The FME unit is fully compliant with the HEVC encoders and it is used to improve the motion estimation by interpolating some new blocks at fractional positions (of ½- and ¼-sample in HEVC) around the best IME result. After the interpolation, one of those new fractional candidates is selected as the best ME candidate. The FME interpolators used in this architecture were defined by the HEVC, but the filters can be adapted to also support other encoders.

The FME process was divided into two units, as presented in Fig. 6b. The first one is the Interpolation Unit, which generates the new candidate blocks at fractional positions. The second one is a SCU, used to evaluate the interpolated candidates and select the one that is more similar to the current block.

The interpolation unit has a multiplexer to select the samples that will be passed to the filters and to interpolate the fractional samples. The samples can be provided either from the best IME candidate block, or from an internal buffer. After that, a set of filters is used to interpolate the new samples at fractional positions around the position of the input block. Then, this architecture generates and evaluates all 48 possible fractional blocks.

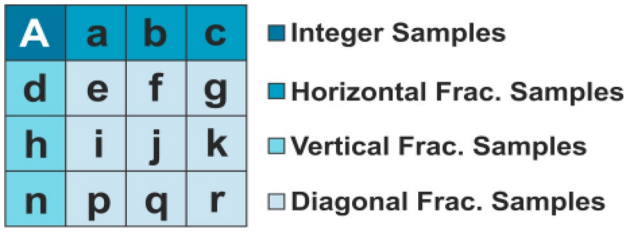


Fig. 8 Fractional samples according to their position with the integer samples

There are three types of filters used to interpolate the new blocks, and their use depends on the position of the fractional sample. Specific details of these filters can be found in [36]. After the filters' implementation, the Fractional Buffer of the interpolation unit is used to store the horizontal fractional samples, since the horizontal samples can be used as input to the interpolation of the diagonal samples. Moreover, a clipping operation is applied at the outputs of the filters to maintain all samples with 8-bit values, which will be processed by the SCU. Therefore, all outputs of the interpolation unit have values between 0 and 255.

Figure 8 presents an integer sample and the fractional samples that can be interpolated around it, according to HEVC. As one can notice, these samples can be categorized into three groups, according to the sample displacement in relation to the integer sample: horizontal, vertical or diagonal. Therefore, the generation of the fractional samples was also divided according to the sample displacement.

The processing starts by generating the horizontal samples, so the filters receive as input one horizontal line of the best IME block at each clock cycle, and they will generate the new samples that will compose six new horizontal fractional blocks. These horizontal blocks are stored in an internal buffer. Later, the vertical samples are generated, so the interpolation unit receives one vertical column of the best IME block at each clock cycle. This processing generates the samples used to compose six new vertical fractional blocks. Finally, the diagonal samples are generated by using the horizontal samples stored in the internal buffer, one column at each clock cycle. The diagonal samples are used to compose 36 new diagonal fractional blocks around the best IME block.

At the end of the processing of each fractional samples category, the generated blocks are sent to the SCU for evaluation. The SCU performs the evaluation in a similar way that was used in the IME computational part. Twelve SAD Trees are used to compute the SAD value of the candidates being evaluated. The SAD of each candidate is stored in one of the 48 SAD Accumulators, while the interpolation unit computes the remaining candidate blocks. When all 48 candidate blocks are generated, the values of the 48 SAD

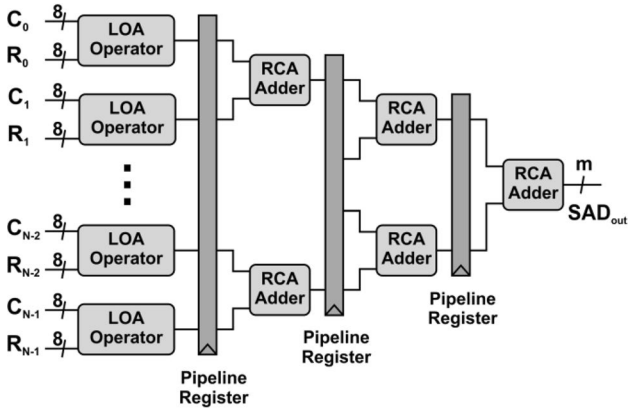


Fig. 9 Block diagram of a SAD tree

Accumulators are passed to the Comparator, which will define which the fractional candidate block has the smallest SAD value and this SAD value is sent to the output together with the respective fractional motion vector.

Since all 48 possible fractional blocks are generated and evaluated by FME unit, the number of clock cycles to complete the FME processing is constant but depends on the block size and on the pipeline stages of the FME architecture.

It is important to note that IME and FME run in parallel, one of which is idle until the other ends. The 8×8 FME requires 63 clock cycles to process each processing unit (PU), while the 16×16 FME requires 104 clock cycles. The IME does not have a specific data throughput rate. It depends on the refinement decisions of TZS. However, the 8×8 IME requires from 31 to 148 clock cycles to process one PU, while the 16×16 IME consumes from 56 to 272 clock cycles per PU.

4.3 Approximate computing SAD architecture

As previously mentioned, the SAD Trees are responsible to compute the SAD value of each line being processed. The SAD Trees of both IME and FME architectures were implemented with approximate adders, as presented in Fig. 9.

Several pipeline stages were used between the adders, in order reach higher throughputs. In the first pipeline stage, the approximated LOA operators were used to obtain the difference between the candidate block (R_n inputs) and the current block (C_n inputs).

In the next pipeline stages, the SAD values of the samples are added two by two. To achieve this, RCA adders were used to accumulate the SAD values of the whole candidate block line, so the result of each SAD Tree is the SAD of each candidate block related to the current block. The choice of RCA operators is justified because we wish to evaluate the throughput impacts with the use of LOA and the use of some fast adder in the other SAD Tree levels could mask these results. The use of RCA also contributes to reduce the hardware consumption since fast adders require much more logic than RCAs.

The LOA operator was only adopted in the first pipeline stage of the SAD Trees, which are 8-bit wise. The reason for this choice, as previously discussed, is to avoid the error accumulation resulting from cascaded LOA operators.

5 Synthesis results and comparisons with related works

The complete ME architecture was synthesized with Cadence Encounter RTL Compiler tool using TSMC 40 nm standard cell library, which is the technology available in our university. The synthesis effort was defined as "strong" and the use of clock gating was enabled.

Concerning the power estimation, the Encounter RTL Compiler tool was set to the default switching activity. After that, the energy efficiency was defined as the ratio between the spent energy (given in Joules) to process a dataset and the number of samples that comprise such dataset. To simplify this evaluation, this dataset corresponds to the number of samples that are processed per second. The energy consumed in one second was obtained using the average power dissipation reported in Watts (Joules per second), by Cadence Encounter RTL Compiler tool.

5.1 Synthesis results

Table 2 summarizes the reached synthesis results. Three different target operating frequencies setups were considered. The first configuration considered a target frequency of 145 MHz. With this operation frequency, the architecture can achieve real-time processing of UHD 4 K (3840×2160) videos at 30 frames per second. This configuration requires 1541 Kgates while dissipating 108.92 mW. The energy efficiency of the developed design at this frequency is 0.44 nJ/sample. The second target frequency was 580 MHz, allowing the processing of UHD 8 K (7680×4320) videos at 30 frames per second. At this frequency the power dissipation is 256.5 mW and the energy efficiency is 0.26 nJ/sample. The third synthesis was performed at the maximum frequency of 2330 MHz. At its maximum frequency, the architecture can process UHD 4 K videos at 480 frames per second or UHD 8 K videos at 120 frames per second and dissipates 585.90 mW, with an energy efficiency of 0.15 nJ/sample.

Considering these three synthesis results, one can observe that the higher the throughput, the higher the energy

Table 2 Comparison with related works

	Jou [11]	He [12]	Medhat [13]	Park [14]	Singh [15]	Jia [16]	This work			
Technology	90 nm	65 nm	65 nm	65 nm	90 nm	65 nm	40 nm			
Supported tools	IME + FME	FME	IME	IME	IME	IME	IME + FME			
Maximum resolution	4096 × 2048 @60fps	7680 × 4320 @30fps	3840 × 2160 @30fps	3840 × 2160 @30fps	3840 × 2160 @30fps	8192 × 4320 @30fps	3840 × 2160 @30fps	7680 × 4320 @30fps	3840 × 2160 @480fps	7680 × 4320 @120fps
Search algorithm	Modified TZS	–	Fast search	TSS	Modified HGS	New fast algorithm	Modified TZS			
T hroughput (Mpixels/s)	503.32	995.33	248.83	248.83	248.83	1061.68	248.83	995.33	3,981.31	
Supported block sizes	64 × 64 to 8 × 8	64 × 64 to 16 × 8	64 × 64 to 4 × 4	32 × 32 to 8 × 4	32 × 32 to 8 × 8	All in HEVC	16 × 16 and 8 × 8			
Area (Kgates)	779	1183	434	617	1441	2310	1541	1544	1660	
Power (mW)	–	198.6	–	–	151.76	362	108.92	256.50	585.90	
Frequency (MHz)	270	188	720	350	250	290	145	580	2320	
BD-rate (%)	5	2.07	–	2.4	–	0.6	16.17			
Energy Efficiency (nJ/sample)	–	0.20	–	–	0.61	0.27	0.44	0.26	0.15	

efficiency. The best energy efficiency result is 0.15nJ/sample which was obtained at the maximum operation frequency.

5.2 Comparison with related works

Table 2 also presents a comparison of the designed architecture with related works. It is not easy to do a complete and fair comparison with related works, since these works considered different technologies and focused in different processing rates. Besides, the related works consider different types of encoding tools, like the support for fractional motion estimation (FME) and integer motion estimation (IME), the number of evaluated CTU sizes, the used search algorithms, and others. Unfortunately, it is not easy to fairly compare solutions that support different encoding tools. The higher the supported CTU sizes or the higher the search algorithm complexity, the greater will be the required level of hardware parallelism or the higher will be the required operating frequency (or both) to achieve real-time processing at a given video resolution, and the higher will also be the consumed energy. The FME use (or not) also contributes to this relation. Then, each published architecture targets some type of application with support of a specific group of tools, hindering a completely fair comparison.

It is important to highlight that, from the works presented in Table 2, only the work presented in [11] and our design are complete solutions for motion estimation considering both IME and FME. However, the work in [11] does not present its results of dissipated power nor its energy consumption characterization [11]. Thus, it is also not possible to determine the energy efficiency to perform further comparisons.

The solution presented in [12] has slightly higher energy efficiency than that presented in our work. However, that work deals only with the FME while the architecture presented in this article has the complete ME solution, including FME and IME and the IME is much more complex than FME. Even supporting the FME only, the solution in [12] can reach a throughput 4 × lower than that reached with our architecture.

The works [13–16] present solutions for IME only. Among these three solutions, only [15, 16] presents power results, making possible to determine its energy efficiency. Compared to [15], our solution is 28% more energy efficient and works at 42% lower frequency when running with the same target of performance (3840 × 2160@30fps). Besides, our architecture can reach a maximum throughput 16 × higher than that reached in [15]. When comparing our architecture with the one presented in [16], the resulting energy efficiency is quite similar, but our solution can reach a maximum throughput almost 4 × higher than that reached in [16]. The works [13, 14] did not present power results, but our architecture is able to reach the same processing rate at an operation frequency almost 5 × lower than [13] and 2.4 × lower than [14], which is a strong indication that our solution reached lower power and higher energy efficiency than these two works. Other important comparison is related with the throughput, since our architecture reached a maximum throughput of 16 × higher than those reached in [13, 14].

Concerning the BD-rate losses, it is important to point out that energy-efficient hardware implementations of video encoders only are possible if some simplifications are considered in the coding tools. In our case, besides the simplifications found in other hardware designs targeting video applications, we have the introduction of imprecision through approximate computing techniques. Thus, quality losses are expected to achieve the desired energy efficiency.

Finally, the architecture presented in this article has the highest maximum throughput among all related works. Moreover, the proposed architecture presents the best result in terms of energy efficiency among the related works, even when compared to those that only implement FME or IME. These results showed that the use of approximate computing and imprecise operators are promising solutions to implement energy-efficient video encoders in dedicated hardware.

6 Conclusions

This article presented an energy-efficient approximate motion estimation architecture supporting both the integer and fractional ME. This architecture is fully compliant with the HEVC standard and it can be easily adapted to be compliant with other current encoders, like AV1, AVS2, and VVC (only adaptions in the FME interpolation filters are required). This architecture explores approximation at algorithmic level, where the supported set of block sizes was limited, and other minor hardware friendly simplifications. The architecture also explores approximation at the arithmetic operator level, by using imprecise LOA adders in the SAD calculations to improve the energy efficiency. The designed ME architecture is able to real-time process 8 K UHD videos (7680 × 4320 pixels) at 120fps when running with its highest frequency.

Several preliminary evaluations and analyses supporting the development of this architecture were presented in this article. The reached results showed a power dissipation of 108.92 mW and an area of 1541 Kgates when focusing in a throughput to process 4 K UHD videos (3840 × 2160 pixels) at 30 frames per second. The area results did not show reductions when compared with related works, as could be expected. This occurs because we used a higher parallelism level to reach the throughput required to process HD videos in real-time.

The architecture presented in this article reached the highest maximum throughput among all related works. Moreover, it offers the best result in terms of energy efficiency, even when compared to those state-of-the-art architectures that only perform FME or IME.

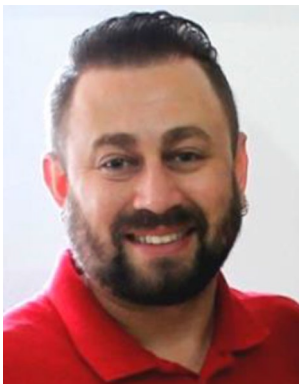
Considering the reached results we can conclude that the use of approximate computing at algorithmic and at arithmetic operator levels can be a powerful tool for dedicated hardware designs targeting energy-efficient high-resolution video encoders.

Acknowledgements This work is partly financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brasil (CAPES) Finance Code 001, by FCT projects PTDC/EEI-HAC/30485/2017 and UIDB/50021/2020, and also by CNPq and FAPERGS Brazilian research support agencies.

References

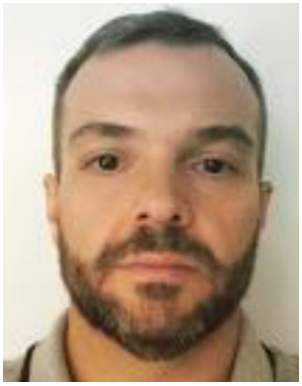
1. Sullivan, G., Ohm, J., Han, W., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. (2012). https://doi.org/10.1109/TCSVT.2012.2221191
2. He, Z., Yu, L., Zheng, X., Ma, S.: AVS2-video coding standard - an application-oriented and high performance video coding standard. IEEE Int. Conf. Multimedia Expo Workshops (2014). https://doi.org/10.1109/ICMEW.2014.6890707
3. AOM: Alliance for Open Media: Get started creating products with AV1. https://aomedia.org/av1-features/get-started/ (2019). Accessed 08 May 2019.
4. MPEG: The Moving Picture Experts Group: Versatile Video Coding. https://mpeg.chiariglione.org/standards/mpeg-i/versatile-video-coding (2019). Accessed 08 May 2019.
5. Han, J., Orshansky, M.: Approximate computing: an emerging paradigm for energy-efficient design. In: 18th IEEE European Test Symposium (2013). https://doi.org/10.1109/ETS.2013.6569370.
6. Chippa, V., Venkataramani, S., Chakradhar, S., Roy, K., Raghunathan, A.: Approximate computing: an integrated hardware approach. Asilomar Conf Signals Syst. Comput. (2013). https://doi.org/10.1109/ACSSC.2013.6810241
7. Raha, A., Jayakumar, H., Raghunathan, V.: A Power efficient video encoder using reconfigurable approximate arithmetic units. In: 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems (2014). https://doi.org/10.1109/VLSID.2014.62.
8. Porto, R., Agostini, L., Zatt, B., Porto, M., Roma, N., Sousa, L.: Energy-efficient motion estimation with approximate arithmetic. In: IEEE 19th International Workshop on Multimedia Signal Processing (2017). https://doi.org/10.1109/MMSP.2017.8122248.
9. Alvar, S., Abdollahzadeh, M., Seyedarabi, H.: A novel fast search motion estimation algorithm in video coding. In: IEEE 23rd International Symposium on Industrial Electronics (2014). https://doi.org/10.1109/ISIE.2014.6864737.
10. Chiang, J., Kuo, W., Su, L.: Fast motion estimation using hexagon-based search pattern in predictive search range. Int. Conf. Comput. Commun. Netw. (2007). https://doi.org/10.1109/ICCCN.2007.4317974
11. Jou, S., Chang, S., Chang, T.: Fast motion estimation algorithm and design for real time QFHD high efficiency video coding. IEEE Trans. Circuits Syst. Video Technol. (2015). https://doi.org/10.1109/TCSVT.2015.2389472
12. He, G., Zhou, D., Li, Y., Chen, Z., Zhang, T., Goto, S.: High-throughput power-efficient VLSI architecture of fractional motion estimation for ultra-HD HEVC video encoding. IEEE Trans. Very Large Scale Integr. Syst. (2015). https://doi.org/10.1109/TVLSI.2014.2386897
13. Medhat, A., Shalaby, A., Sayed, M.: High-throughput hardware implementation for motion estimation in HEVC encoder. In: IEEE 58th International Midwest Symposium on Circuits and Systems (2015). https://doi.org/10.1109/MWSCAS.2015.7282040.
14. Park, S., Choi, B., Lim, I., Park, H., Kang, S.: An efficient motion estimation hardware architecture using Modified Reference Data Access (MRDAS) skip algorithm for high Efficiency Video Coding (HEVC) encoder. In: IEEE 6th International Conference on Consumer Electronics – Berlin (2016). https://doi.org/10.1109/ICCE-Berlin.2016.7684724.
15 Singh, K., Ahamed, S.: Low power motion estimation algorithm and architecture of HEVC/H265 for consumer applications. IEEE Trans. Consum. Elect. (2018). https://doi.org/10.1109/TCE.2018.2867823
16. Jia, L., Tsui, C., Au, O., Jia, K.: A low-power motion estimation architecture for HEVC based on a new sum of absolute difference computation. IEEE Trans. Circuits Syst. Video Technol. (2018). https://doi.org/10.1109/TCSVT.2018.2890204
17. Mahdiani, H., Ahmadi, A., Fakhraie, S., Lucas, C.: Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans. Circuits Syst. I Regul. Pap. (2010). https://doi.org/10.1109/TCSI.2009.2027626
18. Porto, R., Agostini, L., Zatt, B., Roma, N., Porto, M.: Power-efficient approximate SAD architecture with LOA imprecise adders. In: IEEE 10th Latin American Symposium on Circuits & Systems (2019). https://doi.org/10.1109/LASCAS.2019.8667554.
19. Perleberg, M., Afonso, V., Conceição, R., Susin, A., Agostini, L., Porto, M., Zatt, B.: Energy and rate-aware design for HEVC motion estimation based on Pareto efficiency. J. Integrat. Circuits Syst. **13**, 1 (2018)
20. Hwang, S., Ha, J., Sunwoo, M.: Efficient integer motion estimation algorithm using sub-sampling. Int. SoC Design Conf. (2009). https://doi.org/10.1109/SOCDC.2009.5423844
21. Kang, S., Yoo, D., Lee, S., Kim, Y.: Hardware implementation of motion estimation using a sub-sampled block for frame rate up-conversion. Int. SoC Design Conf. (2008). https://doi.org/10.1109/SOCDC.2008.4815694
22. Frustaci, F., Lanuzza, M., Zicari, P., Perri, S., Corsonello, P.: Designing high-speed adders in power-constrained environments. IEEE Trans. Circ. Syst. II Express Briefs (2009). https://doi.org/10.1109/TCSII.2008.2010187
23. Dutt, S., Nandi, S., Trivedi, G.: A comparative survey of approximate adders. In: 26th International Conference Radioelektronika (2016). https://doi.org/10.1109/RADIOELEK.2016.7477392.
24. Kahng, A., Kang, S.: Accuracy-configurable adder for approximate arithmetic designs. Design Autom. Conf. (2012). https://doi.org/10.1145/2228360.2228509
25. Camus, V., Schlachter, J., Enz, C.: A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision. In: 53nd ACM/EDAC/IEEE Design Automation Conference (2016). https://doi.org/10.1145/2897937.2897964.
26 Zhu, N., Goh, W., Zhang, W., Yeo, K., Kong, Z.: Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. IEEE Trans. Very Large Scale Integr. Syst. (2010). https://doi.org/10.1109/TVLSI.2009.2020591
27. Zhu, N., Goh, W., Wang, G., Yeo, K.: Enhanced low-power high-speed adder for error-tolerant application. Int. SoC Design Conf. (2010). https://doi.org/10.1109/SOCDC.2010.5682905

28. Shafique, M., Ahmad, W., Hafiz, R., Henkel, J.: A low latency generic accuracy configurable adder. ACM/EDAC/IEEE Design Autom. Conf. (2015). https://doi.org/10.1145/2744769.2744778
29. Bossen, F.: Common test conditions and software reference configurations. Document JCTVC-L1100, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 Joint Collaborative Team on Video Coding (JCT-VC). (2013).
30. Bjontegaard, G.: Improvements of the BD-PSNR model, VCEG-AI11, ITU-T SG16/Q6 VCEG 35th meeting, Berlin, Germany, 16–18 (2008).
31. Silvaco: Nangate FreePDK45 open cell library. https://www.silvaco.com/products/nangate/FreePDK45_Open_Cell_Library/index.html (2019). Accessed 08 May 2019.
32. High Efficiency Video Coding (HEVC): Reference software. https://hevc.hhi.fraunhofer.de (2019). Accessed 08 May 2019.
33. Ghosh, A., Dehuri, S.: Evolutionary algorithms for multi-criterion optimization: a survey. Int. J. Comput. Inf. Sci. **2**, 38–57 (2004)
34. Song, C., Ju, L., Jia, Z.: Hybrid scratchpad and cache memory management for energy-efficient parallel HEVC encoding. IEEE Int. Conf. Comput. Design (2015). https://doi.org/10.1109/ICCD.2015.7357185
35. Chen, C., Huang, C., Chen, Y., Chen, L.: Level C+ data reuse scheme for motion estimation with corresponding coding orders. IEEE Trans. Circuits Syst. Video Technol. (2006). https://doi.org/10.1109/TCSVT.2006.871388
36. Afonso, V., Maich, H., Audibert, L., Zatt, B., Porto, M., Agostini, L., Susin, A.: Hardware implementation for the HEVC fractional motion estimation targeting real-time and low-energy. J. Integrat. Circuits Syst. **11**, 106–120 (2016)



Roger Porto is a Professor at Sul-rio-grandense Federal Institute of Science, Education and Technology (IFSul), Brazil, and a member of Group of Architectures and Integrated Circuits (GACI) and Video Technology Research Group (ViTech) at Federal University of Pelotas (UFPel), Brazil. His research interests include hardware design, approximated computing and video coding. He received his Ph.D. degree in Computer Science from Federal University of Pelotas (UFPel), Brazil.

Murilo Perleberg received the B.S. degree in computer engineering from the Federal University of Pelotas (UFPel), RS, Brazil, in 2018. He is currently pursuing the M.S. degree in computer science at UFPel. He is member of the Video Technology Research Group (ViTech) at the UFPel. His research interests include VLSI hardware design for video coding standards.



Vladimir Afonso received the M.S. degree in computer science from Federal University of Pelotas, Brazil, in 2013 and the Ph.D. in Microelectronics at the Federal University of Rio Grande do Sul, Brazil, in 2019. Since 2009 he is a Professor at the Sul-rio-grandense Federal Institute, Brazil. His research interests include digital systems, embedded systems and video coding.

Bruno Zatt is a Professor at the Federal University of Pelotas (UFPel), Brazil, and a member of the Group of Architectures and Integrated Circuits (GACI). His research interests include digital hardware design, digital signal processing, video processing, and embedded systems project. He received his Ph.D. degree in Microelectronics from Federal University of Rio Grande do Sul (UFRGS), Brazil. He is a Senior Member of IEEE. He is a Brazilian Distinguished Researcher through a CNPq PQ-2 grant.

Nuno Roma received the Ph.D. degree in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisbon, Portugal, in 2008. Currently, he is an Assistant Professor with the Department of Electrical and Computer Engineering of IST and a Senior Researcher of the Signal Processing Systems Group (SiPS) of Instituto de Engenharia de Sistemas e Computadores R&D (INESC-ID). His research interests include computer architectures, specialized and dedicated structures for digital signal processing, energy-aware computing, parallel processing and high-performance computing systems. He contributed to more than 100 manuscripts to journals and international conferences and served as a Guest Editor of

Springer Journal of Real-Time Image Processing (JRTIP) and of EURASIP Journal on Embedded Systems (JES). He has also acted as the organizing chair of several workshops and special sessions. He has a consolidated experience on funded research projects leadership and he is member of several research Networks of Excellence (NoE), including HiPEAC (European Network of Excellence on High Performance and Embedded Architecture and Compilation). Dr. Roma is a Senior Member of IEEE and ACM.

Luciano Agostini received the M.S. and Ph.D. degrees in Computer Science from Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2002 and 2007 respectively. He is a professor since 2002 at Federal University of Pelotas (UFPel), Brazil, where he leads the Video Technology Research Group (ViTech) and the Group of Architectures and Integrated Circuits (GACI). He is advisor at the UFPel Master and Doctorate in Computer Science courses. He was the Executive Vice President for Research and Graduate Studies of UFPel from 2013 to 2017. He has more than 200 published papers in respected international journals and conferences. His research interests include 2D and 3D video coding, algorithmic optimization, arithmetic circuits, and dedicated hardware design. Dr. Agostini is a Senior Member of IEEE and ACM, and a Member of SBC and SBMicro Brazilian societies. He is also a member of the IEEE SPS, CS and CAS societies and at CAS he is a member of the Multimedia Systems and Applications Technical Committee (MSATC). He is a Brazilian Distinguished Researcher through a CNPq PQ-1D grant.

Marcelo Porto is Professor at the Federal University of Pelotas (UFPel), Brazil, and a member of the Video Technology Research Group (ViTech) and the Group of Architectures and Integrated Circuits (GACI). He received the M.S. and Ph.D. degree in Computer Science, from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2008 and 2012, respectively. His research interests include video coding, motion estimation algorithms, complexity reduction and energy-efficient VLSI design for video coding. He is currently the Coordinator of the Postgraduate Program in Computing (PPGC) of UFPel, he is also a Brazilian Distinguished Researcher through a CNPq PQ-2 grant and a senior member of IEEE.