Energy and Rate-Aware Design for HEVC Motion Estimation Based on Pareto Efficiency

Murilo Perleberg¹,², Vladimir Afonso¹,²,³, Ruhan Conceição¹,²,³, Altamiro Susin³, Luciano Agostini¹,², Marcelo Porto¹,², Bruno Zatt¹,²

¹Group of Architectures and Integrated Circuits – GACI, Federal University of Pelotas – UFPel, Pelotas, Brazil
²Video Technology Research Group – ViTech, Federal University of Pelotas – UFPel, Pelotas, Brazil
³Graduate Program on Microelectronics – PGMicro, Federal University of Rio Grande do Sul – UFRGS, Porto Alegre, Brazil

Abstract—This paper presents a high-throughput energy and rate-aware hardware design for the Motion Estimation (ME) according to the High Efficiency Video Coding (HEVC) standard. The hardware design implements a modified Test Zone Search (TZS) algorithm to perform Integer Motion Estimation (IME) as well as the Fractional Motion Estimation (FME) defined by the HEVC standard. Based on evaluations with the HEVC Reference Software, a complexity-reduction strategy was adopted in the developed architecture that mainly consists of supporting only the 8x8, 16x16, 32x32, and 64x64 Prediction Unit (PU) sizes rather than using the 24 possible PU sizes. The architecture allows an external control unit selects a subset of these four PU sizes according to the energy and rate targets for a specific application. The possible operation points were determined based on Pareto Efficiency. The architecture was described in VHDL, and the synthesis results for ASIC 45nm Nangate standard cells show that the developed architecture can process at least 53 frames per second (fps) considering Ultra-High Definition (UHD) 4320p videos. When an average-case of processing is considered, the architecture is able to process 112fps at UHD 4320p resolution.

Index Terms—Video Coding; Hardware Design; HEVC Standard; Motion Estimation; Pareto Efficiency.

I. INTRODUCTION AND MOTIVATION

Nowadays, the popularity of digital videos is continuously growing, mainly due to the possibility of streaming from the internet and the increasing video-capability of mobile devices, such as smartphones and tablets. Given the huge amount of data needed to represent a high-definition video and, consequently, to transmit and store it, the use of compression techniques is mandatory. Even so, the applications and devices with support for digital videos usually require a dedicated hardware to implement the compression techniques and obtain real-time processing.

Among the available standards, the High Efficiency Video Coding (HEVC) [1] is considered the state-of-the-art [2] in video coding. A video encoder consists of several steps, which each one explores different data redundancies presented in the video sequence. To efficiently encode videos, the HEVC encoder divides each frame into smaller square-shaped blocks called Coding Tree Units (CTUs) with 64x64 samples, by default [3]. The CTUs can also be divided into smaller blocks to improve the coding efficiency. In the HEVC, the step responsible for the most important gains in terms of compression is the Motion Estimation (ME) [4], which is part of the inter-frames prediction and the subject of this work.

According to [4], the inter-frames prediction is responsible for reducing the BD-Rate of the HEVC encoder in 84.71% when applied (evaluated with RA configuration – see Section III), i.e., it reduces 84.71% the bit rate considering the same image quality. To apply the ME, the HEVC divides each CTU into blocks of variable size called Prediction Units (PU) [3]. PUs can assume sizes from 4x8 up to 64x64 samples, with a total of 24 different PU sizes available during the ME process. Thus, ME represents the most complex encoding tool in the HEVC encoder [4]. According to [4], the encoding time increases between 163%–1,567% when using inter-frames prediction, depending on the video-input characteristics (evaluated with RA configuration)

The HEVC ME consists of two steps that are applied to each PU. The first step of the ME is the Integer Motion Estimation (IME), where the video encoder uses a Block Matching Algorithm (BMA), such as Test Zone Search (TZS), and a similarity criterion, such as Sum of Absolute Differences (SAD) [4], to compare each block of the current frame with blocks from reference frames (frames previously encoded). After the IME finds the most similar block, a second step, called Fractional Motion Estimation (FME), is applied over this block to refine the IME result and to improve the coding efficiency. This refinement consists of an interpolation process that generates up to 48 new blocks at fractional positions (of ½ and ¼ of a sample) around the best IME result. A new search among these interpolated blocks is performed looking for a better matching if compared to the IME result. According to the [4], the FME is responsible for reducing the BD-Rate of the HEVC encoder in 9.63% (evaluated with RA configuration) at the cost of an encoding-time increase of 59.44%.

Several works as [5] and [6] can be found in the literature proposing a dedicated hardware for the IME step according to the HEVC standard. Other works as [4] and [7], focusing on the HEVC FME step, can also be found. However, few works, such as [8], implement a complete architecture for the ME, including IME and FME steps. Thus, there is an uncovered research space for complete HEVC ME architectures that consider both the IME and FME features. Furthermore, the works found in the literature rarely take into account energy and rate issues, and when they pre-
sent a discussion about energy and rate features, some ME aspects are neglected, as will be better explained throughout this article.

This paper presents an energy and rate-aware hardware design for a complete HEVC ME, including both IME and FME steps, based on Pareto Efficiency to determine optimal operation points. The architecture can process at least 53 frames per second (fps) at Ultra-High Definition (UHD) 4320p (7680x4320 pixels) resolution. This architecture contains specific modules to efficiently deal with four different PU sizes: 8x8, 16x16, 32x32, and 64x64. These PU sizes were defined through evaluations performed with the HEVC Reference Software (HM – HEVC Test Model) [9]. Each one of these modules is activated according to pre-defined rate-energy targets to achieve the maximum tradeoff between coding efficiency and energy consumption.

II. HEVC ME BACKGROUND AND RELATED WORKS

This section presents an HEVC ME background and discusses some prominent related works.

A. HEVC Motion Estimation Background

As previously mentioned, the ME is considered the most important HEVC encoding tool due to its impact on the compression results. The ME uses previously encoded frames to find blocks similar to the current block (block being encoded). Thus, it avoids transmission and storage of a huge amount of data, since only the difference between these blocks and a motion vector (that represents the displacement between the position of the current block and the chosen block) are required to represent the video.

Given the complexity of finding similar blocks searching in the complete reference frames, ME applies the BMA to a pre-defined and reduced Search Area (SA) (a.k.a. Search Window (SW)) within the reference frames. This process is divided into two major steps, the IME, where the BMA is applied, and the FME, where a refinement is applied around the best matching found during IME. By default, the HM defines the TZS fast algorithm as BMA of the IME step. In the sequence, two subsections briefly explain the HEVC ME. First, the IME is explained focusing on the TZS algorithm. After, a subsection explains the FME.

1) Integer Motion Estimation using TZ Search

There are several algorithms to perform the IME. By default, the HM uses a prominent IME algorithm, the TZS, which is also the main component of the IME step. The TZS drastically reduces the computational effort while maintaining the bit rate approximately unchanged, and it is composed of four steps, which are the Prediction, First Search, Raster, and Refinement [10]. The TZS flowchart is presented in Fig. 1 according to the standard configurations of the HM.

The Prediction is the first step of the TZS, and it has five different predictors, which select specific blocks to be compared based on their probability be the best matching block. The predictors are: (Zero) the motion vector of the co-located block (block inside the reference frame with the same position of the current block); (2Nx2N) the motion vector of the bigger 2Nx2N partition [3]; (Upper) the motion vector of the block above the co-located block; (Righter) the motion vector of the block to the left of the co-located block; and (Median) which uses the average of the motion vectors of neighbor blocks. The next TZS steps are performed around the best result of the Prediction step.

The First Search starts searching with four blocks around the position defined by the Prediction step. Then, the search expands in a diamond scheme until it reaches a stop condition. The second, third, and fourth expansion levels search eight blocks at a distance of two, four, and eight samples apart from the initial position, respectively. In the next expansion levels, the search keeps expanding, but comparing 16 blocks in each expansion, at a distance of 16, 32 and 64 samples from the initial position (see Fig. 2-a). The First Search step has two stop conditions: 1) If the step reaches the limit of the SA, distant 64 samples (see Section III-D) to the block defined by the Prediction step, which results in up to seven iterations; 2) If the algorithm expands three consecutive times without finding a better matching than previous expansion levels. Thus, if any of these conditions occur, the First Search step is finished, and the Raster or the Refinement steps are applied.

The Raster step is used when the distance between the best block found in the previous steps and the center of the SA is bigger than the iRaster constant (iRaster is defined as 5 by default in the HM). In this case, a subsampled Full Search is performed before the Refinement step.

The Refinement step, as the name suggests, is responsible for refining the best result obtained with other TZS steps previously performed. This Refinement applies several iterations, where each iteration uses a similar process when compared with the First Search, i.e., it uses the same stop conditions and the same heuristic to expand the search, as presented in Fig. 2-a.

Each one of these iterations starts from different positions. While the first iteration starts around the position of...
the best result previously found by another TZS step (First Search or Raster), the next iterations are made around the position of the best result of the previous refinement iteration. Only in the case that the iteration does not find a result better than the previous best result, the Refinement ends, finishing the execution of the TZS. So, the Refinement will run recursively with several iterations while it finds a better result. Finally, after the Refinement, the TZS completes its execution, and delivers, as a result of the IME step, the most similar block to the current block.

2) Fractional Motion Estimation

FME is a refinement technique applied to the reference blocks that presented the best result in the IME. As defined in the HEVC [1], this refinement allows the use of motion vectors with quarter-pixel precision, improving the coding efficiency. The FME uses the samples of the best IME result as the input of interpolation filters defined by the HEVC [1] to generate sample values at fractional positions and, consequently, new blocks at fractional positions. Since these filters can use some samples that are outside the limits of the reference block (block chosen by the IME), an edge of four samples from each side of the block is needed for the interpolation.

Each one of the new blocks generated by the FME represents a small movement to the sides and, therefore, one of them can better represent the current block. So, the new blocks can be compared with the current block. By default, in the HM, first the FME performs the interpolation and comparison of the eight half-position blocks. Only after that, it performs the interpolation and comparison of eight quarter-position blocks located around the best block at half positions. So, the FME process implemented in the HM uses 16 of 48 possible fractional blocks to refine the best IME result.

Given its processing, the FME can be divided into two steps: 1) The Interpolation, which generates the sample values at fractional positions by applying filters on the samples of the best IME result; 2) The Search and Comparison, which compares the similarity of all new blocks with the current block by using some similarity criterion [4].

B. Related Works

There are several works available in the literature proposing a hardware design for one of the ME steps following the definitions of the HEVC standard. However, only a few works implement complete architectures for both IME and FME steps. In general, these works are not able to handle real-time encoding of UHD 4320p videos.

The works [5] and [6] present solutions for the IME. In [5] the authors propose an algorithm similar to the Test Zone Search for the IME and its architecture, comparing much more blocks on the First Search, but with only one Refinement step. It uses two SRAM memories and a data-reuse scheme to ensure the throughput of UHD 2160p (3840x2160 pixels) videos at 30 fps. The work [6] also presents a hardware design for the IME focusing on an ASIC design, which implements the Full Search algorithm. It can process all PU sizes supported by the HEVC and its architecture can also process UHD 2160p videos at 30 fps. Whereas the work [5] presents power results and partially measures the impact of its strategy in terms of compression, [6] does not present power analysis or the impact of its strategy regarding compression. Results in [5] do not report the impact of not using bidirectional prediction in its evaluations.

The works [4] and [7] present hardware designs for the FME. In [4] it is proposed an interpolation unit to generate the new blocks at fractional positions and a search and comparison unit to perform the comparison between the generated blocks and the current block. The architecture processes the four square-shaped PU sizes by composing all supported PU sizes with 8x8 blocks. It can achieve real-time processing of UHD 2160p videos at 60 fps. In [7] it is presented an architecture focusing ASIC technology where a strategy to reduce the complexity of the FME is used. This strategy consists of replacing the 7-tap and 8-tap filters used to calculate the quarter-pixel samples in the HEVC by bilinear filters. The architecture developed by [7] has a maximum processing of 30 fps considering UHD 4320p videos. Both [4] and [7] present a power analysis and coding-efficiency evaluation, but they do not consider the impact of using only one reference frame and disable the bidirectional prediction.

The work proposed in [8] is one of the few works that present a hardware design for both the IME and the FME algorithms. For the IME, it applies the TZS only over the 8x8 PU size, so the bigger PU sizes are predicted based on this result. The developed hardware design can process UHD 2160p videos at 30 fps. Despite [8] presenting a power analysis and evaluations to verify the compression losses of its strategy, the presented results do not consider the impact of avoiding bidirectional prediction.

III. HEVC ME EVALUATIONS

Given the HEVC ME high complexity, the use of strategies to speed up the process is highly desired in order to satisfy the requirements for real-time processing of high and ultra-high resolution videos. Considering energy-constrained devices, as smartphones, tablets, camcorders, and others, those strategies are mandatory. In this scenario, different strategies to reduce the HEVC ME complexity can be used. However, any modification in the coding process must be rigorously evaluated before being adopted. It is important to notice that these modifications can be applied to the encoder since the bit stream remains compliant with the decoder HEVC standardization. The next subsections show the experimental setup used to guide all evaluations done in this work with the HEVC reference software. Different analyses were made to lead to the development of a high-throughput energy and rate-aware hardware design for the HEVC ME based on Pareto Efficiency.

A. Experimental Setup

With the goal of finding a tradeoff between complexi-
ty/energy decrease and quality/compression losses, several experiments with the HEVC Reference Software were conducted according to the Common Test Conditions (CTC) document [11] recommended by the Joint Collaborative Team on Video Coding (JCT-VC). The CTC document recommends 24 video sequences for tests, which are divided into classes according to their resolutions and features. These test sequences have from small resolutions as the WQVGA (416x240 pixels) to high resolutions as the WQXGA (2560x1600 pixels) with a different number of frames and frame rates. In general, each class has video sequences of a specific resolution, except in Class F, that presents screen content videos at different resolutions. According to the CTC document, all sequences must be encoded in the experiments for four Quantization Parameters (QPs): 22, 27, 32, and 37.

This way, the experiments performed in this work used the 24 test sequences, and the four QPs defined by the CTC document. The number of frames to be encoded was stipulated to be equivalent to two seconds based on the frame rate of the video sequences. Furthermore, all evaluations were performed through the HM 16.15 version [9] using the Main Profile (low-complexity profile) and considering the Random Access (RA) temporal configuration.

B. PU-size Occurrence and Representativeness Analyses

As previously mentioned, the HEVC ME brought innovation compared to its previous standards, by supporting 24 different PU sizes (square-shaped, symmetric and asymmetric sizes) in order to obtain high coding efficiency. However, this improvement results in a high complexity due to the number of calculations processed by the ME. In addition, the decision about the best PU size, i.e., the size that results in the best tradeoff between compression and distortion, only occurs after the evaluation of these 24 PU sizes by the other steps in the encoding loop (as the Transform and Quantization steps) and the comparison with the other encoding methods implemented in the HEVC encoder (Intra-frame prediction, for instance).

A simple way to reduce the high complexity of HEVC ME is reducing the number of PU sizes that must be evaluated during the ME process, which has been done by some works presented in the literature. In a previous work [4], we reduced the PU sizes to the four square-shaped sizes (8x8, 16x16, 32x32, and 64x64) based on a wide and careful software evaluation. Such strategy of reducing the PU sizes evaluated in the ME process was based on measuring compression and image quality achieved by using some specific PU sizes. Such work investigates the incidence of each PU size in the inter-frames prediction and its representativeness on the frames through experiments with an older version of the HM. The representativeness is defined as the percentage of pixels that are encoded by each PU size considering all videos.

It was observed that the 8x8 PU size is the most frequently selected block size and the 16x16 is the second most often selected size on average. Given that bigger PUs encode a larger amount of pixels in the image, the percentage of pixels that were covered by each PU size was also verified in [4]. Such analysis concluded that the four square-shaped PU sizes (64x64, 32x32, 16x16, and 8x8) are the most representative sizes considering the average of the values for the video sequences.

Since the evaluation in the previous work was made using an older version of the HM, we retrace this analysis about the occurrences and representativeness of the PU sizes using the HM 16.15 version and RA configuration. The results of these analyses can be seen in Fig. 3-a for the percentage of PU-size occurrences and in Fig. 3-b for the percentage of pixels encoded with each PU size. It is important to notice that the results with the current HM version are similar to the results of the previous work and, therefore, the conclusions remain the same.

This way, it was possible to conclude that the square-shaped PU sizes are both frequent and representative when compared to the non-square PUs. These data are used as starting point for the next evaluations presented in this work in order to find efficient complexity-reduction strategies for the HEVC ME.

C. Definition of Complexity-Reduction Strategies Related to the PU-size Constraints

In this work, the impact of reducing the evaluated PU sizes in the HEVC ME over the coding efficiency was investigated. Based on the study presented in [4] about the most selected and representative PU sizes using the HEVC Reference Software, we evaluated the impact in the coding efficiency considering 11 configurations. The methodology used to decide what configurations should be tested, i.e., what PU sizes should be evaluated during the ME process in each case, is based on the conclusion that the four square-shaped PU sizes are more frequent and representative when compared to the non-square PUs. So, the use of the four square-shaped PU sizes was considered the baseline for the sequence of the evaluations. All the possible combinations of the square-shaped PU sizes were tested, totaling the 11 configurations.

The use of only one PU size was not considered here since they led to huge coding-efficiency degradation, as showed in other works in the literature as [12] and [13]. By restricting the evaluated PU sizes to one, the BD-Rate in-
creases at least 19.31% without considering any other constraint to the HEVC encoder [4], which is undesirable for most applications. BD-Rate metric [14] represents the percentage variation in the bit rate for the same image quality, and it is usually used to measure the coding efficiency.

Table I shows the impact regarding BD-Rate of the 11 configurations (called C0-C10 in this work). Note that the impact in the BD-Rate presents a wide variation according to the configuration, with an increase of 3.10% in the BD-Rate when using all square-shaped PU sizes (best case) and an increase of 17.33% when only the PU sizes 32x32 and 64x64 are enabled in the ME (worst case). Note that an efficient configuration tends to present good results for all video classes and resolutions. It could be expected that the use of a fewer set of PUs but with bigger PU sizes could better encode bigger resolutions. However, these results show that smaller PU sizes remain to be important even considering bigger resolutions.

The most works in the literature present an ME (IME and/or FME) complexity-reduction strategy and developed its hardware architecture presenting synthesis results, throughput, among other results, but they do not consider all questions linked to the ME performance, as the number of reference frames, the adopted similarity criterion, and modifications of the BMA algorithms. Frequently, only some of these aspects are considered in these works. Targeting to reduce this misconception presented in the literature, the next subsection measures and discusses additional constraints to be added to the PU-size constraint showed in this subsection.

### D. Definition of Additional ME Constraints

The amount of evaluated PU sizes in the HEVC ME process is responsible for an important part of the ME complexity, however, several other HEVC ME aspects contribute to increasing the computational effort employed in the ME step. It is not uncommon works in the literature that attack only a few aspects related to the ME complexity, neglecting other or just not informing its decisions for that. In order to reduce this misconception presented in the literature, this section aims to show as clearly as possible all the additional constraints used for the architecture development and the total impact on the compression when using these constraints.

As the amount of PU sizes, the Search Range (SR) and, consequently, the size of the SW at the BMA algorithm is applied are determinant to the ME complexity, and it is directly related to memory issues. By default, the CTC document [11] defines an SR equal to 64, corresponding to an SW of \((2*64)+\text{PUsize}-1\)x\((2*64)+\text{PUsize}-1\) pixels. In this work, the adopted SW remains the one defined in CTC document. The BMA algorithm used during the encoding is also responsible for an important part of the HEVC ME complexity. HEVC inherently uses the TZS algorithm in order to efficiently reduce the computational effort at the cost of a small impact in terms of compression and image quality. Even TZS using heuristics to reduce the amount of compared blocks, the computational effort related to TZS remains high in a real-time application context.

In this work, we modified one step of the TZS, the Prediction, and we avoided another TZS step, the Raster. In the Prediction step, only the Zero predictor is maintained so that the other predictors are avoided. This way, the SW cannot be moved to a distant region of the collocated block position, which would cause a significant overhead in terms of memory access and energy consumption. By default, the HM defines SAD as similarity criterion for the IME at integer pixel positions while the Sum of Absolute Transformed Differences (SATD) is applied for the FME at fractional positions increasing the computational effort. Aiming to achieve an energy-aware design, SAD is considered as the similarity criterion for both IME and FME steps. Also, the HM supports bidirectional prediction and multiple reference frames. In order to further reduce the computational effort and allow a memory and energy-efficient hardware we also disabled the bidirectional prediction and limited the number of reference frames to one.

Table II presents the total BD-Rate increase considering all these additional constraints when combined to the PU-size constraint defined in the previous subsection. The accumulated results show that the combination of all constraints little affects the behavior previously observed in the previous subsection, by confronting different configuration results. The magnitude of the results increased, but the conclusions about the impact of the configurations are similar. The changes are in the configuration C2 that presents a better result when compared to the configuration C4, and in the configuration C1 that presents a better result compared with configurations C7 and C8.

The total BD-Rate increase results for the 11 configurations, presented in Table II, are the input data to the application of Pareto Efficiency (see Section V) allowing obtaining a rate and energy-aware design for the ME architecture. The next Section presents the developed ME architecture.

<table>
<thead>
<tr>
<th>Sequence Classes</th>
<th>Configurations (C0-C10) according to the supported square PU sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C0 (8,16)</td>
</tr>
<tr>
<td>Class A - 2560x1600</td>
<td>6.48%</td>
</tr>
<tr>
<td>Class B - 1920x1080</td>
<td>7.81%</td>
</tr>
<tr>
<td>Class C - 832x480</td>
<td>5.74%</td>
</tr>
<tr>
<td>Class D - 416x240</td>
<td>5.97%</td>
</tr>
<tr>
<td>Class E - 1280x720</td>
<td>7.83%</td>
</tr>
<tr>
<td>Class F - Several</td>
<td>4.73%</td>
</tr>
<tr>
<td>Average</td>
<td>6.42%</td>
</tr>
</tbody>
</table>
IV. DEVELOPED ME ARCHITECTURE

The developed ME architecture implements the IME, with a modified TZS algorithm, and the FME defined by the HEVC standard with its respective interpolation filters. Dedicated modules have been developed to work with the different PU sizes, where each module has all IME and FME units needed to process one PU size. The SAD was adopted in this work as the similarity criterion for the comparison between two blocks, because it can be easily implemented in hardware, only using adders and subtractions. In this work, the dedicated modules process the four PU sizes defined in Section III.

The ME architecture maintains a high throughput since it employs four modules to deal with PUs of 8x8 samples, two for PUs of 16x16 samples, one for PUs of 32x32 samples, and one for PUs of 64x64 samples (all modules can work in parallel, depending on the rate-energy operation point).

A. Modified TZS Implementation

The hardware design for the IME implements a modified version of TZS algorithm, which was previously implemented into the HM. This way, only the Zero predictor was maintained in the Prediction step. Furthermore, all TZS expansions of the First Search and Refinement steps receive 16 reference blocks from memory. To do this, the first and second expansions were unified along with the blocks more closed to the current block for reach the 16 reference blocks, and the third and fourth expansions were also unified. Thus, the total number of expansions was decreased of seven (original TZS) to five in this modified version, as can be seen in Fig. 2.

The operative part of the TZS design can be seen in the Fig. 4-a. The adopted scheme for the operative part can process all samples of one block line per clock cycle, where each TZS module receives the current block and 16 reference blocks from memory and processes one line of all 16 reference blocks per clock cycle. Due to this, the TZS module needs eight clock cycles to calculate one expansion with 8x8 PUs, 16 clock cycles to calculate one expansion with 16x16 PUs, 32 clock cycles with 32x32 PUs and 64 clock cycles with 64x64 PUs.

To achieve the real-time processing of UHD videos, the comparison between the reference blocks and the current block is made using a Search and Comparison Unit (SCU) similar to one developed by [4], but with modifications and optimizations. Also, while [4] developed its comparator based on the processing of 8x8 blocks for any of the supported blocks of its architecture, in this work we developed new SCU units specialized for processing of bigger block sizes, increasing the parallelism level and the throughput.

The SCU, each reference block line passes along with the current block line by one SAD Tree, which is responsible for calculating the difference between those lines. Each SAD Tree performs the absolute difference between all samples of the lines. After that, the absolute differences between the samples are added two by two. Thus, the difference between these block lines results in 16 SAD values, one of each reference block line.

Fig. 5 shows the architecture of a SAD Tree used for processing 8x8 PU size blocks, and its three pipeline registers (it requires four cycles). For larger PU sizes each SAD Tree has more inputs and series-connected arithmetic operators, so a greater number of pipeline registers is needed to maintain a similar throughput. These SAD Trees require four, five, and six pipeline registers for the processing of 16x16, 32x32, and 64x64 PU sizes, respectively.

The values from the SAD Trees are accumulated by 16 SAD Accumulators (one for each reference block) while the SAD Tree calculates the rest of the block lines. After the SAD Trees calculate all lines of blocks, the values on the SAD Accumulators are exactly the SAD of each reference block considering the current block. The architecture of the SAD Accumulator for one block can be seen in Fig. 6-a, where the value of m (input sample bit depth) depends on the processed PU size, assuming the values of 11, 12, 13, and 14 bits for the 8x8, 16x16, 32x32, and 64x64 PU sizes, respectively. The SAD Accumulator outputs, represented by n (output sample bit depth), can assume values of 14, 16, 18, and 19 bits for the 8x8, 16x16, 32x32, and 64x64 PU sizes, respectively.

Finally, the SAD Comparator step processes these SAD values along with their motion vectors. The SAD Comparator

---

Table II. Percentage variation in the BD-Rate with limited PU sizes by using the additional ME constraints.

<table>
<thead>
<tr>
<th>Sequence Classes</th>
<th>Configurations (C0-C6) according to the supported square PU sizes</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A - 2560x1440</td>
<td>Class B - 1920x1080</td>
<td>Class C - 832x480</td>
<td>Class D - 416x240</td>
<td>Class E - 1280x720</td>
<td>Class F - Several</td>
<td>Average</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cs (8,16)</td>
<td>Cs (8,32)</td>
<td>Cs (8,64)</td>
<td>Cs (16,64)</td>
<td>Cs (32,64)</td>
<td>Cs (8,16,32)</td>
<td>Cs (8,16,64)</td>
<td>Cs (8,32,64)</td>
<td>Cs (16,32,64)</td>
</tr>
<tr>
<td>16.82%</td>
<td>16.50%</td>
<td>18.62%</td>
<td>16.52%</td>
<td>17.24%</td>
<td>22.72%</td>
<td>14.37%</td>
<td>15.08%</td>
<td>16.02%</td>
</tr>
<tr>
<td>21.06%</td>
<td>19.97%</td>
<td>23.10%</td>
<td>20.01%</td>
<td>21.20%</td>
<td>27.30%</td>
<td>17.30%</td>
<td>18.42%</td>
<td>19.05%</td>
</tr>
<tr>
<td>16.68%</td>
<td>18.64%</td>
<td>20.64%</td>
<td>21.17%</td>
<td>21.65%</td>
<td>35.32%</td>
<td>15.53%</td>
<td>16.23%</td>
<td>18.43%</td>
</tr>
<tr>
<td>16.64%</td>
<td>19.02%</td>
<td>19.87%</td>
<td>24.64%</td>
<td>24.85%</td>
<td>48.27%</td>
<td>16.06%</td>
<td>16.53%</td>
<td>18.92%</td>
</tr>
<tr>
<td>13.39%</td>
<td>13.49%</td>
<td>16.28%</td>
<td>12.92%</td>
<td>13.19%</td>
<td>20.94%</td>
<td>10.75%</td>
<td>11.68%</td>
<td>12.99%</td>
</tr>
<tr>
<td>10.53%</td>
<td>11.12%</td>
<td>13.69%</td>
<td>13.22%</td>
<td>14.38%</td>
<td>21.94%</td>
<td>9.50%</td>
<td>10.03%</td>
<td>10.71%</td>
</tr>
<tr>
<td>Average</td>
<td>16.17%</td>
<td>16.73%</td>
<td>18.98%</td>
<td>18.38%</td>
<td>19.08%</td>
<td>29.68%</td>
<td>14.19%</td>
<td>14.94%</td>
</tr>
</tbody>
</table>
tor compares all SADs, two by two, and delivers the smallest SAD and the motion vector related to this SAD. The SAD Comparator needs four cycles to process 16 SAD values. The Fig. 6-b shows the SAD Comparator of two blocks.

As the FME developed by [4] simultaneously compares a maximum of 12 blocks, it used 12 SAD Trees. The IME SCU proposed in the current work compares 16 blocks, so 16 SAD Trees are needed in any SCU unit. Furthermore, [4] used 48 SAD Accumulators because it generates 48 new blocks for comparisons in the FME. Here, only 16 SAD Accumulators are needed since the TZS module always compares 16 reference blocks.

To improve energy efficiency, some optimizations on the SCU are done. The entire SCU (units for different PU sizes) supports clock gating. The clock gating adds a logic circuit to the clock for reducing the energy consumption, as can be seen in the Fig. 6-c. This way, the clock is switched according to the architecture works. Inside the SCU, energy is saved by deactivating the registers used by the SAD Tree, by the SAD Accumulator, and by the SAD Comparator, when a given PU size is not processed.

The control part of the TZS is responsible for dealing with the stop conditions in each step of the TZS, as explained in Section II. As the TZS has some stop conditions, and none of the stop conditions limits the number of iterations that can be done in one SA, the TZS can freely go through all the blocks of the SA if the Refinement step always finds a smaller SAD. So, the number of cycles that are needed for obtaining the SAD of a block is undefined and depends on the data. In our hardware design, we limited the number of candidate blocks evaluated for each PU-size module to 240. This choice is determinant to the guarantee throughput, and it is based on evaluations with the HM using the same experimental setup of the Section III (considering all ME constraints). The Fig. 7 shows a histogram that represents the number of TZS candidates that are tested until the TZS ends. It is possible to observe that in 93.33% of the cases, TZS evaluates a number of candidate blocks lower than 240. On average, TZS tests 113 candidate blocks before it ends.

Fig. 8 shows a data flow of our modified TZS algorithm with three different examples of termination (without the pipeline stages). The center gray squares represent the TZS steps, including the First Search, and in the sequence, the Refinement steps. Green squares represent expansions where a better result than the previous result was found, while red squares represent expansions where a better result was not found. In the first iteration of the TZS, the First Search ends by reaching the maximum of five expansions due to the maximum range of 64 samples from the center (CTU size). Also, in the first iteration, several instances of the Refinement are used while a better result is found. This way, the Fig. 8 shows the unpredictability of clock cycles required to process a block by TZS because a new Refinement step is done while a better result than the previous best result is found. The second iteration of TZS shows when the Refinement is not used because the First Search does not find a better result that the collocated block. Finally, the third iteration of the TZS shows a case at the First Search ends after three expansions without found a better result. The minimal number of cycles needed by the TZS design is 31, 56, 105, and 202 to 8x8, 16x16, 32x32, and 64x64 PU-size modules, respectively. These values consider that the Refinement is not used due to the First Search does not find a better result that the collocated block.

B. Fractional Motion Estimation Implementation

As shown in Section II, the FME algorithm has two
steps, one to perform the interpolation of new blocks at fractional positions, and another to compare the SAD of these new blocks with the best IME result. Here, these FME steps are implemented with the Interpolation Unit and the SCU, respectively. These units are similar to the ones developed in our previous work [4], but with some hardware-oriented optimizations. The FME architecture can be seen in Fig. 4-b.

The Interpolation Unit has a multiplexer to select what samples will be used as the input to interpolate the fractional samples. These samples can be provided from a reference frame (based on the best result found by the IME step) stored in the memory, or from an internal buffer. After that selection, a set of filters is used to generate the samples at fractional positions around the positions of the block provided by the IME. As implemented in our previous work [4] and different to the HM implementation (but HEVC compliant), all 48 possible fractional blocks are generated to be after compared in the SCU.

Three types of filters are used to calculate each sample at fractional position depending on the sample position [1]. To generate the fractional blocks, these three types of filters work in parallel, using the same input samples for a given instant. Details of the development of these filters can be found in [4].

The generated samples may be displaced horizontally, vertically or both (diagonal) in relation to the integer positions given by the IME result, as shown in Fig. 9. The processing is sequentially divided according to the position of the samples generated by the filters. Since the filters can use some samples that are outside the limits of the reference block (block chosen by the IME), an edge of four samples located outside the block to be interpolated is also required from the memory or the internal buffer.

While the filters generate new blocks, the Interpolation Unit stores the horizontal fractional samples in an internal buffer once they are needed to interpolate the diagonal samples. Finally, a Clip operation is used after the filtering process to maintain the samples with 8-bit before the processing by the SCU. The outputs of the Clip operation have values between 0 and 255, that is, if the input value is negative, it is set to 0, and if the value is higher than 255, it is set to 255. Values between 0 and 255 are maintained.

Due to the number of new blocks generated by the filters and, consequently, the processing time to compare these blocks with the current block, the processing is sequentially divided into horizontal, vertical and diagonal positions.

Firstly, the horizontal positions will be generated and processed, so the Interpolation Unit receives one horizontal line of the best IME block with an edge of samples per clock cycle, and generates the new samples to compose six new horizontal fractional blocks. These blocks are sent to the SCU and also stored in the internal buffer (no clipping in this case). The internal buffer also stores an edge of samples since they are needed for the diagonal-samples processing. Some additional horizontal fractional lines are calculated using lines at integer positions from memory. In the sequence, the vertical positions are processed, where the Interpolation Unit receives one vertical column of the IME result block and an edge samples per clock cycle. So, six new vertical fractional blocks are generated and sent to the SCU. Finally, the diagonal positions are processed by the Interpolation Unit using the horizontal samples stored in the internal buffer to generate 36 new diagonal fractional blocks. In the sequence, these diagonal fractional blocks are sent to the SCU.

The FME SCU receives the samples at fractional positions generated by the Interpolation Unit and performs the comparisons in a similar way to the operative part of the TZS (see Fig. 4-a). Twelve SAD Trees are used to calculate the difference between the samples at fractional positions and the samples of the current block. Forty-eight SAD Accumulators are used to accumulate the SAD values of the fractional blocks. After that, the SAD Comparator compares the SAD of all 48 new blocks at fractional positions with the best IME SAD.

Some optimizations are done on the FME architectures targeting a low-power circuit. These low-power optimizations include the insertion of clock gating in the entire FME architecture, allowing disabling it when the FME finished the processing of a block and it waits for the next IME result to start the processing of a new block.

The clock-gating logic of SAD Trees and the SAD Comparator has an extra control. This extra control disables half of the SAD Trees in the processing of horizontal and vertical samples because only six SAD Trees are needed for processing those samples. It also disables the SAD Comparator in the most of the FME processing, because that unit is only needed for six cycles, which occurs after the processing of all new blocks.

The number of cycles needed to complete the FME process is constant to each PU size and sample position, in ad-
dition to clock cycles due to the pipeline registers. The Interpolation Unit has three pipeline registers that are located inside the filters. In the FME SCU, the SAD Trees have the same number of pipeline registers of the SAD Trees used on the IME, and the FME SAD Comparator needs five pipeline registers to compare the 48 SADs of the fractional blocks and the SAD of the IME result. One additional cycle is needed because of the register used on the SAD Accumulators. This way, 12, 13, 14 and 15 pipeline registers are used in the FME for 8x8, 16x16, 32x32 and 64x64 PU sizes, respectively. With this, the total clock cycles needed to complete the FME processing is 63 clock cycles for 8x8 PUs, 104 clock cycles for 16x16 PUs, 185 clock cycles for 32x32 PUs, and 346 clock cycles for 64x64 PUs.

**C. Complete ME Architecture**

The complete ME architecture can be seen in Fig. 10, where the TZS and FME modules for all PU sizes can be seen along with their connections. Each module has predefined blocks to process. A global control unit controls the flags of those modules and the moments when the clock gating of each module that process a determinate PU size enable or disable the processing, waiting for the results of other units. When each FME module finishes its execution, the global control unit copies the result to a SAD table. This table stores the SAD value and the associated motion vector. Four tables are needed to store the results of one CTU, one table for each used PU size.

The Fig. 11 shows a global timeline for the operation of all modules. It can be seen that each 8x8 module runs 16 times while the 64x64 module runs only once. The time sampling used to depict the TZS processing in Fig. 11 considers the minimal conditions for occurring of Refinement, i.e., at least four expansions in the First Search and other three in the Refinement. When each module completes the processing of its predefined blocks, it stops and waits the other modules also finish the CTU processing.

When all modules end the CTU processing, the Global ME Comparator starts. It compares the results for each PU size to find the PU partition scheme that will result in the smallest final SAD to encode a CTU considering the ME, thus ensuring the high coding efficiency. The Global ME Comparator needs 21 clock cycles. In cycles 1-16 it compares the results of 8x8 PUs with the 16x16 PUs, comparing the sum of four 8x8 SAD results with one 16x16 SAD result that was in the same position on CTU (one comparison per clock cycle). In cycles 17-20, the sum of four comparisons results, made in the first 16 cycles, are compared with one 32x32 SAD result (one comparison per clock cycle). At cycle 21, the sum of four comparison results, made in the 17-20 cycles, are compared with the result of using a 64x64 PU.

In each of these clock cycles, the ME sends as output the SAD resultant of the comparisons and the vector associated with this SAD. If a PU size is not used, the Global ME Comparator ignores its comparison and sends the result of the modules that should be compared in that cycles, if it is valid.

**D. Memory Requirement Discussion**

As previously stated, the HEVC ME process demands a huge number of operations in order to decide which is the most similar reference block. This process is performed for each PU size supported by the architecture, leading to an intense processing and memory communication.

The memory requirements to satisfy real-time processing of UHD videos considering the same HEVC ME implemented in the HM is not feasible with current memory technology. Even using all ME constraints applied in this work that significantly reduces the processing and the memory communication, when no on-chip SRAM is employed, the communication demanded for encoding HD 1080p videos in real time reaches 105.75GB/s (worst case for our architecture). Such performance is not feasible, especially for embedded low-power memories such as LPDDR4, which provides no more than 51.2GB/s, in ideal cases [15].

The focus of this work is in the ME processing-unit, and the architecture was design to provide real-time processing of UHD videos when integrated to a system with an effi-
cient memory managing hierarchy. Data-reuse schemes, as the Level C / Level C+, are widely used in the literature to reduce the memory communication of the ME, achieving reductions up to 81% in external memory communication [16]. Other prominent data-reuse schemes proposed in the literature reduces the memory communication of the ME with the external memory by 77-88% when compared with the Level C [17]. Therefore, there are data-reuse schemes in the literature able to reduce the memory communication of the ME in up to 97%, which shows that our ME processing-unit design is completely feasible.

These memory management solutions should explore the HEVC ME at the level of a system, including memory issues and reducing external memory communication, reducing on-chip SRAM size (to reduce static/leakage consumption), implementing a data-reuse scheme and memory hierarchy.

E. Individual PU-size Modules Synthesis Results

In order to increase the ME architecture parallelism and to reach high throughput for processing videos UHD 4320p in real time, four specialized modules were developed in this work, so that each one deals with a specific PU size of the four supported square-shaped PU sizes. This subsection separately presents the synthesis results of these modules to allow the estimation of energy consumption for each one of the 11 configurations presented in Section III. These results are needed to define operation points and apply the Pareto Efficiency method that will be presented in the next Section. This strategy of estimating the energy consumption from results of each PU-size module was adopted because, on the contrary, the generation of energy results for all configurations would need a very costly synthesis process.

The developed ME architecture, which includes the PU-size modules, was described in VHDL and synthesized for 45nm Nangate standard-cells [18] using the Cadence RTL Compiler [19] (version 11.1). The gate count is calculated based on 2-input NANDs, and the power dissipation is generated considering a voltage supply of 0.95V. Table III presents the synthesis results considering the frequency needed to process HD 1080p videos at 30 frames per second (36.23MHz). It is important to notice that this frequency already considers the worst-case scenario to reach the target throughput, i.e., the module that requires more clock cycles to process the video (8x8 PU-size module) and the maximum number of TZS candidates supported by our architecture. Furthermore, no low-power techniques are applied to the architectures for this analysis.

As previously discussed, four instances of 8x8 PU-size modules, two instances of 16x16 PU-size modules, and one instance of 32x32 and 64x64 PU-size modules are used to reach the desired parallelism. Note that the power and gate-count results are similar to the different PU-size modules, except to the 64x64 PU-size module. The 64x64 PU-size module presents a larger power dissipation when compared to the other PU-size modules since its targeted parallelism is the processing of one line of 16 reference blocks per cycle. Therefore, as it processes a bigger PU size, it also uses a larger gate count.

V. PROPOSED PARETO-BASED OPTIMAL RATE-ENERGY OPERATION POINTS

The main objective of this Section is to determine the optimal operation points (lower energy consumption and lower bit rate) for the developed ME architecture by using the concept of the Pareto Efficiency and the configurations based on the ME constraints presented in Section III. As previously discussed, those configurations were defined through an offline analysis with the HEVC Reference Software after applying some constraints in the ME process, as the restriction of the PU sizes, the number of reference frames and the BMA algorithm, among others. The more prominent configurations are select by the Pareto Efficiency method based on the BD-Rate impact of each configuration that is presented for the average case of the video sequences and the energy estimation presented in the previous Section.

The Pareto Efficiency explores the fact that multi-objective optimizations could not have a unique solution, but a set of optimal results. This way, the method results in a Pareto frontier (convex curve) composed of the optimal operation points for a given multi-objective optimization problem [20]. An operation point is considered an optimal Pareto point when no other point presents a better result in both observed axis of the multi-objective optimization problem. In other words, an operation point is considered an optimal Pareto point if it is not possible to improve a determinant axis without degrading the other axis.

The HEVC Reference Software uses two variables in RDO (Rate Distortion Optimization) [3] to decide which PU size must be used in the encoding process (including the ME process) based on the bit rate and distortion (bi-dimension problem). Considering the energy-consumption, the problem has now three dimensions to be evaluated. Real-time power evaluation is a very costly process that can be simplified with an offline analysis. Similarly, this three dimension analysis can be simplified to two dimensions by using the BD-Rate metric along with the energy consumption. As previously mentioned, the BD-Rate metric represents the percentage variation in the bit rate for the same image quality. This way, the distortion variable is fixed while the bit rate and the energy can be analyzed with Pareto Efficiency.

The Pareto-based optimal rate-energy operation points applied to the developed ME architecture are also based on the energy consumption. To define the operation points, the energy consumption is estimated through ASIC synthesis, where the synthesis results are generated for each specialized module created to process each PU size (8x8, 16x16,
32x32, and 64x64) as presented in Section IV. This strategy was adopted in order to allow the estimation of energy consumption for any configuration, without making a costly synthesis process for each one of the possible configurations. Furthermore, the energy consumed by the specialized modules of the HEVC ME architecture was estimated considering the processing of one CTU.

Table IV shows the 11 possible configurations used in this work to determine the operation points of the ME architecture based on the Pareto frontier. The supported PU sizes, the BD-Rate increase, and the energy consumption (µJ) for each configuration can be seen in Table IV. Note that the energy consumption is calculated considering the frequency needed to process HD 1080p videos at 30 frames per second (36.23MHz) and the maximum number of cycles spent by the PU-size modules to process a CTU (worst case, with 240 blocks evaluated in TZS).

![Fig. 12 Pareto Efficiency in the rate-energy space.](image)

Table IV. Configurations used to determine the Pareto frontier.

<table>
<thead>
<tr>
<th>Configurations (C0-C10)</th>
<th>PU sizes</th>
<th>BD-Rate Increase (%)</th>
<th>Energy consumption (µJ per CTU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>8x8,16x16</td>
<td>16.17</td>
<td>5.337</td>
</tr>
<tr>
<td>C1</td>
<td>8x8,32x32</td>
<td>16.73</td>
<td>5.578</td>
</tr>
<tr>
<td>C2</td>
<td>8x8,64x64</td>
<td>18.98</td>
<td>6.045</td>
</tr>
<tr>
<td>C3</td>
<td>16x16,32x32</td>
<td>18.38</td>
<td>5.317</td>
</tr>
<tr>
<td>C4</td>
<td>16x16,64x64</td>
<td>19.08</td>
<td>5.784</td>
</tr>
<tr>
<td>C5</td>
<td>32x32,64x64</td>
<td>29.68</td>
<td>6.025</td>
</tr>
<tr>
<td>C6</td>
<td>8x8,16x16,32x32</td>
<td>14.19</td>
<td>8.116</td>
</tr>
<tr>
<td>C7</td>
<td>8x8,16x16,64x64</td>
<td>14.94</td>
<td>8.583</td>
</tr>
<tr>
<td>C8</td>
<td>8x8,32x32,64x64</td>
<td>16.27</td>
<td>8.824</td>
</tr>
<tr>
<td>C9</td>
<td>16x16,32x32,64x64</td>
<td>17.97</td>
<td>8.563</td>
</tr>
<tr>
<td>C10</td>
<td>8x8,16x16,32x32,64x64</td>
<td>13.79</td>
<td>11.362</td>
</tr>
</tbody>
</table>

Fig. 12 Pareto Efficiency in the rate-energy space.

VI. RESULTS AND COMPARISON

The developed architecture was described in VHDL and synthesized for 45nm Nangate standard-cells [18] using the Cadence RTL Compiler [19]. The ASIC results show that the developed architecture reaches a maximum frequency of 1.025 GHz. The power results by processing 1080p@30fps according to the operation point (C0, C6, C10) are showed in Table V, while other results are detailed in the right-most column of Table VI, considering the intermediate energy-efficient operation point (C3), and all techniques of low-power applied to the ME architecture. Due to the high parallelism, where each CTU needs 2368 clock cycles, by assuming that the bottleneck is when the 8x8 modules find the best result using at most 15 iterations, with 148 cycles each, the developed architecture can process at least 53fps considering the UHD 4320p resolution. However, considering the average case showed in Section IV, the architecture can process UHD 4320p videos at 112fps. Therefore, our developed ME architecture is the only solution able to process UHD 4320p videos in real time.

The works [4] and [7] show lower power dissipation than the results achieved in this work, but they only implement the FME. Furthermore, [4] cannot process UHD 4320p videos in real time while [7] can only process UHD 4320p videos at 30fps. The works [6] and [5] only implement the IME. The work [6] is based on the Full Search algorithm, and it needs an elevated frequency to process HD 1080p videos at 30fps. Furthermore, [6] does not present its power dissipation results and its maximum performance does not allow real-time processing of UHD 4320p videos. The work [5] uses an algorithm similar to the TZS, but testing a higher number of candidate blocks. Its maximum performance is limited to processing of UHD 2160p videos at 30fps.

Table V. Power results by processing 1080p@30fps according to the operation point.

<table>
<thead>
<tr>
<th>Operation Point</th>
<th>Power without clock gating (mW)</th>
<th>Power with clock gating (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>58.83</td>
<td>28.17</td>
</tr>
<tr>
<td>C6</td>
<td>92.12</td>
<td>44.92</td>
</tr>
<tr>
<td>C10</td>
<td>178.99</td>
<td>86.21</td>
</tr>
</tbody>
</table>
The work [8] implements both the IME and the FME, and it adopts various simplifications in the IME and FME algorithms to reach throughput enough to process UHD 2160p videos at 30 fps. Among the simplifications adopted by [8], it uses the TZS algorithm only in 8x8 blocks and bigger PU sizes are predicted based on this result. Furthermore, the paper [8] does not clearly explain how the throughput is calculated, given the unpredictability of the TZS termination. Our hardware resource usage is bigger than [8] due to the high level of parallelism of the developed architecture. Our energy-efficient operation point of our architecture.

VII. CONCLUSION

This paper presented a high-throughput energy and rate-aware hardware design for the HEVC ME. The hardware development employs a complexity-reduction strategy based on evaluations performed in the HEVC Reference Software. This design supports the processing of 8x8, 16x16, 32x32, and 64x64 Prediction Unit (PU) sizes with some ME constraints. Optimal rate-energy operations points, based on Pareto Efficiency, allow the selection of a subset of these four PU sizes according to the energy and rate targets using an external control unit. The synthesis results for ASIC show that the developed architecture can process at least 53fps considering UHD 4320p videos when operating at the maximum frequency. Considering the average-case of processing, the architecture is able to process 112fps at UHD 4320p resolution. By operating in 579.69MHz, the ME architecture can process UHD 4320p videos at 30fps, with a power dissipation of 426.91mW.

ACKNOWLEDGEMENTS

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

REFERENCES