UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

An Energy-Efficient Hardware Design for the 3D-HEVC Motion Estimation Adopting Reuse Strategies for Data and Operations

Murilo Roschildt Perleberg

Pelotas, 2020

Murilo Roschildt Perleberg

An Energy-Efficient Hardware Design for the 3D-HEVC Motion Estimation Adopting Reuse Strategies for Data and Operations

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Advisor: Prof. Dr. Marcelo Schiavon Porto

Coadvisors: Prof. Dr. Luciano Volcan Agostini

Prof. Dr. Vladimir Afonso

Pelotas, 2020

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

P426a Perleberg, Murilo Roschildt

An energy-efficient hardware design for the 3D-HEVC motion estimation adopting reuse strategies for data and operations / Murilo Roschildt Perleberg ; Marcelo Schiavon Porto, orientador ; Luciano Volcan Agostini, Vladimir Afonso, coorientadores. — Pelotas, 2020.

88 f. : il.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2020.

1. Hardware design. 2. Motion estimation. 3. 3D-HEVC. 4. Operations reuse. 5. Data reuse. I. Porto, Marcelo Schiavon, orient. II. Agostini, Luciano Volcan, coorient. III. Afonso, Vladimir, coorient. IV. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Murilo Roschildt Perleberg

An Energy-Efficient Hardware Design for the 3D-HEVC Motion Estimation Adopting Reuse Strategies for Data and Operations

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 6 de abril de 2020

AGRADECIMENTOS

Agradeço aos meus pais, Marilene Roschildt Perleberg e Almiro Perleberg, e também aos demais familiares, por toda a motivação para que eu mantivesse o foco nos estudos após a graduação, além de sempre me apoiaram e acreditaram no meu sucesso.

Agradeço imensamente a minha namorada e companheira, Analice Megiato da Silva, por estar comigo tanto nos momentos felizes como também por ter me aguentado nos diversos momentos conturbados do mestrado.

Agradeço aos meus orientadores, Marcelo Porto, Luciano Agostini, Vladimir Afonso, e também ao professor Bruno Zatt, por todo o auxilio e idéias compartilhadas que possibilitaram o desenvolvimento deste trabalho.

Ainda, agradeço aos diversos amigos e colegas de grupo, tanto do ViTech como do GACI, por toda a ajuda e conhecimento compartilhado, além dos diversos momentos de descontração que guardo na memória. Vocês são fera piazada! #12

Follow the party.

— Gaucho Proverb

ABSTRACT

PERLEBERG, Murilo Roschildt. **An Energy-Efficient Hardware Design for the 3D-HEVC Motion Estimation Adopting Reuse Strategies for Data and Operations**. Advisor: Marcelo Schiavon Porto. 2020. 88 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2020.

Currently, there is a growing demand for video streaming through the internet, and also a crescent number of portable devices capable of capture and reproduce those videos. Moreover, the 3D videos allow an improved user experience when compared with traditional videos, since in the 3D videos the scene is simultaneously captured from different points of view. However, due to the amount of data required to represent digital videos, compression techniques are mandatory, which are a series of tools responsible for reducing the redundancies present in video data. In the 3D-High Efficiency Video Coding (3D-HEVC) standard, the most complex tool is the Motion Estimation (ME), while it is also responsible for a huge part of the compression efficiency of this standard. The ME is divided in Integer ME (IME), which performs the comparison a block from the frame being encoded with several candidate blocks from already encoded frames, searching for the candidate block most similar with the block being encoded, and the Fractional ME (FME), which performs a refinement around the candidate result of the IME. By default, the 3D-HEVC adopts the Test Zone Search (TZS) algorithm to select the candidates to be evaluated in the IME, since the TZS evaluates a reduced number of candidates without result in huge losses in image quality when compared with a full search algorithm, which compares all possible candidate blocks. Also, in 3D-HEVC the IME was applied in up to 24 different block sizes. This implies several redundant operations, where parts from a specific candidate can be compared with part of the block being encoded several times, besides a huge memory communication to perform the processing of several block sizes of each candidate block. Aiming at reducing these operation redundancies, it is possible to reuse the operations performed to small block sizes to compose the result of higher block sizes. This strategy also allows data reuse, since it reduces the memory access needed to obtain the samples to process all block sizes. There have only a few works on literature proposing hardware architectures for the IME of the 3D-HEVC standard. Between the IME works of other video coding standards, only a few presents solutions considering data and operations reuse strategies and a fast algorithm as TZS. Therefore, this work presents an IME hardware design adopting the TZS algorithm, with support to all block sizes supported by 3D-HEVC standard and with operations and data reuse strategies to take advantage of already processed operations and reduce the memory communication. The 3D-HEVC IME algorithm was modified aiming at an efficient hardware implementation, and the evaluations show that these modifications present an increase of 9.016% in the BD-rate metric. The developed IME architecture was synthesized for an ASIC with TSMC 40nm standard cells technology, and the results show that the hardware requires 269 K gates, while dissipates 108.48 mW when processing 3 views from different cameras, where each view is composed by the video of the two channels (Texture and Depth Maps) with FHD 1920x1080p resolution with 30 frames per second. The synthesis results have also indicated that the IME hardware design can process up to 3 views with UHD 3840x2160p resolution with 60 frames per second. Moreover, an FME architecture was also presented, which is able to evaluate all possible 48 fractional blocks around the IME result.

Keywords: Hardware Design. Motion Estimation. 3D-HEVC. Operations Reuse. Data Reuse.

RESUMO

PERLEBERG, Murilo Roschildt. **Uma arquitetura de Hardware Energéticamente Eficiente para a Estimação de Movimento do padrão 3D-HEVC Adotando Estratégias de Reúso de Dados e de Operações**. Advisor: Marcelo Schiavon Porto. 2020. 88 f. Dissertation (Masters in Computer Science) – Technology Development Center, Federal University of Pelotas, Pelotas, 2020.

Atualmente existe uma grande demanda por *streaming* de vídeos digitais através da internet, além de um crescente número de dispositivos móveis capazes de gravar e reproduzir estes vídeos. Além disso, vídeos em 3 Dimensões (3D) permitem ainda uma experiência maior do usuário se comparado com os videos tradicionais, visto que nos videos 3D a cena é capturada de pontos de vista diferentes. Contudo, devido a grande quantidade de dados necessários para representar os vídeos digitais, técnicas de compressão se tornam obrigatórias, as quais são uma série de ferramentas responsáveis por reduzir as redundâncias presentes nos dados dos vídeos. No padrão *3D-High Efficiency Video Coding* (3D-HEVC), a etapa mais complexa é a Estimação de Movimento (ME), a qual é também a etapa responsável por grande parte da eficiência de compressão deste padrão. A ME é divida em Estimação de Movimento Inteira (IME), a qual realiza a comparação de um bloco do quadro que está sendo codificado com diversos blocos candidatos de quadros já codificados em busca do bloco candidato mais similar ao bloco sendo codificado, e a Estimação de Movimento Fracionária (FME), a qual realiza um refinamento sobre o candidato resultante da IME. Por padrão, o 3D-HEVC utiliza o algoritmo *Test Zone Search* (TZS) para escolher os candidatos a serem avaliados pela IME, visto que o TZS realiza a avaliação de um número reduzido de candidatos sem resultar em grandes perdas na qualidade da imagem quando comparado com o algoritmo de busca completa, que avalia todos os blocos possíveis. Além disso, no 3D-HEVC a ME pode ser aplicada sobre blocos de até 24 diferentes tamanhos. Isso implica na ocorrência de diversas operações redundantes, onde uma parte de um dos blocos candidatos pode ser comparada com uma parte do bloco sendo codificado inúmeras vezes, além de uma enorme comunicação com a memória para realizar o processamento dos diferentes tamanhos de bloco de cada bloco candidato. Visando a redução das operações redundantes, é possível reutilizar as operações realizadas em blocos pequenos para compor o resultado dos blocos maiores. Esta estratégia também permite o reuso de dados, visto que será reduzindo o número de acessos a memória necessários para obter as amostras de todos os tamanhos de bloco. Existem apenas poucos trabalhos na literatura propondo arquiteturas de *hardware* para a IME do padrão 3D-HEVC. Dos trabalhos de IME para outros padrões de codificação, apenas poucos apresentam soluções considerando estratégias de reuso de dados e operações e um algoritmo rápido como o TZS. Portanto, este trabalho apresenta uma arquitetura de IME adotando o algoritmo TZS, com suporte a todos os tamanhos de bloco suportados pelo 3D-HEVC e utilizando estratégias de reuso de operações para obter vantagem das operações já processados. O algoritmo de IME do 3D-HEVC foi alterado visando uma implementação de *hardware* eficiente, e experimentos mostraram que essas modificações apresentam um aumento de 9,016% na métrica BD-rate. A arquitetura de IME desenvolvida foi sintetizada para ASIC utilizando a biblioteca de células padrão de 40nm da TSMC, e os resultados mostraram que a arquitetura requer 269 K gates, enquanto dissipa 108,48 mW quando processa 3 vistas de diferentes câmeras, sendo cada vista é composta pelo video dos 2 canais (textura e mapas de profundidade) com resolução de FHD 1920x1080p com 30 quadros por segundo. Os resultados de síntese também mostraram que a arquitetura de IME é capaz de processar até 3 vistas com resolução UHD 3840x2160p e com uma taxa de mostragem de 60 quadros por segundo. Além disso, uma arquitetura de FME também é apresentada, capaz de avaliar todos os 48 possiveis blocos fracionarios ao redor do resultado da IME.

Keywords: Arquitetura de Hardware. Estimação de Movimento. 3D-HEVC. Reuso de Operações. Reuso de Dados.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

3D	Tree Dimension
3D-HEVC	3D-High Efficiency Video Coding
AU	Access Units
AMVP	Advanced Motion Vector Prediction
BV	Base View
BD	Bjontegaard Difference
BMA	Block Matching Algorithm
CTU	Coding Tree Unit
CU	Coding Unit
CTC	Common Test Conditions
DV	Dependent Views
DIBR	Depth-Image-Based Rendering
DE	Disparity Estimation
FSM	Finite State Machine
FCO	Flexible Coding Order
FME	Fractional Motion Estimation
fps	frames per second
FHD	Full High-Definition
FS	Full Search
HEVC	High Efficiency Video Coding
IME	Integer Motion Estimation
JCT-VC	Joint Collaborative Team on Video Coding
JCT-3V	Joint Collaborative Team on Video Coding Extension Development
MSE	Mean Squared Error
MSB	Most Significant Bit
ME	Motion Estimation

MPEG	Moving Picture Experts Group
MVD	Multiview Video plus Depth
NCO	Normal Coding Order
PU	Prediction Unit
QP	Quantization Parameter
RD	Rate-Distortion
SAO	Sample Adaptive Offset
SA	Search Area
SR	Search Range
SAD	Sum of Absolute Differences
SATD	Sum of Absolute Transformed Differences
TZS	Test Zone Search
UHD	Ultra High-Definition
VCEG	Video Coding Experts Group

CONTENTS

1 INTRODUCTION

Nowadays, the Three Dimension (3D) contents given by virtual reality glasses, video games, and movies, allows an increase in the experience of the spectator when compared with traditional 2D videos. Considering high-resolution 3D videos, the huge amount of data needed to represent them requires the employment of several video coding techniques. Even so, the applications and devices with support for digital videos usually require dedicated hardware to implement the compression techniques and obtain real-time processing.

The compression techniques available to be applied for video encoding are defined by a video coding standard. The encoding tools apply those compression techniques, which exploit some kind of redundancy present in the video data, thus reducing the amount of data required to store it. Among the current video coding standards, the 3D-High Efficiency Video Coding (3D-HEVC) (ITU-T, 2013) was developed by the experts of the Joint Collaborative Team on Video Coding Extension Development (JCT-3V) in 2015 (HEVC, 2020) to be a 3D extension of the High Efficiency Video Coding (HEVC) standard (ITU-T, 2013). The HEVC has emerged in 2013 (JCT-VC, 2020a) being developed by the experts of the Joint Collaborative Team on Video Coding (JCT-VC), aiming to obtain high efficiency in the compression while maintaining high image quality at the cost of using high complexity algorithms.

The 3D-HEVC was developed to efficiently encode the characteristics of 3D videos. For that, the 3D-HEVC maintains the same encoding tools of HEVC to encode each video sequence (TECH et al., 2016) (JCT-VC, 2020a) (JCT-3V, 2020), while it introduces several new encoding tools and adopts the Multiview Video plus Depth (MVD) format (TECH et al., 2016) to enhance the coding efficiency. In the MVD format, several views can be used to represent the same video scene (see Section 2.2.1). Each view was captured from a different point of view of the scene, and each one is composed of two channels, which are the texture and the depth map. These two channels are captured and encoded for each view while a Depth-Image-Based Rendering (DIBR) process (FEHN, 2004) can generate intermediary texture views (which were not captured by any camera) by using the information of the two channels at the decoder side.

The DIBR process allows the capture, processing and transmission of a reduced number of views on the encoder side, aiming at obtaining a higher number of views at the decoder side (see Section 2.2.1).

To encode each frame (or channel) of a view, the 3D-HEVC adopts the same partitioning scheme used by the HEVC standard (TECH et al., 2016). This scheme divides each frame into square-shaped blocks called Coding Tree Unit (CTU), which has the maximum size of $64x64$ samples. Then, each CTU is divided into one or more square-shaped blocks known as Coding Unit (CU). Finally, each CU is divided into Prediction Unit (PU), which size can vary from $4x4$ to $64x64$ samples. The PUs are the units evaluated by the prediction steps of the video encoders. These divisions occur based on the result of the encoding tool over a specific PU size, aiming to obtain the best encoding efficiency based on the Rate-Distortion (RD) cost (JCT-3V, 2020) calculation. After calculating the RD cost for different PU sizes, the CTU is encoded using the different PU sizes and prediction modes that present the smaller RD cost (consequently, the one that requires fewer bits to be encoded with minimum image distortion).

Most of the computational complexity of video encoders is to evaluate all possible PU sizes. The 3D-HEVC has a total of 24 possible PU sizes to be evaluated in the Motion Estimation (ME) tool, while only seven PU sizes were evaluated in the H.264/AVC (WIEGAND et al., 2003). This significant amount of PU sizes supported by 3D-HEVC was responsible for a huge part of prediction tools complexity. Therefore, it results that the ME tool is the most complex prediction tool of the 3D-HEVC standard, while it was also the prediction tool more present in a video encoding (see Section 2.4).

The ME is applied to each PU, and it allows the PU representation by using the information of a block from a reference frame. For that, the ME searches in the reference frame a block most similar to the PU being encoded, being this search performed into two stages. The first stage applied to each PU is the Integer Motion Estimation (IME), which is responsible for applying a Block Matching Algorithm (BMA) to defines several candidate blocks from reference frames to be evaluated. After this evaluation, the most similar candidate block among the comparisons is determined. The second stage is the Fractional Motion Estimation (FME), which is applied over the candidate determined by the IME to refine that block, improving the encoding efficiency.

There are several works in the literature proposing dedicated hardware architectures for one of the two ME stage, disregard the other stage (MEDHAT; SHALABY; SAYED, 2015) (FAN et al., 2018) (LIAO; SHEN; TSENG, 2019) (GU et al., 2019) (AFONSO et al., 2016a). Also, there are only a few works proposing hardware architectures for both ME steps (PERLEBERG et al., 2018) (XU et al., 2018) (PASTUSZAK; TROCHIMIUK, 2016). However, considering the ME of the 3D-HEVC context, only AFONSO et al. (2019) presents an architecture that encapsulates ME and Disparity Estimation (DE) tool. This DE tool shares several similarities with the ME tool (see

Section 2.2.2.2). Thus, this work proposes a hardware solution for the ME tool. It presents several hardware-friendly simplifications in the IME stage which allow obtaining a low-power and high-throughput architecture capable of dealing with all PU sizes supported by 3D-HEVC.

Since the FME performs a refinement after the IME processing, the RD cost process depends on the FME processing to decide between the several PU sizes and prediction modes supported by 3D-HEVC. Therefore, this work also presents a hardware architecture for the FME capable of processing all PU sizes supported by 3D-HEVC. Although, the FME presented is a primary architecture that will be optimized in future works. The FME was implemented only to explore all potential of the developed IME architecture considering the encoding efficiency, since only the IME cannot explore all features supported for the ME tool of 3D-HEVC, thus requiring the FME to explore the fractional candidates and reaches the maximum encoding efficiency.

There are three main contributions of this Master's dissertation work:

- The development of energy-efficient hardware design for the IME tool of the 3D-HEVC encoding standard was presented. The architecture was developed to process all PU sizes supported on 3D-HEVC standard in parallel, thus the developed IME architecture has reached a high throughput, being capable of processing up to 5 views of Ultra High-Definition (UHD) 2160p videos at 60 frames per second.
- It presents a strategy that allows the reuse of operations and data already processed. This strategy allows an efficient evaluation of the candidate blocks considering all PU sizes since only the smallest PU size must be fully processed, and its results are used to compose the processing of all other higher PU sizes.
- Presents a Test Zone Search (TZS) Master-Slave algorithm among all PU sizes, which performs that all PU sizes will evaluate the same candidates for a better exploration of the reuse strategies. In this algorithm, the TZS for $64x64$ PUs was applied firstly, and then its decisions of which candidates should be evaluated is adopted for all smaller PU sizes.

The organization of this work is given as follows: The chapter 2 explains the major concepts of the 3D-HEVC video coding standard, focused on the operation of ME, presenting all content needed to understand this work. After, the chapter 3 shows the evaluations performed to define the developed architecture, while the chapter 4 shows in detail the developed architecture. At chapter 5, the obtained results are presented and compared with related work results. Finally, the chapter 6 shows the conclusions of this work.

2 BASIC CONCEPTS AND MOTIVATION

This chapter presents the basic concepts of the contextualization of this work, including the basic concepts of digital videos, the Tree Dimension (3D) video encoding process, and the algorithm of encoding tools. This chapter also presents the complexity analysis that justifies the development of an Integer Motion Estimation (IME) architecture due to its computational complexity and its importance between the other encoding tools of 3D-High Efficiency Video Coding (3D-HEVC), the related works proposing hardware architectures for the ME tool and the 3D-HEVC, and also the main motivations of the adopted strategy among the number of operations and memory requirements of the IME tool.

2.1 Digital Videos

A digital video is a sequence of independent images that were captured with a determinate time interval between each image. If these images are viewed with an adequate rate of frames per second, they provide motion sensations to the spectator. A small video sequence can be seen in Figure 1, which presents a set of images that composes a digital video.

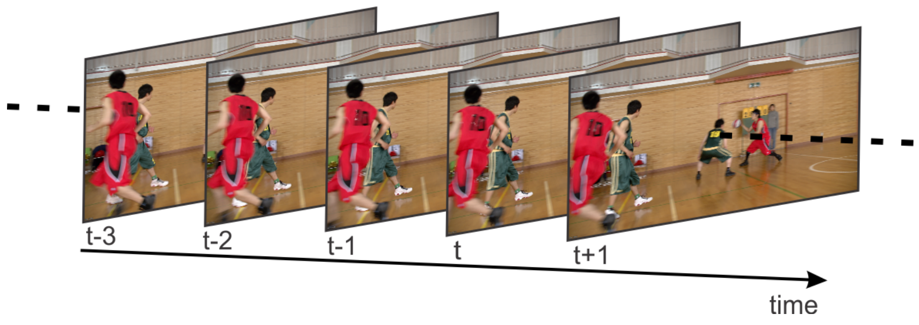

Figure 1 – Sequence of frames that compose a digital video

In a digital video, each one of the images is called a frame, and each frame is composed of a determinate number of pixels. The pixels are the smaller elements of a digital video, being generally composed of a combination of three samples. These three samples represent the luminance and chrominance information and are used to compose the color seen by the spectator (MIANO, 1999).

There are different metrics to represent the amount of information present in a digital video. One of them is the spatial resolution, which represents the number of pixels in the height and width of each frame of the video. Since each frame is composed of a determinate number of pixels, the more pixels a frame has, the more information can be represented in a frame, which can result in better video quality due to the increase of details of each frame (PORTO, 2012).

Another metric to classify a digital video is the temporal resolution, which measures the rate that the frames were captured and/or presented. To obtain a motion sensation, a minimum of 24 to 30 frames per second (fps) must be adopted, and higher values result in less sudden movements of the objects seen by the spectator. Therefore, the visual quality can be related to fps, since higher frame rate indicates more information that can be presented in one second.

To increase the user experience while the spectators watch a video, stereo Tree Dimension (3D) contents emerged bringing the viewing of the scene from two or more different points of view (KAUFF et al., 2007) (TECH et al., 2016). For this, two or more camera devices were adopted to capture the scene, which can later be explored in different ways, as using one view for each eye of the spectator or even given the desired angle selection (TOSHIBA, 2020).

However, there is a huge amount of redundancy in video data, since that in normal situations spatial neighbor samples from a frame could be equal or very similar, and since two temporal neighbor frames could share its information, differing only according to the movement of the scene (AGOSTINI, 2007). Moreover, the 3D contents have even more redundancies considering that the cameras will capture the same objects from the scene (TECH et al., 2016). Therefore, video compression techniques must be adopted to efficiently exploit these redundancies and reduce the amount of data required to represent a video.

Different video encoding tools can be used to encode digital videos. A video encoding standard defines a set of available tools. Therefore, the next sections explain in detail the operation of the available encoding tools for the video coding standard addressed in this work.

2.2 3D-High Efficiency Video Coding

Among the current video coding standards, the High Efficiency Video Coding (HEVC) was emerged in 2013 by the Joint Collaborative Team on Video Coding (JCT-VC), a joint of experts from the ISO/IEC Moving Picture Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG) organizations (JCT-VC, 2020b). The JCT-

VC was created in 2010 aiming to develop a new video coding standard able to reach half the video file size when compared with the previous state-of-the-art standard, the H.264/AVC (WIEGAND et al., 2003) while maintaining the visual quality.

To encode 3D contents, the 3D-HEVC emerged in 2015 by the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V), a joint of experts from the same organizations of the HEVC, the MPEG and VCEG. The 3D-HEVC was created as a 3D extension for the HEVC. Therefore, it shared several encoding tools from HEVC, while it adopts some new techniques to efficiently encode the characteristics present on 3D videos.

The next subsections present the characteristics and features of the 3D-HEVC, the encoding process of the 3D-HEVC, and also the coding tool covered in this work.

2.2.1 3D-HEVC Characteristics and Features

As mentioned before, digital videos are a sequence of frames captured in sequence, traditionally by an independent camera. The 3D-HEVC allows the efficient encoding of 3D videos that uses the MultiView Video plus Depth (MVD) format (TECH et al., 2016) (KAUFF et al., 2007).

In the MVD format, different views are used to represent a scene, being each view captured in parallel by different cameras (KAUFF et al., 2007). Moreover, in the MVD format, each view is composed of the Texture channel (Normal image captured from each view) and also a Depth Map. The Depth Map represents the distance between the camera and each object captured from the scene from a specific point of view. The MVD format is represented in Figure 2. The Figure presents the Texture (Colored image) and also the Depth Map represented in Gray-scale, where the clearest samples represent the objects closest to the camera. Moreover, Figure 2 presents three different views captured of the same scene, being each one composed of both Texture and by the Depth Map. Also, it should be noted that each of those three views was captured from a different angle of the scene.

From the two captured channels, only the Texture is presented to the spectator, while the Depth Maps were used at the decoder side by the Depth-Image-Based Rendering (DIBR) process (FEHN, 2004). The DIBR process allows generating intermediary Texture views between two views, as represented in Figure 3. Briefly, based on the depth map of two original views, the DIBR process knows if an object was visible or occluded in the intermediary point of view, so deciding the texture of which view should be used to compose the texture of the intermediary view (FEHN, 2004).

Therefore, the MVD format and DIBR process allow reducing the amount of data manipulated by the encoder and consequently transmitted into the bitstream, since by using it, a reduced number of views can be adopted at the encoder side, while obtaining several intermediary views at the decoder side.

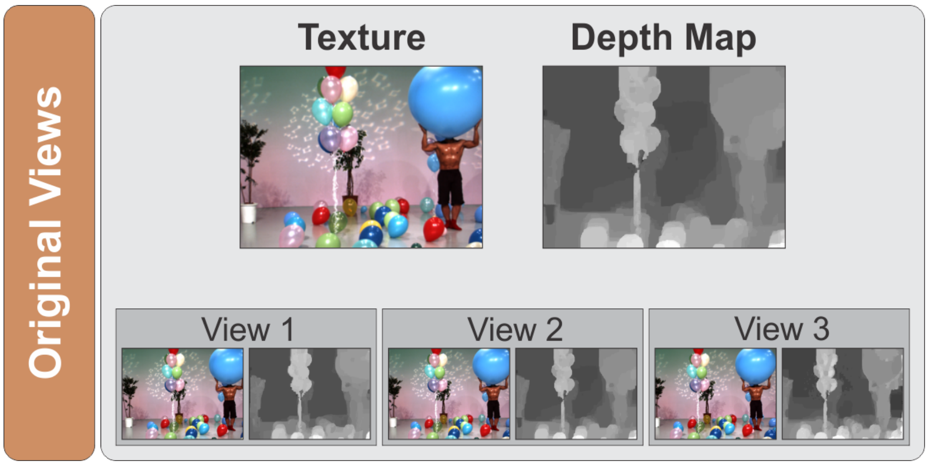

Figure 2 – MVD Format where different views can be used to represent the same scene

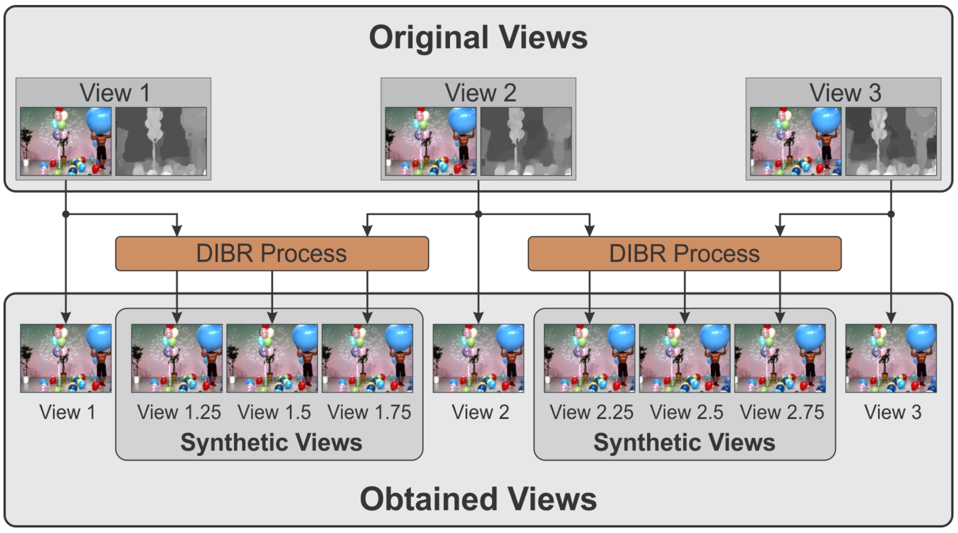

Figure 3 – DIBR process to generate intermediary texture views

To encode the Texture and Depth Maps channels from all views, the 3D-HEVC adopts sets of Access Units (AU) (TECH et al., 2016). Each AU encapsulates the information from all views that belongs to the same time instant (TECH et al., 2016). An encoding sequence of several AUs is presented in Figure 4, where the red arrows represent the encoding order of the two channels from each view. As can be seen, only after finish the encoding of one AU the next AU were encoded.

From Figure 4, it can also be seen that the first view encoded from each AU is called Base View (BV) since it is encoded independently of the information of neighboring views in the same AU. The other views of each AU are called Dependent Views (DV) since they can explore the information of already encoded views from the same AU.

The experiments of this work consider two coding orders to be adopted on DV from each AU, being the Normal Coding Order (NCO), adopted by default by 3D-HEVC (JCT-3V, 2020), and also the Flexible Coding Order (FCO) (GOPALAKRISHNA; HANNUKSELA; GABBOUJ, 2013). These two encoding orders are represented in Figure

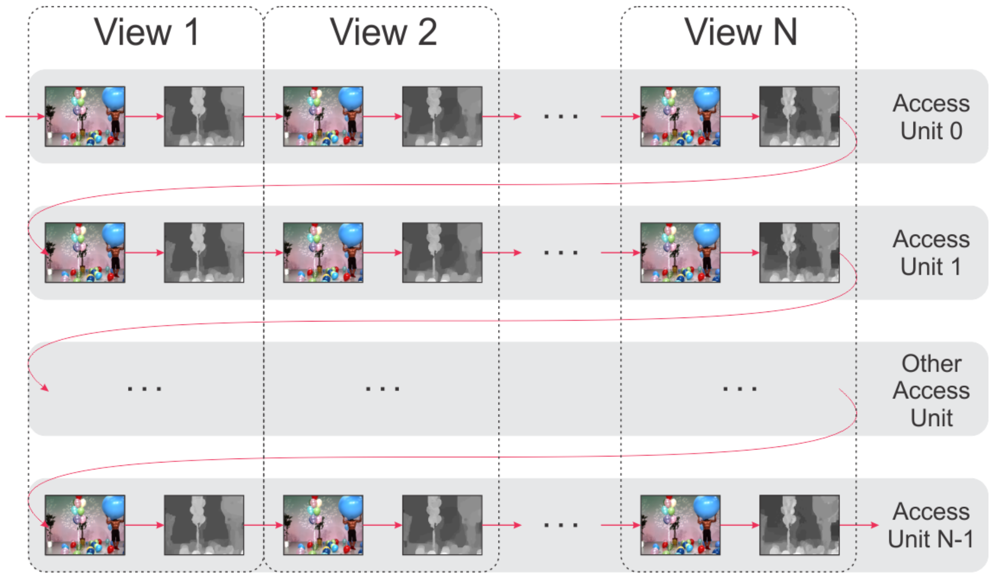

Figure 4 – Sequence of Access Units to be encoded
Source: Adapted from AFONSO (2019).

5 and Figure 6, respectively, where the green arrows indicate the order in which the channels will be encoded. While in the NCO the Texture channel from DV is encoded firstly than its respective Depth Map, in the FCO the Depth Map is encoded firstly than its respective Texture channel.

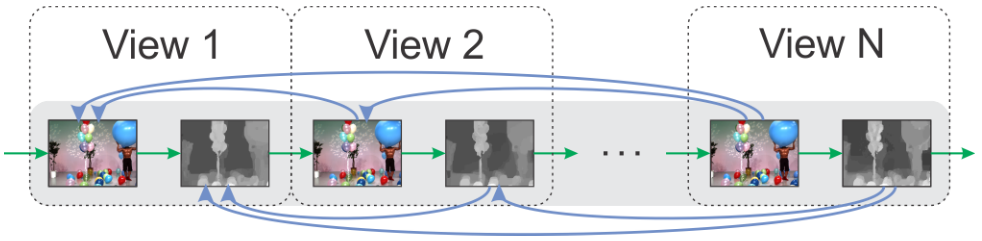

Figure 5 – Normal Coding Orders

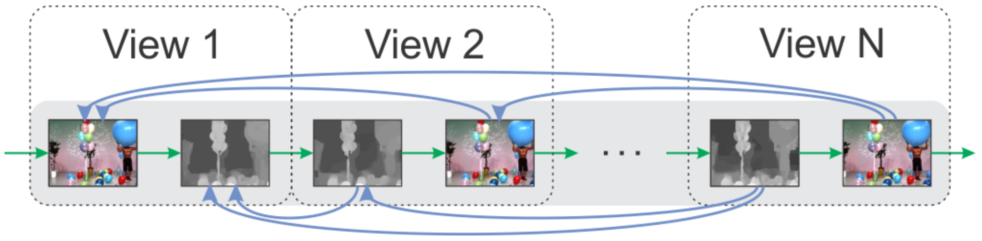

Figure 6 – Flexible Coding Orders

As mentioned before, the DV channels can be encoded by using the information of other views. The Blue arrows in Figures 5 and 6 depict the dependency between different frames of each AU, by representing which neighboring frames are available to encode each frame. This information to be explored in other views is based on the disparity information of the scene. In the NCO, the disparity information is derived from motion information of neighboring blocks of the block being encoded. Although, by encoding the depth map before the texture in the DV, the texture can be encoded by using the disparity estimation precisely obtained by the depth map information, thus improving the encoding efficiency according to GOPALAKRISHNA; HANNUKSELA; GABBOUJ (2013).

2.2.2 3D-HEVC Encoding Process

The 3D-HEVC is a block-based hybrid video coding standard, where each frame to be encoded is divided into blocks of samples. To be encoded, each of those blocks of samples passes through a series of encoding tools, as represented in Figure 7, which presents the 3D-HEVC encoding flow and its main tools.

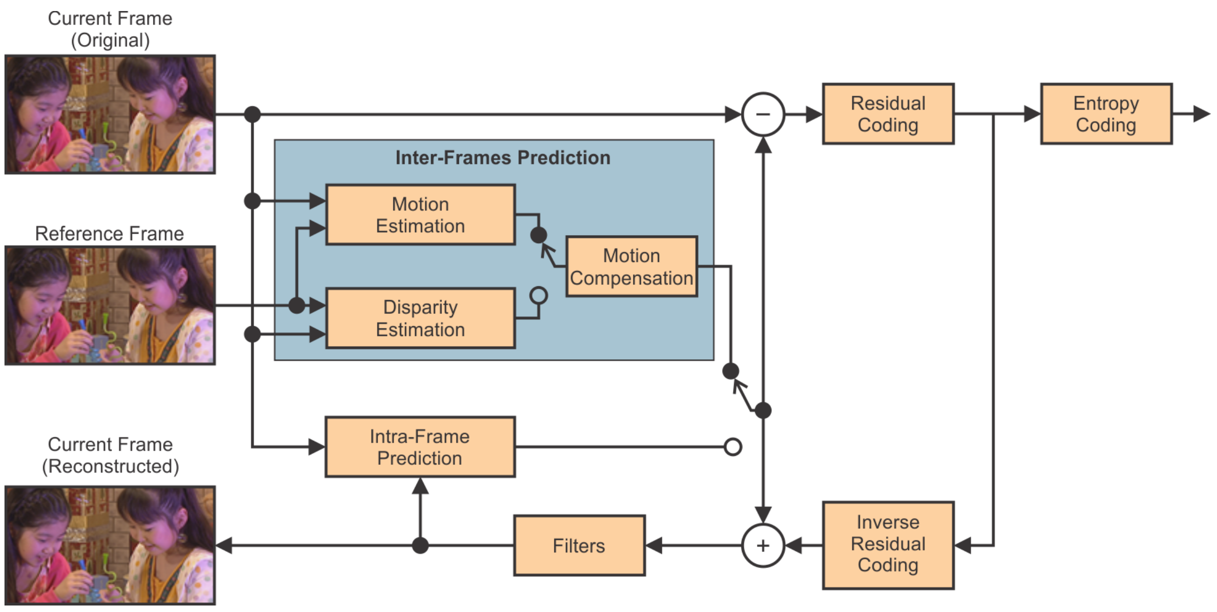

Figure 7 – Encoding flow of 3D-HEVC standard

As can be seen in Figure 7, each frame is partitioned in several small blocks to be encoded and evaluated by the next encoding tools. The next tools used to encode each block are the prediction tools, residual coding, and entropy coding, which will be explained in the next subsections. The inverse residual coding and the filters were used to generate the reconstructed frame, which was required by the prediction tools.

2.2.2.1 Partitioning scheme

The 3D-HEVC inherits the partitioning scheme used by the HEVC standard to encode each frame (TECH et al., 2016). This scheme divides each frame into smaller square-shaped blocks called Coding Tree Unit (CTU) (SZE; BUDAGAVI; SULLIVAN, 2014), the basic processing unit of 3D-HEVC. By default, the CTUs are defined with its maximum size of $64x64$ samples (JCT-3V, 2020).

After, each CTU is divided into one or four square-shaped units called Coding Unit

(CU). If it splits into four CUs, then each CU can be further divided into one or four CUs (SZE; BUDAGAVI; SULLIVAN, 2014) (JCT-3V, 2020). This split process continues until the CU reaches the $8x8$ block size. Therefore, the CUs can assume the size of $64x64$, $32x32$, $16x16$, or $8x8$ samples. Typically, higher CUs were adopted for homogeneous regions of the CTU, while smaller CUs were adopted for regions with several details. Figure 8 presents an example of the splitting process of one CTU in several CUs, where the numbers indicate the coding order of the CUs.

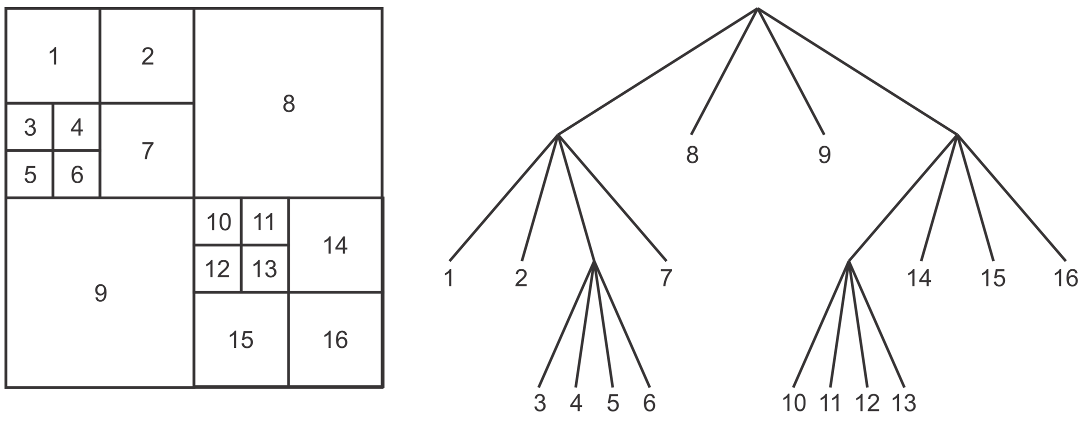

Figure 8 – Example of the splitting of one CTU
Source: Adapted from SZE; BUDAGAVI; SULLIVAN (2014).

Finally, each CU is further divided into smaller blocks called Prediction Unit (PU), where the prediction tool signaled by its respective CU is evaluated (SZE; BUDAGAVI; SULLIVAN, 2014). These PUs can have three different shapes according to width and height size of PU: They can be Square, where the width equals to the height; Symmetric, where one size is half of another; or Asymmetric, where one size is a quarter of the other. To reach the best efficiency, the 3D-HEVC reference software (3D-HTM) (JCT-3V, 2020) starts evaluating the higher PU sizes. Based on the trade-off between compression and image quality, the encoded decides if the smaller PU sizes will be evaluated (SZE; BUDAGAVI; SULLIVAN, 2014).

2.2.2.2 Prediction Tools

The Prediction tools are adopted to represent the current PU being processed using the information of previous processed blocks (AGOSTINI, 2007) (PORTO, 2012). Therefore, the prediction output is the predicted block, generated using the samples from already processed blocks. There is two class of prediction tools, being the intra-frame prediction and inter-frames prediction (AGOSTINI, 2007).

The intra-frame prediction tools are responsible for reducing the spatial redundancy individually present on each frame (AGOSTINI, 2007). It evaluates several different ways to represent the current PU using the information of neighboring already processed blocks of the same frame (TECH et al., 2016) (JCT-3V, 2020) (ITU-T, 2013).

The inter-frames prediction tools are responsible for reducing the temporal redundancy between different frames. These tools search for similar information in two different frames, thus representing the PUs from the frame being encoded using the information of frames previously processed (AGOSTINI, 2007). The 3D-HEVC has two main inter-prediction modes, being the Motion Estimation (ME) and the Disparity Estimation (DE) (TECH et al., 2016).

The ME is used to represent the current PU using information from temporal neighboring frames (PORTO, 2012). Its processing is represented in Figure 9. The ME searches for redundancies of the current PU in candidate blocks from a reference frame, by searching the candidate block most similar to the current PU. After that candidate block is found, the ME result is the vector that represents the location of that candidate block. After, the obtained vector is passed to the Motion Compensation, which will reconstruct the current PU using the information from the ME result.

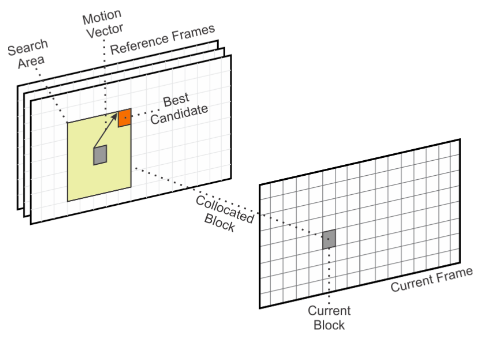

Figure 9 – Motion Estimation Processing
Source: Adapted from PORTO (2012).

The DE has several similarities with the ME tool (JCT-3V, 2020). It also is used to represent the current PU using the information of candidates from other frames. However, in the DE tool, the reference frames belong to a neighboring view that was captured at the same time instant but with another camera, while the reference frame from the ME tool belongs to the same view but from a different time instant (TECH et al., 2016). The DE processing was represented in Figure 10, where can be seen that the reference frame belongs to another view. The vector obtained from the DE tool is also passed to a Disparity Compensation to reconstruct the current PU using the reference frame information.

As it will be presented in section 2.4, the ME is the most complex encoding tool of the 3D-HEVC, while it was the most important tool in the 3D-HEVC due to its impact on the encoding efficiency. Therefore, the ME is the focus of this work, and so it will be presented in detail in the next subsection.

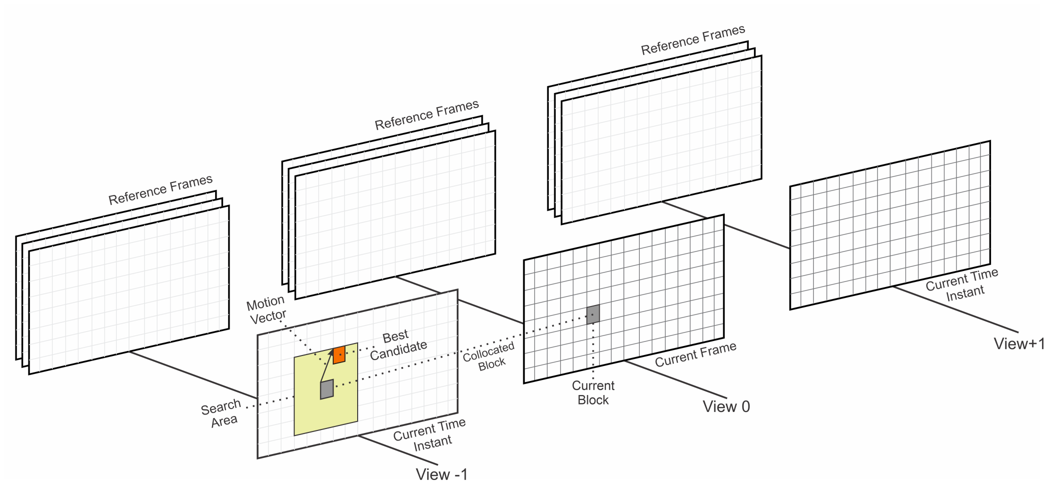

Figure 10 – Disparity Estimation Processing

2.2.2.3 *Residual Coding*

After representing the current PU using information from samples already processed, the 3D-HEVC encoder obtains a residue block, as can be observed in Figure 7. The residue block is the difference between the reconstructed PU and the original PU (AGOSTINI, 2007). This residue block is then encoded by the residual coding.

The residual coding was composed of two tools, being Transform and Quantization (SZE; BUDAGAVI; SULLIVAN, 2014). The first is the Transform tool, which will transform the residual block from the spatial domain to the frequencies domain, generating a transform block that typically has its main information in the top-left region of the block (SZE; BUDAGAVI; SULLIVAN, 2014). After the Transform, the Quantization tool was used to removing the frequencies that were imperceptible to the human eye, so reducing the amount of data required to represent the transform block (SZE; BUDAGAVI; SULLIVAN, 2014). The amount of information to be removed is given by a Quantization Parameter (QP) value.

2.2.2.4 *Entropy Coding*

The last step from the encoding flow is the Entropy coding (TECH et al., 2016) (SZE; BUDAGAVI; SULLIVAN, 2014). This step is used to compress the block output of the Residual coding and also the lateral information required to decode the information (partitioning from each CTU, intra mode adopted, the motion/disparity vector resulting from the prediction tool, among others (SZE; BUDAGAVI; SULLIVAN, 2014)). It uses statistical properties of the data to be encoded to change the data representation so that the symbols that have more probability to appear are represented using a lower number of bits. After the Entropy coding, the resulting bits are ready to be stored and transmitted.

2.2.2.5 Inverse Residual Coding

As mentioned before, the Quantization tool removes the frequencies that were imperceptible to the human eyes, thus resulting in small losses in the image quality. These losses need to be considered in the reference frames used by the prediction tools. Therefore, the current block needs to be decoded. For that, the 3D-HEVC has Inverse Residual coding, which applies an Inverse Quantization and an Inverse Transform that perform the transform of the block from the frequencies domain to the spatial domain (SZE; BUDAGAVI; SULLIVAN, 2014), so obtaining again the residue block, but without the irrelevant frequencies. These residues are summed with the predicted block, thus composing the reconstructed block.

2.2.2.6 In-loop Filters

The 3D-HEVC has two in-loop filters to be applied after obtaining the reconstructed block (TECH et al., 2016) (SZE; BUDAGAVI; SULLIVAN, 2014). These filters were used to improve the visual quality by reducing visual artifacts that can appear in each frame due to the different encoding tools available on 3D-HEVC. While the Deblocking filter is used to remove discontinuities that can occur in different blocks boundaries, the Sample Adaptive Offset (SAO) is used to attenuate ringing artifacts and changes in sample intensity of the frame.

2.3 Motion Estimation Algorithms

As mentioned before, the ME is responsible for representing the current PU using the information from reference frames, by using only the location of a similar block in the reference frame and the difference between the current PU and the most similar block (AGOSTINI, 2007). The 3D-HEVC allows the selection of one between different reference frames to be used to predict the current PU. It also allows the adoption of two reference frames to simultaneously predict a block, being this technique called bi-prediction (TECH et al., 2016) (SZE; BUDAGAVI; SULLIVAN, 2014). In this case, the current block was represented by using the weighted average of the best candidate of each reference frame (SZE; BUDAGAVI; SULLIVAN, 2014).

To search and select the most similar candidate block to represent the current PU, the location of the Search Area (SA) in the reference frames needs to be defined firstly, where the candidate blocks will be selected for evaluation. The SA size is defined according to the Search Range (SR). By default, the 3D-HEVC adopt the SR as pm 64 samples, which represents a SA of 192x192 samples.

On the 3D-HEVC standard the SA can be located centered in the collocated block, or even defined by adopting the Advanced Motion Vector Prediction (AMVP) algorithm (LIN et al., 2013). The AMVP uses heuristics to inherit the motion vector of some neighboring blocks (LIN et al., 2013), as the blocks represented around the current block in Figure 11. In Figure 11, the brown candidates represent spatial neighboring blocks, while the blue candidates represent temporal neighboring blocks. These motion vectors were used to define the direction of the SA location, thus resulting in the probable region to find the most similar candidate.

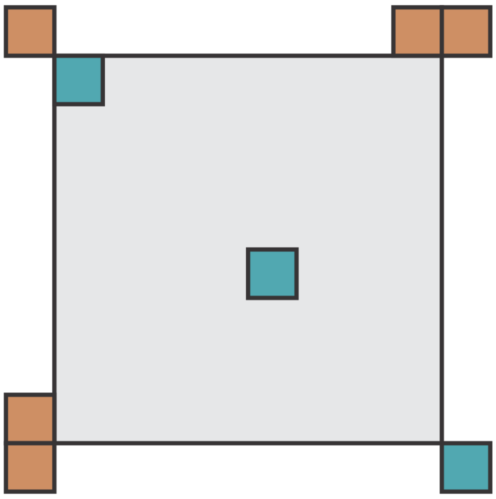

Figure 11 – Neighboring blocks information available for AMVP
Source: Adapted from LIN et al. (2013).

After defining the SA location, the ME step evaluates the similarity between the candidate blocks of the SA and the current block. For that, a similarity criterion must be adopted, which computes the difference level between two blocks. There are different available similarity criterion, as the Mean Squared Error (MSE), the Sum of Absolute Differences (SAD), and the Sum of Absolute Transformed Differences (SATD).

The SAD is the criterion commonly adopted for hardware implementations since it has low complexity and, therefore, can be implemented only with an adder and subtractor operators, while another criterion can require multiplier operators. The Equation (1) represents the SAD value computing, where C, R, x, and y variables represent the current PU sample, candidate block sample, horizontal position, and vertical position, respectively. As can be seen, it obtains the difference between every sample from the current PU to the candidate block sample at the same position. After, it obtains the absolute value of every difference. Finally, it accumulates all absolute values obtained, thus generating the SAD value of the evaluated candidate.

$$SAD = \sum_{y=0}^{PU_{height}} \sum_{x=0}^{PU_{width}} \left| C_{(x,y)} - R_{(x,y)} \right| \tag{1}$$

The SATD is similar to the SAD, but on SATD the differences are transformed before accumulating all values, as given by (2). The W denotes the transformed differences, computed by (3), where $C - R$ represents the matrix of differences between the current and the candidate block. The H and HT of (3) represent the transform matrix its transposed, respectively. An example of a $4x4$ transform matrix can be seen on (4).

$$SATD = \sum_{y=0}^{PU_{height}} \sum_{x=0}^{PU_{width}} \left|W_{(x,y)}\right| \tag{2}$$

$$W = H \cdot (C - R) \cdot H^T \tag{3}$$

$$H = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{4}$$

The selecting of candidate blocks from reference frames to be evaluated is performed with two consecutive steps on 3D-HEVC, being the IME, and the Fractional Motion Estimation (FME) steps (SZE; BUDAGAVI; SULLIVAN, 2014) (JCT-3V, 2020). The IME applies a Block Matching Algorithm (BMA) to define the heuristics of what candidates of the SA will be evaluated, resulting in the location of a candidate block of the SA. The FME was responsible for refining this candidate by generating new blocks at fractional positions around it, so one of those new blocks can be even more similar to the current PU.

To decide between the different possible prediction modes to be used to represent each PU, the 3D-HEVC computes the Rate-Distortion (RD) cost (SZE; BUDAGAVI; SULLIVAN, 2014), which is a value that measures the image distortion of the evaluated prediction mode, and the number of bits required to represent that prediction mode. Due to its high cost, several candidates of ME were evaluated only by using the SAD criterion, and the complete RD-cost was only computed at the end of ME processing.

In the next subsections, two subsections explain in detail the algorithms of the two ME steps.

2.3.1 Integer Motion Estimation

As previously mentioned, there are different BMA that can be adopted to select the candidate blocks to be evaluated. By default, the 3D-HEVC adopts the Test Zone Search (TZS) algorithm (JCT-3V, 2020) (LI et al., 2014), since it drastically reduces the encoding complexity to find the most similar candidate, while maintaining a high compression efficiency when compared with a Full Search (FS) algorithm, which compares all possible blocks from SA. Figure 12 presents the flowchart of the four TZS steps, which was the Prediction, First Search, Raster, and Refinement.

As can be seen in Figure 12, firstly the Predictor and First Search steps were ap-

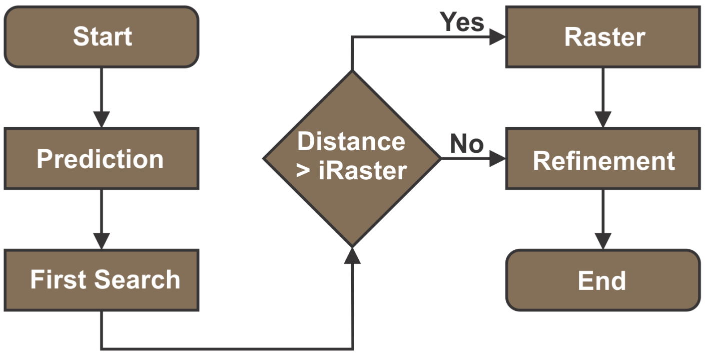

Figure 12 – Flowchart of TZS steps

plied. Then, if the distance between the start point and the result found in these steps is greater than an iRaster constant, the Raster Step is applied (JCT-3V, 2020). If that distance is smaller than iRaster constant, the Raster is avoided so the Refinement is applied. These four steps are performed as follows.

2.3.1.1 Predictor Step

The Predictor if the first step of TZS, used to set the start position for the evaluations performed by the next steps (JCT-3V, 2020). Similarly to the AMVP, the Predictor evaluates the candidates given by movement information of neighboring blocks, so the best evaluated are used as the start point, thus the TZS can converge to the best result quickly, reducing the ME complexity.

The TZS evaluates the candidates related of up to six different Predictors to find the best start point, being that each Predictor evaluates the motion vector of different neighboring blocks (JCT-3V, 2020):

- Predictor Median: Average of the motion vector from neighboring blocks
- Predictor Left: motion vector from the left neighboring block
- Predictor Upper: motion vector from the upper neighboring block
- Predictor Upper-Right: motion vector from the upper-right neighboring block
- Predictor Zero: motion vector from the collocated block
- Predictor 2Nx2N: motion vector from the highest PU of current CTU

2.3.1.2 First Search Step

The First Search adopts an expansive scheme search to selects the candidate blocks of the SA to be evaluated (JCT-3V, 2020). Figure 13 presents the diamond scheme followed by this step, being the start position located in the center of the figure. It should be noted that 3D-HEVC adopts the diamond scheme by default, but a squared scheme can also be used (JCT-3V, 2020). Each colored square represents the first sample of a candidate to be evaluated. The scheme was expansive since it starts evaluating the four candidates with one sample of distance to the start position, and after that, it expands to evaluate more distant candidates, until reaching a stop condition that ends the First Search processing.

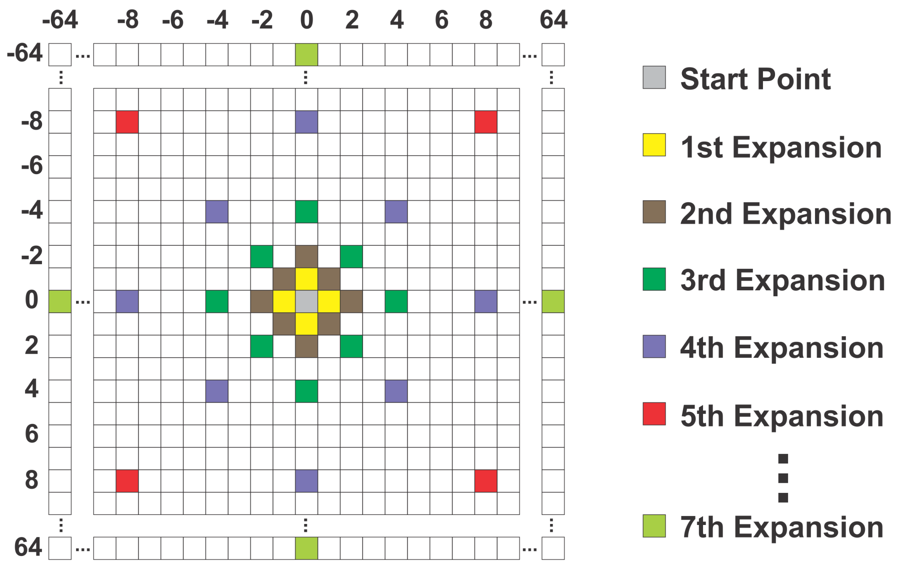

Figure 13 – Expansive diamond scheme adopted by TZS algorithm to selects the candidate blocks for evaluation

The second, third and fourth expansion levels evaluate the eight blocks located at two, four, and eight samples of distance from the start position, respectively, and these expansions were represented by brown, green, and purple colors on Figure 13. In the next expansion levels, 16 candidates were evaluated on each level. These candidates are distant by 16, 32, and 64 samples from the start position (JCT-3V, 2020).

In a default SA of $192x192$ samples, the First Search selects up to 76 candidate blocks for evaluation, where the last expansion performed is distant of 64 samples from the start position. However, the First Search has one stop condition that can end the First Search processing without evaluating all expansions. It occurs if the algorithm expands three consecutive times without finding a best result than the obtained in previous expansions. In both cases, the location of the best candidate evaluated defines the next TZS steps.

2.3.1.3 Raster Step

Briefly, the Raster Step is a sub-sampled Full Search, performed according to the distance of the First Search result to the Predictor result (JCT-3V, 2020). This step assumes that the Predictor Result was not good and thus it has the objective of redirect the search to a new position in the SA where the Refinement Step can find a better result.

Therefore, as mentioned before, the Raster only occurs if this distance is greatest than an iRaster constant. If that distance is smaller, then the Raster is avoided and only the Refinement Step is applied. The 3D-HTM (JCT-3V, 2020) defines the iRaster constant as five.

The sub-sampled FS algorithm of the Raster Step is similar to the FS algorithm. It evaluates the candidates in the whole SA. However, the algorithm is sub-sampled since only one at every iRaster samples is evaluated. Therefore, considering a SR of pm 64 samples, the Raster Step can evaluate 625 candidates.

2.3.1.4 Refinement Step

The Refinement Step is responsible for refining the best result found by the previously performed steps. This refinement performs an iteration with a similar process than the First Search step, i.e., it adopts the same heuristics present in Figure 13 to perform the expansions, which evaluates up to 76 candidates in one iteration (JCT-3V, 2020) considering the default SA of $192x192$ samples.

The conditions of Refinement to stop the expansions were similar to the First Search. However, when a condition occurs, the Refinement starts a new iteration, which was a new Refinement around the best candidate of previous refinement (JCT-3V, 2020). In other words, the Refinement was recursive, since while an iteration can find a better candidate than the result of previous iterations, a new iteration around the best candidate found was started. The TZS only ends when the Refinement performs an iteration without finding a better candidate than the result of the previous iterations. Due to this recursion, the computational effort to the TZS finish and result in the best evaluated candidate is uncertain.

Finally, considering the number of candidates evaluates by each of the four TZS steps, searching the most similar block can require the evaluation of up to 1011 candidates considering the execution of the Raster Step and four Refinement iterations and the adoption of a SA of $192x192$ samples. Then, after the TZS ends its processing, the best candidate evaluated are passed to the next operation, the Fractional Motion Estimation.

2.3.2 Fractional Motion Estimation

The FME performs a refinement in the candidate block result of the IME. This refinement assumes that the movement of the scene is not always limited by integer samples, thus the IME candidate being slightly shifted to a side being even more similar to the current block. For that, the FME was performed in two steps, the Interpolation, and the Evaluation steps.

Interpolation is the step responsible for generating new fractional samples between the IME candidate and its neighbor. It adopts the use of interpolation filters (ITU-T, 2013) to generate new samples at half and quarter positions around the IME candidate. These new samples are represented in Figure 14, which presents the new samples generated around an $8x8$ block size. In Figure 14, the blue squares are the integer samples of the IME candidate block, the gray squares are the samples at half position, and the white squares represent the samples at quarter positions.

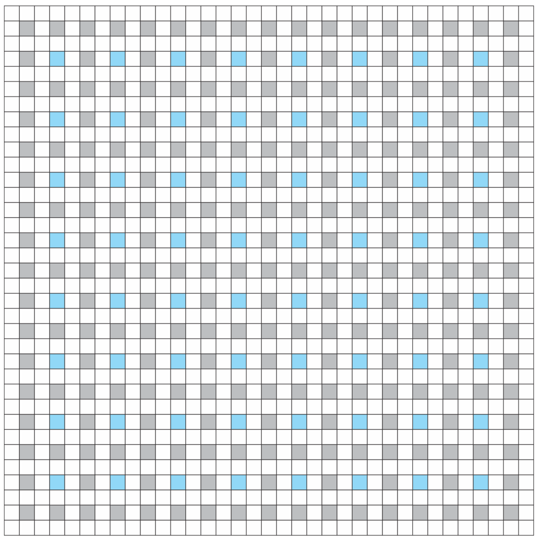

Figure 14 – Fractional samples around a $8x8$ block size. Blue samples represent the integer $8x8$ block, while gray and white samples represent the fractional samples generated at half and quarter position

For generating these new samples, the 3D-HEVC interpolation filters require eight integer samples displaced horizontal or vertical according to the target fractional position, as represented in Figure 15. In Figure 15-(a), the eight integer yellow samples are used to interpolate the three green fractional samples, while in Figure 15-(b), the eight integer yellow samples are used to interpolate the three red fractional samples. It should be noted that generating the samples next to the block limits requires up to four integer samples located outside of the input block (AFONSO et al., 2016b).

The Evaluation step is responsible for grouping these new interpolated samples to generate the new fractional blocks, and after evaluating these new blocks against the current block. In the total, 48 new fractional blocks can be generated around the IME candidate (JCT-3V, 2020), as represented in Figure 16-(a), where the samples 1 to 48 represent the first sample of each one of those 48 new fractional blocks.

For evaluating these fractional blocks, the 3D-HTM (JCT-3V, 2020) adopts the same algorithm as the HEVC reference software (JCT-VC, 2020a), where only 16 of the 48 blocks were generated and evaluated. Firstly, only the half sample positions were interpolated, thus generating eight half blocks, which was evaluated against the current block. After, the best block evaluated (Including the original IME candidate) defines the

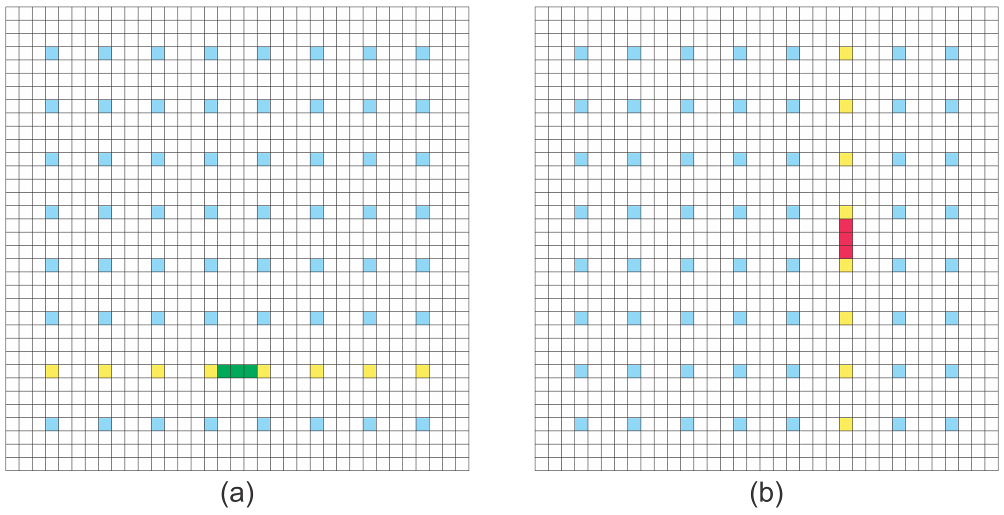

Figure 15 – Input samples (yellow squares) required for generating the fractional samples (green and red squares)

location of eight quarter blocks to be evaluated, which were generated around this best block. This algorithm was represented on Figure 16, where the eight half samples evaluated firstly was the gray squared, and the eight quarter samples evaluated are the yellow squared, which assumes that the best half block is the block 40.

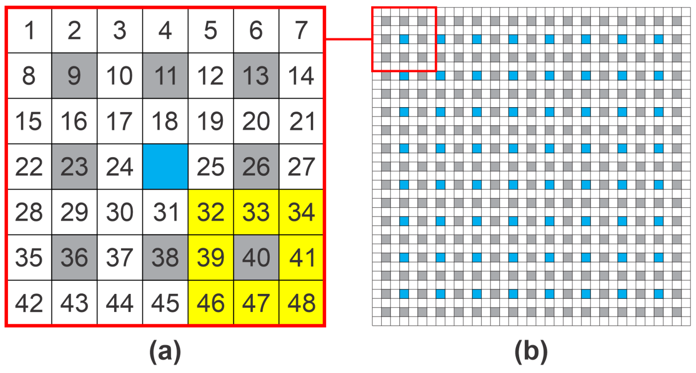

Figure 16 – FME algorithm assuming that the best half block is the block 40
Source: Adapted from AFONSO et al. (2016a).

These fractional blocks are very similar to each other. Therefore, to perform a more precise comparison between these fractional blocks, the 3D-HTM (JCT-3V, 2020) adopts the SATD similarity criterion (See section 2.3) on FME processing.

2.4 Complexity Analysis

The several encoding tools available on 3D-HEVC can efficiently encode 3D contents that adopt the MVD format. These encoding tools are highly complex to be computed, mainly due to the total number of block sizes supported by 3D-HEVC. Although, the evaluation of the best prediction mode to encode each block is mandatory to achieve the best trade-off between compression efficiency and image distortion.

To analyze the cost of the 3D-HEVC encoding tools, AFONSO (2019) has evaluated the time consumed by different encoding tools of this standard. Its evaluation was performed considering the Random-Access (RA) configuration (MÜLLER; VETRO, 2014), considering two video sequences (Balloons and Undo_Dancer) and two Quantization Parameters (QP) values (30 and 39) as specified in Common Test Conditions (CTC) document (MÜLLER; VETRO, 2014).

Figure 17 presents the encoding time of the prediction tools of 3D-HEVC, considering both texture and depth map channels. As can be seen, the encoding time required by the texture channel (51.1%) is similar to the encoding time required by the depth map channel (48.9%). Although, the amount of time required by the prediction tools has several differences through these two channels. To encode the texture channel, the inter-frames prediction tools require a half of the encoding time, being the ME the most costly inter-frames prediction tool by requiring 40.95% of the time, while the DE requires 10.58%, and other encoding tools (intra-frame prediction, residual coding, and entropy coding) uses the rest of the encoding time. To encode the depth map channel, the inter-frames prediction tools require only a small part of the encoding time, where the ME requires 6.33% of the encoding time, while the DE requires only 2.29%. The other encoding tools use the rest of the encoding time.

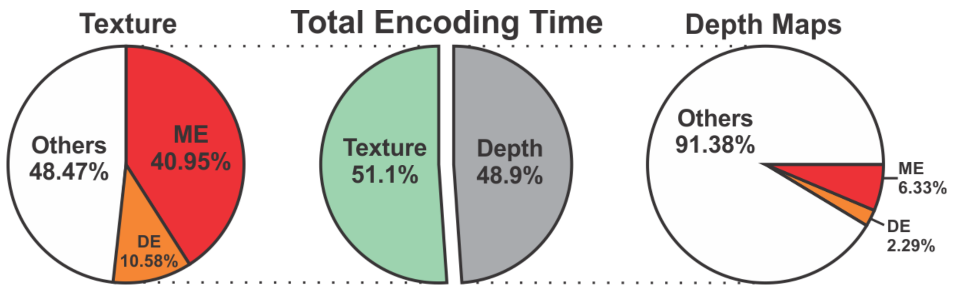

Figure 17 – Encoding time portions of prediction steps
Source: Adapted from AFONSO (2019).

Similar results can be obtained considering both channels, texture, and depth maps, where the ME tool requires a high portion of the total encoding time. This can be seen in Figure 18, which presents the encoding time of prediction steps considering both channels. As can be seen in Figure 18, while the ME requires 23.45% of the encoding time, the Intra-prediction tools require 12.52%, about half of the ME time. Finally, the DE tool requires only 6.39%, and the other encoding tools (residual and entropy coding) require 57.64% of the total encoding time.

While the ME is the prediction tool that requires a high computational time, it has also huge importance in video encoding. According to (AFONSO, 2019), when considering the NCO, the ME was the encoding tool used to represent more than 80% of the Inter-frames predicted samples. When considering the FCO, the ME was used to represent more than 90% of the Inter-frames predicted samples.

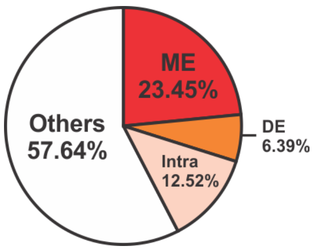

Figure 18 – Accumulated encoding time portions of prediction steps
Source: Adapted from AFONSO (2019).

Therefore, as presented, there are a huge number of operations performed by the ME tool to find the most similar block in a reference frame. These several operations are responsible for the high time demanding of video encoding. Thus, the applications which require the encoding of 3D-videos in real-time have to adopt dedicated hardware devices to efficiently perform these operations of the ME encoding tool with a desirable throughput.

2.5 Related Works

There are several works on literature proposing hardware architectures for different tools of different video encoding standards. There are a few works proposing hardware architecture for the 3D-HEVC. The works ÜCKER et al. (2018), SANCHEZ et al. (2018), SANCHEZ; MARCON; AGOSTINI (2017), AMISH; BOURENNANE (2019) proposes hardware architectures for the Intra-Prediction tools of 3D-HEVC. There also are some works on literature proposing hardware solutions for the DE tool, as AFONSO et al. (2018), PERLEBERG et al. (2018) and PERLEBERG et al. (2019). The DE shares several similarities with the ME tool. Therefore, any solution for the DE tool can also be used to perform the ME context. For that processing, it must only be selected as reference the frames from the same view of the frame being encoded. However, the algorithm used on these works was specifically developed for the DE tool, thus they were not effective in the ME context, which can result in a high impact in encoding efficiency.

However, only AFONSO et al. (2019) presents a hardware architecture for the ME tool by proposing a complete solution for both ME and DE tools. It has different modules for each encoding tool, being that each module specifically performs the BMA algorithm. Moreover, AFONSO et al. (2019) can only deal with two PU sizes. The results of its proposed algorithm show an average impact of 23.22% in the encoding efficiency considering all evaluated sequences. Its synthesis results show that the developed hardware requires 40,888 K NAND-2 Gates, with a power dissipation of 6.447 W when processing three views of Full High-Definition (FHD) $1920x1080p$ video at 30 fps.

In order to achieve a more complete comparison in the results of the developed hardware, hardware architectures targeting the ME of the HEVC was also selected for comparison. Although, it should be noted that only values related to the hardware results (as the power dissipation and resources required) can be compared since the ME of both HEVC and 3D-HEVC was pretty similar. Other results, as the impact in compression efficiency, cannot be compared.

The works MEDHAT; SHALABY; SAYED (2015), FAN et al. (2018), LIAO; SHEN; TSENG (2019) and GU et al. (2019) presents Hardware Architectures for the IME tool of HEVC. The work of MEDHAT; SHALABY; SAYED (2015). The work MEDHAT; SHALABY; SAYED (2015) presents a hardware design for the IME focusing on an ASIC design, which implements the FS algorithm. It can process all PU sizes supported by the HEVC and its architecture is able to process Ultra High-Definition (UHD) $3840x2160p$ videos at 30 fps. Although, the work MEDHAT; SHALABY; SAYED (2015) does not present power analysis or the impact of its strategy regarding the compression efficiency. The work FAN et al. (2018) 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 videos at 30 fps. Whereas the work FAN et al. (2018) presents power results and partially measures the impact of its strategy in terms of compression. Also, FAN et al. (2018) does not report the impact of not using bidirectional prediction in its evaluations.

The work LIAO; SHEN; TSENG (2019) proposes an IME algorithm that adapts the SA of the FS algorithm, thus evaluating 96% fewer candidate blocks than using the original SA. Its developed hardware is able to process UHD 2160p videos at 60 fps. The work GU et al. (2019) also proposes an IME algorithm that can use different patterns of search with a reduced SA. The developed hardware reaches the best throughput when using the diamond pattern, being able to process at UHD 2160p videos at 139 fps for the square-shaped PU sizes. However, the power results of the hardware proposed by both LIAO; SHEN; TSENG (2019) and GU et al. (2019) were not provided. Moreover, both these works consider processing only one reference frame, thus it cannot use bidirectional prediction, and the impact of it was not measured on its evaluations.

The work AFONSO et al. (2016a) presents a Hardware Architecture for the FME of HEVC. Its architecture performs the interpolation and comparison of all fractional blocks around an $8x8$ IME result. It can only perform the processing of the four square-shaped PU sizes of HEVC, by inheriting the results of $8x8$ PU size. Its synthesis results show that the proposed hardware can reach the throughput needed to process UHD 2160p videos at 60 fps.

The work PERLEBERG et al. (2018), XU et al. (2018), PASTUSZAK; TROCHIMIUK (2016) presents Hardware Architecture for the complete ME tool of HEVC. On PER-

LEBERG et al. (2018) a hardware architecture for the complete ME was developed, able to process the four square-shaped PU sizes supported by HEVC. It adopts several independent modules, being that each of those modules was specifically developed to performs the IME according to a modified TZS algorithm and evaluate all fractional blocks to only one PU size. The proposed architecture is able to process $7680x4320p$ videos at 53 fps. The work XU et al. (2018) proposes an IME algorithm that uses three stages of a sub-sampled FS, being that in each stage the SA was reduced and also the sub-sampling. Its developed hardware sequentially performs both IME and FME. The IME was performed to all PU sizes supported by HEVC. However, it does not mention which IME result will be processed by the FME unit, and how many PU sizes were supported by the FME unit. Its synthesis results show that the proposed architecture is able to process UHD 2160p videos at 30 fps while supporting bi-prediction.

2.6 Main Motivation

As previously mentioned, the ME is the encoding tool that requires most of the encoding time and was responsible for encoding a significant part of inter-predicted samples. The critical part of the ME is on the IME tool, due to the high dependency of the decisions of TZS steps among the several candidates evaluated. To be applied, the IME demands a huge computational effort considering the number of operations to be performed and also a significant memory communication to obtain the sample information, as follows.

As previously mentioned, processing the TZS for a PU can require the evaluation of up to 1011 candidates, and this processing requires computing the similarity criterion value for each candidate. As can be seen on Equation (1) (on Section 2.3), the complexity of one SAD operation are defined according to the PU size. Therefore, in this work the complexity to compute one SAD value will be represented in terms of the number of SAD operations for $4x4$ sub-blocks. As an example, the cost for computing the SAD of a $8x4$ PU is approximated the same as the computing 2 SAD values of two neighboring $4x4$ sub-blocks. Then, considering the PU Height and Width, the Equation (5) results in the number of SAD operations for $4x4$ sub-blocks to compute its SAD value.

$$c(PU_{height}, PU_{width}) = (PU_{height}/4) * (PU_{width}/4) \tag{5}$$

Considering the encoding of a CTU through the IME encoding tool, the 3D-HEVC has the support of up to 24 different PU sizes to be evaluated. Moreover, should be noted that a CTU can be divided into N PUs with the same size, being N given by Equation (6). Although, it can be obtained by Equation (7) that the cost for processing all PUs with the same size is the same as computing 256 SAD operations for $4x4$

sub-blocks, considering any of the 24 PU sizes.

$$N(PU_{height}, PU_{width}) = (64/PU_{height}) * (64/PU_{width}) \quad (6)$$

$$j(PU_{height}, PU_{width}) = N(PU_{height}, PU_{width}) * c(PU_{height}, PU_{width}) \quad (7)$$

Therefore, considering that in the 3D-HTM all 24 PU sizes are evaluated sequentially and independently (without operations reuse techniques) and that the TZS can evaluate up to 1011 candidates, the processing of the IME tool to the current CTU requires the processing of up to 6,211,584 SAD values of $4x4$ sub-blocks. This value is given by K value on Equation (8), which considers the 24 PU sizes to be evaluated, the 1011 candidates to be evaluated by the TZS algorithm, and the 256 SAD operations to evaluate all PUs with the same size.

$$K = 24 * 1011 * 256 \quad (8)$$

Similarly than the number of SAD operations, it could be observed by Equation (1) (on Section 2.3) that the SAD operation requires the samples from the current block and from the candidate block. Thus, the number of samples requested from the memory to compute the SAD operation also depends on the PU size, as given by V on Equation (9).

$$V(PU_{height}, PU_{width}) = 2 * PU_{height} * PU_{width} \quad (9)$$

Moreover, similarly than the number of SAD operations, since a CTU can be divided in several PUs with the same size, the number of samples requested from the memory to process one candidate through all PUs with the same size is given by G on Equation (10), considering any PU size supported on IME of 3D-HEVC.

$$G(PU_{height}, PU_{width}) = N(PU_{height}, PU_{width}) * V(PU_{height}, PU_{width}) \quad (10)$$

Therefore, considering that the TZS can evaluate up to 1011 candidates and that the evaluation of 24 PU sizes will be performed independently, the processing of the current CTU through the IME tool can request up to 198,770,688 samples from the memory. This value is given by H value on Equation (11), which considers the 24 different PU sizes supported on IME, the 1011 candidates to be evaluated by the TZS algorithm, and the number of samples requested from the memory to process one candidate through all PUs with the same size.

$$H = 24 * 1011 * G(PU_{height}, PU_{width}) \quad (11)$$

Finally, to encode 3 views of an FHD 1080p resolution video with 30 fps in real-time can request an unfeasible memory communication of 18.25TB/s as given by P on Equation (12), which considers the memory communication to process each CTU, that each FHD 1080p frame has 510 CTUs, the two channels to be encoded (Texture and Depth Map), the 3 views from different cameras, the temporal resolution, and that each sample has 8 bits (MÜLLER; VETRO, 2014).

$$P = H * 510 * 2 * 3 * 30 * 8 \tag{12}$$

Thus, can be seen that computing the IME tool to one CTU requires the processing of up to 6,211,584 SAD values of $4x4$ sub-blocks, while it can request up to 198,770,688 samples from the memory. Aiming the development of hardware architectures for the IME, the computational effort and memory communication should be as lower as possible.

As can be seen on Equation (8) and (11), one strategy to reduce the amount of SAD operations and the memory communication to process a CTU is by reducing the number PU sizes supported on IME, as performed by several works on literature (LIAO; SHEN; TSENG, 2019) (GU et al., 2019) (AFONSO et al., 2016a) (PERLEBERG et al., 2018) (XU et al., 2018) (AFONSO et al., 2019). Other strategies to reduce ME complexity is by perform several simplifications in the IME algorithm, by applying several modifications in the TZS algorithm to reduce the number of candidates evaluated (GU et al., 2019) (FAN et al., 2018) (PERLEBERG et al., 2018) (PASTUSZAK; TROCHIMIUK, 2016) (AFONSO et al., 2019), or even by reducing the size of the SA (MEDHAT; SHALABY; SAYED, 2015) (LIAO; SHEN; TSENG, 2017) (GU et al., 2019) (XU et al., 2018). However, these strategies can result in a high decrease in the compression efficiency of the 3D-HEVC standard.

Another strategy is to adopt reuse strategies, which were adopted in this paper. Considering the different PU sizes supported by the 3D-HEVC standard, there are several redundancies in the SAD operations performed by the TZS algorithm. As an example, considering the same candidate, the SAD of an $8x8$ PU is computed similarly than the SAD of the four $4x4$ PUs, but accumulating all four SADs to compose the SAD of the $8x8$ PU. Considering the TZS algorithm, several candidates can be evaluated similarly several times when processing different PU sizes, mainly the candidates from Prediction Step, from First Search and Raster Step. Thus, by knowing in advance which candidates will be evaluated by different PU sizes, operations reuse strategies could be adopted to compute firstly the SAD of smaller PUs, and then just join the SAD of two small neighboring PUs to compose the SAD of higher PUs. These strategies also reduce memory communication, since any data will be requested from the memory for computing the higher PU sizes.

If all candidates are shared between the 24 PU sizes, the reuse strategies allow that only the smallest PU size needs to be completely processed, and its results can be used to compose the results of higher PUs. So, the complete processing of only one PU size can represent a reduction in both the number of SAD operations and the number of memory communication of up to 95.83%.

Therefore, the hardware architecture of this work explores strategies of reusing SAD operations by using the SAD of smaller PUs to compute the SAD of higher PUs. To know which candidates of the TZS will be repeatedly evaluated by different PU sizes, the TZS of all PU sizes was synchronized, thus all PU sizes evaluate the same candidates. In the sequence, the next chapter presents the hardware-oriented constraints applied in the IME to allow the development of the IME architecture with those reuse strategies and the impact of those constraints in the encoding efficiency.

3 SOFTWARE EVALUATIONS

As previously mentioned, the ME is the encoding tool that requires most of encoder complexity. To obtain a high-throughput hardware design capable to deal with three views of a FHD 1080p video at 30 fps, this tool must be simplified by applying hardware-oriented constraints or even modifying its algorithm to allow efficient resources usage. Therefore, this chapter presents the constraints and their impact in the encoding efficiency considering the use of the 3D-HTM, the reference software of the 3D-HEVC (JCT-3V, 2020). The first section presents the setup and conditions adopted to perform the experiments, while the second section presents each experiment performed and its respective results.

3.1 Experimental Setup

To evaluate the impact of the modifications in the encoding efficiency, the experiments were performed using the 3D-HTM software in version 16.2 (JCT-3V, 2020). These experiments consider the 3D-HEVC Common Test Conditions (CTC) (MÜLLER; VETRO, 2014), which was recommended by the JCT-3V.

The CTC document defines eight 3D videos sequences to be evaluated, and specific details of each sequence can be seen in Table 1. The CTC defines three sequences with $1024x768$ resolution (Balloons, Kendo, Newspaper), and five sequences with $1920x1088$ resolution (GT_Fly, Poznan_Hall2, Poznan_Street, Undo_Dancer, Sha rk). It should be noted that each sequence has a different frame rate, varying from 25 to 30 fps. Moreover, between 200 and 300 AUs were encoded from each video sequence, as recommended by the CTC document (MÜLLER; VETRO, 2014). Each AU contains all texture pictures and their respective depth maps of the same time instant.

Table 1 also presents the index of the views to be encoded, being that each view is composed by the texture channel plus its respective depth-map channel.

The CTC document also defines the QP values to be used in the experiments, where different QPs are used for the texture and the depth maps. For the texture view, it defines the use of 25, 30, 35 and 40 values. The QPs for the depth maps is pre-

Table 1 – Reccomended video sequences according to Common Test Conditions document

Resolution	Sequence	Frame Rate (fps)	Access Units	Views
1024x768	Balloons	30	300	3-1-5
	Kendo	30	300	3-1-5
	Newspaper	30	300	4-2-6
1920x1088	GT_Fly	25	250	5-9-1
	Poznan_Hall2	25	200	6-7-5
	Poznan_Street	25	250	4-5-3
	Undo_Dancer	25	250	5-1-9
	Shark	30	300	5-1-9

defined according to the QP of the texture view. Therefore, the four texture QPs result in the QP values 34, 39, 42 and 45 for the depth, respectively.

The results of the experiments performed in this chapter considers the metric Bjontegaard Difference bit rate (BD-rate) metric (BJONTEGAARD, 2008). This BD-rate metric processes the results of all four QP sets, resulting in the percentage variation in the bit rate to obtain the same objective image quality when compared with a baseline experiment - in this work, compared with the original algorithm of 3D-HTM software in version 16.2. Therefore, higher values of BD-rate indicate a decrease in the compression efficiency, since it represents a higher bit rate, while smaller or negative values of BD-rate indicate a increase in compression efficiency, since it represents a smaller bit rate.

Two evaluations were performed to each experiment proposed on the next section, one of them considering the RA temporal configuration with the NCO configuration, and the other one running on the RA configuration with the FCO configuration.

3.2 Hardware-Oriented Constraints Evaluation

The hardware constraints experiments were divided into five incremental levels of experiments. The first four levels are composed of hardware-oriented constraints in the ME algorithm, while the last one adopts a different approach for the FME. Each of those five levels is described in the next subsections.

3.2.1 ME Hardware Constraints (1L)

The first level (1L) of the experiments was conducted to set some hardware constraints in the ME algorithm of the 3D-HTM, aiming to obtain a hardware-friendly implementation of the algorithm. This first experiment includes four modifications: 1) As mentioned before, the 3D-HTM supports bi-directional prediction with several reference frames for each direction, which increases the ME computational effort and memory requirements. In this first experiment, the bi-directional prediction was disabled, the search was limited to only one reference frame. 2) Also, since the 3D-HTM adopts the SATD similarity criterion over the FME encoding tool, which also demands a high computational effort, the SAD was adopted instead of the SATD for the FME. 3) Moreover, as previously explained, the AMVP algorithm is adopted by the 3D-HTM to predict the position of the IME Search Area, which could result in arbitrary memory access. Thus, in this first experiment, the AMVP was disabled. 4) In addition, there are different predictors of the TZS algorithm that can be used. In order to break the dependency of the neighboring block previously evaluated, only the Zero predictor was used in the TZS. In other words, the search was always performed around the collocated block, resulting in regular memory access.

Table 2 presents the impact in the encoding efficiency in terms of the BD-rate value of the 1L considering both NCO and FCO. In general, the accumulated results show an average BD-rate increase of 6.385% in the NCO and an average BD-rate increase of 6.313% in the FCO. Also, it is possible to observe in both encoding orders that these constraints show a higher impact in higher resolutions, probably due to the modifications in the SA and TZS predictors.

3.2.2 TZS Hardware Constraints (2L)

The second level (2L) of the experiments is incremental to the 1L experiment while it was focused on modifying the TZS steps in order to reduce its high complexity and aiming to turn them hardware-friendly. Although the TZS has a better performance when compared with the Full Search algorithm, it still can evaluate up to 1011 candidate blocks considering the case presented in Section 2.3.1.4. Moreover, the TZS algorithm has a high dependency due to the number of decisions related to it.

Therefore, there are three modifications performed in the TZS algorithm: 1) The Raster is the TZS step that requires the higher number of candidates to be processed. In order to reduce the Raster complexity, the Search Range was reduced to half of its original size, so the Raster is applied in only 25% of its normal size, which is sufficient since this reduced SA contemplates 62.3% of the best candidates found by the Raster using the original SA considering the HEVC standard (Goncalves et al., 2018). Thus, when the Raster were applied, only 169 candidates are evaluated. 2) The First Search has some dependencies on its processing due to its stop conditions, which difficult its

Table 2 – BD-Rate Increase according to the video sequence from adopting hardware constraints in 1L experiment

Experiment	Resolution	Sequence	NCO	FCO
1L (Hardware Constraints)	1024x768	Balloons	4.535%	4.501%
		Kendo	5.283%	4.961%
		Newspaper	3.187%	3.160%
	1920x1088	GT_Fly	14.332%	14.202%
		Poznan_Hall2	4.523%	4.486%
		Poznan_Street	3.728%	3.516%
		Undo_Dancer	5.916%	5.918%
		Shark	9.572%	9.758%
	Average 1024x768		4.335%	4.207%
	Average 1920x1088		7.614%	7.576%
	Average		6.385%	6.313%

parallelism. Therefore, the stop conditions of the First Search (presented in Section 2.3.1.2) were removed, so the search always expands to the limit of the SA, evaluating all 76 candidates of the expansive diamond search. This change was also performed to supply the impact of the Raster constraint, thus allowing a small increase in the encoding efficiency. 3) The Refinement step from the TZS has a problem to be implemented in hardware since the refinement could be performed in several consecutive iterations while a better candidate is found. Moreover, all those iterations cannot be performed in parallel due to the dependency between iterations. Since any algorithm to be hardware-implemented must comply with time constraints, the 2L experiment also considers limiting the refinement step to the maximum of seven consecutive refinements.

These modifications ensure that the TZS algorithm will be restricted to the evaluation of less than 778 candidate blocks (worst case). Table 3 presents the impact in the encoding efficiency in terms of the BD-rate value of the 2L experiment. As can be seen, these modifications presented an increase of 6.401% and 6.266% on average considering the NCO and FCO schemes, respectively. Compared with the average of 1L experiments, the 2L experiment was reached an average increase of only 0.016% and 0.047%, respectively.

Moreover, it should be noted that in the $1024x768$ videos, the 2L modifications present a small improvement in encoding efficiency than 1L experiments, on both encoding orders. This probably occurs because in the $1920x1088$ videos the movement

Table 3 – BD-Rate Increase according to the video sequence of the TZS modifications in 2L experiment

Experiment	Resolution	Sequence	NCO	FCO
2L (TZS Constraints) + 1L	1024x768	Balloons	4.420%	4.334%
		Kendo	5.070%	4.771%
		Newspaper	3.085%	3.047%
	1920x1088	GT_Fly	15.004%	14.628%
		Poznan_Hall2	4.254%	4.184%
		Poznan_Street	4.206%	3.986%
		Undo_Dancer	5.530%	5.547%
		Shark	9.636%	9.628%
	Average 1024x768		4.192%	4.051%
	Average 1920x1088		7.726%	7.595%
	Average		6.401%	6.266%
1L (Previous Experiment)	Average 1024x768		4.335%	4.207%
	Average 1920x1088		7.614%	7.576%
	Average		6.385%	6.313%

is higher and thus it suffers a greater impact due to the limitations in the Raster step, while in $1024x768$ videos the movement is smoother, thus the limitations in the Raster step does not affect its performance, while the modifications in the First Search helps in increasing the encoding efficiency.

3.2.3 TZS Master-Slave (3L)

The third level (3L) of the experiments increments the 2L experiment by attaching the TZS decisions of a specific PU size for all other evaluated PU sizes. By default, the TZS decisions are applied according to the SAD value of the evaluated candidates, which depends on the processed PU size. Thus, different decisions could be performed to different PU sizes. As an example, the $64x64$ PU can decide to perform the Raster step, while all smaller PUs from the same CTU goes straight from the First Search to refining a different candidate. Therefore, it was impossible to reuse SAD processing between different PU sizes.

The 3L solves this problem by applying the same TZS decisions to all PU sizes by using a Master-Slave TZS algorithm. The TZS Master is the one applied to the $64x64$ PU, which stores all decisions of which candidates will be evaluated. The Slave was the TZS applied to all smaller PUs of the same CTU, which applies the same decisions of the Master TZS.

An example of a TZS processing for a $64x64$ PU was represented in Figure 19. In Figure, the green squares represent the first sample of each candidate evaluated in the First Search, while the yellow squares represent the candidates evaluated in a Refinement step centered in the position [4,4], the best result of the First Search. Finally, the red square represents the location of the best candidate evaluated for the $64x64$ PU.

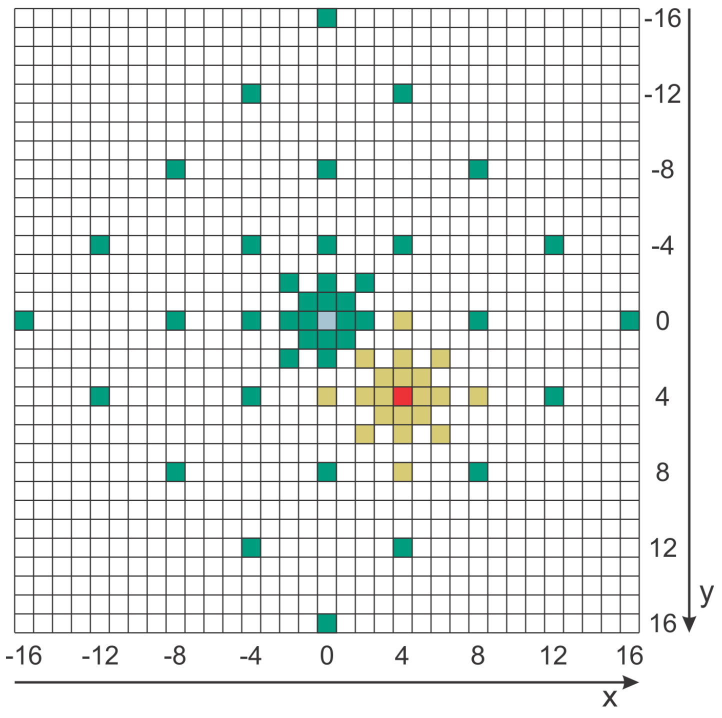

Figure 19 – TZS processing: Candidates from the First Search and from the first Refinement

Figure 19 represents the execution of the TZS for the $64x64$ PUs. Although, in the TZS Master-Slave algorithm all smaller PU sizes will evaluate the same candidates represented in Figure 19. This means that the smaller PU sizes will lose their independence. By applying the First Search from Figure 19 to $8x8$ PUs, even if all $8x8$ PUs result in Refining a candidate different than the [4,4], the Refinement to be applied must be in the same location of the refinement applied for the $64x64$ PU.

Although, by applying the Master-Slave TZS, all PU sizes will evaluate the same candidates, ensuring the possibility to apply techniques for data reuse of different PU sizes. It should be noted that although all PU sizes will evaluate the same candidates, it does not restrict all PU sizes resulting in the same candidate. So, neighboring PUs from a CTU can result in motion vectors that are related to different candidates. As an example, the $64x64$ PU result in the vector [4,4], while some $8x8$ PUs result in the vector [-2,0] evaluated in the First Search, and some $8x8$ PUs result in vector [6,6] evaluated in the Refinement.

Table 4 present the results of the proposed Master-Slave TZS. It is possible to observe that the 3L results in an average increase of 9.628% and 9.502% in the BD-rate value considering both NCO and FCO schemes, respectively. Moreover, comparing with the results of the previous experiment, an increase of 3.227% and 3.236% was obtained in NCO and FCO schemes, respectively. This increase is mainly due to removing the independence of TZS in smaller PU sizes, which has for evaluation only the same candidates evaluated by the TZS in $64x64$ PU size.

Table 4 – BD-Rate Increase according to the video sequence from the TZS Master-Slave algorithm in 3L experiment

Experiment	Resolution	Sequence	NCO	FCO
3L (Master-Slave TZS) + 2L	1024x768	Balloons	6.608%	6.461%
		Kendo	7.501%	7.113%
		Newspaper	3.982%	4.025%
	1920x1088	GT_Fly	23.047%	22.873%
		Poznan_Hall2	6.074%	6.103%
		Poznan_Street	6.034%	5.848%
		Undo_Dancer	7.511%	7.419%
		Shark	16.269%	16.174%
	Average 1024x768		6.030%	5.866%
	Average 1920x1088		11.787%	11.683%
	Average		9.628%	9.502%
2L (Previous Experiment)	Average		6.401%	6.266%

3.2.4 Full Splitting (4L)

The fourth level (4L) of the experiments was focused on enabling a feature of the hardware implementation while maintaining the modifications of previous experiments. By adopting the Master-Slave TZS from the 3L experiment, the TZS from all PU sizes will evaluate the same candidates. In the hardware design, the processing of each candidate can be implemented with data reuse techniques to process all PU sizes in parallel at a low cost, by computing the similarity value of the smallest PU sizes firstly and, after, accumulating these similarity values to compose the result of higher PU sizes.

However, by default, the 3D-HTM software adopts the RD cost to define the prediction tool and the partition scheme to used on each CTU. This scheme applies the TZS algorithm on the higher PU sizes firstly, while skipping the evaluation of smaller PU sizes when some conditions are reached to reduce run-time.

Therefore, in order to evaluate the gains on the encoding efficiency of evaluating all PU sizes supported by the 3D-HEVC (a feature that will be supported by the developed hardware due to the adoption of data reuse techniques), the 4L experiment performs the evaluation of all PU sizes to each TZS candidate.

The obtained results of grouping the techniques proposed in the four levels of experiments are presented in Table 5. In Table 5 can be observed an average impact of 9.061% and 8.823% in terms of BD-rate value considering both NCO and FCO, respectively. Compared with the results of the 3L experiment, the evaluation of all PU sizes to each candidate resulted in a small reduction of 0.612% in the NCO scheme, and of 0.679% in the FCO scheme, which represents a small gain in terms of compression efficiency, since by evaluating all PU sizes, the encoder has more possibilities to represent the movement of smaller objects present inside a CTU, while the original partitioning algorithm can interrupt the evaluation of smaller PUs according to the RD cost.

Table 5 – BD-Rate Increase according to the video sequence from adopting a Full Splitting mode in 4L experiment

Experiment	Resolution	Sequence	NCO	FCO
4L (Full Splitting) + 3L	1024x768	Balloons	5.980%	5.814%
		Kendo	6.988%	6.670%
		Newspaper	3.943%	3.690%
	1920x1088	GT_Fly	21.761%	21.491%
		Poznan_Hall2	5.184%	5.115%
		Poznan_Street	5.722%	5.511%
		Undo_Dancer	7.083%	6.960%
		Shark	15.469%	15.335%
	Average 1024x768		5.637%	5.392%
	Average 1920x1088		11.044%	10.883%
	Average		9.016%	8.823%
3L (Previous Experiment)	Average		9.628%	9.502%

3.2.5 Modified Approach for the FME (5L)

This work was focused on a hardware solution for the IME tool, being all experiments around it. However, the result of the IME tool is the input of the FME tool, since traditionally each PU is processed by either the IME and after by the FME to conclude the motion estimation process. Therefore, an IME implementation must consider how the FME will be performed. So, the fifth level (5L) of experiments was necessary to complement the result of the 4L experiment by evaluating the encoding efficiency of two opposite approaches of the FME encoding tool.

The first approach is to evaluate the impact in the encoding efficiency of disabling the FME tool, the worst case among the encoding efficiency. It assumes that no hardware for the FME will be implemented, so the vector result of the IME will be used as the final result for the ME. The impact in the encoding efficiency of disabling the FME is presented in Table 6. By disabling the FME resulted in a BD-rate increase of 17.93% in the NCO, and a BD-rate increase of 17.712% in the FCO scheme, on average. When compared with the 4L experiment, disabling the FME resulted in a degradation of 8.914% and 8.889% in the encoding efficiency (on average) considering both NCO and FCO, respectively.

Table 6 – BD-Rate Increase from disabling the FME according to the video sequence

Experiment	Resolution	Sequence	NCO	FCO
5L (Disabling FME) + 4L	1024x768	Balloons	14.879%	14.855%
		Kendo	15.244%	14.749%
		Newspaper	7.417%	7.129%
	1920x1088	GT_Fly	38.702%	38.436%
		Poznan_Hall2	12.177%	12.189%
		Poznan_Street	8.632%	8.360%
		Undo_Dancer	19.634%	19.455%
		Shark	26.757%	26.525%
	Average 1024x768		12.513%	12.244%
	Average 1920x1088		21.180%	20.993%
	Average		17.930%	17.712%
4L (Previous Experiment)	Average		9.016%	8.823%

By default, the 3D-HTM software evaluates only a set of 16 fractional blocks according to its similarity result. However, the fractional blocks share several fractional samples. So, after interpolating these new samples, it is necessary to just group these samples to compose each fractional block to be evaluated, which can be easily performed in an FME implementation at a low cost (assuming that the higher cost is related to the interpolation filters).

Therefore, the second approach assumes that all fractional samples will be generated and used to compose all possible fractional blocks to all PU sizes supported, the best case for the FME since this can result in a small improvement in the encoding efficiency. The obtained results of evaluating all possible fractional blocks to all PU sizes are presented in Table 7. Evaluating all 48 fractional blocks resulted in an average of 8.945% and 8.766% in the BD-rate metric in the NCO and FCO schemes, respectively. Also, compared with the 4L experiment that adopts the original FME algorithm, this indicates a small gain of 0.071% and 0.057% in both NCO and FCO, respectively.

Table 7 – BD-Rate Increase from evaluating all fractional blocks according to the video sequence

Experiment	Resolution	Sequence	NCO	FCO
5L (All fractional blocks) + 4L	1024x768	Balloons	5.961%	5.862%
		Kendo	6.890%	6.568%
		Newspaper	3.847%	3.708%
	1920x1088	GT_Fly	21.590%	21.344%
		Poznan_Hall2	5.166%	5.082%
		Poznan_Street	5.726%	5.407%
		Undo_Dancer	6.964%	6.873%
		Shark	15.414%	15.288%
	Average 1024x768		5.566%	5.379%
	Average 1920x1088		10.972%	10.799%
	Average		8.945%	8.766%
4L (Previous Experiment)	Average		9.016%	8.823%

3.3 Experiments Conclusions

The evaluation results allow the development of efficient hardware design for the IME tool. The TZS dependency was reduced by limiting its steps, and also regular memory access was obtained by disabling the algorithms that could move the SA away.

Moreover, since the TZS from all PU sizes will evaluate the same candidates, it is possible to explore operations reuse techniques to process higher PU sizes based on the result of lower PU sizes, which also reduces in up to 95.83% the number of SAD operations and the memory communication needed to request the samples of larger PU sizes.

According to the evaluation results, if only the IME was developed, disregarding the FME, it is expected degradation of 17.93% in the encoding efficiency in terms of BD-rate metric if using the NCO scheme, and degradation of 17.712% in the encoding efficiency if using the FCO scheme. Meantime, if an FME that processes all fractional blocks were integrated with the developed IME, a degradation of 8.945% in the encoding efficiency is expected if considering the NCO scheme. In the FCO scheme, an FME that processes all fractional blocks being integrated with the developed IME results in a degradation of 8.766% in the encoding efficiency.

Although these results seem to represent a high impact in the encoding efficiency, they are acceptable when considering all hardware-oriented constraints adopted that allows the development of hardware with reduced complexity and regular memory access. Moreover, the main related work presenting a hardware architecture for the ME tool of the 3D-HEVC standard (AFONSO et al., 2019) reaches a BD-rate impact of up to 23.2%.

Thus, based on these results, the next chapter presents the development of the proposed architecture with the hardware-oriented constraints presented in this chapter, and also adopting data and operations reuse strategies provided by the TZS Master-Slave and from evaluating all partitioning to reduce the memory communication and the number of operations performed.

4 PROPOSED MOTION ESTIMATION HARDWARE ARCHITECTURE

This chapter presents in detail the developed architecture, which was divided into four sections. The first section presents in detail the developed IME architecture, while the second section presents a solution for an FME architecture. After, the third section explains the synchronism between the IME and the FME Units, while the last one discusses memory management.

The developed architecture presents a solution to perform in parallel the whole Motion Estimation algorithm. The IME Unit was developed with the constrains previously defined in order to adopt data reuse strategies to compute the SAD value from all PU sizes efficiently, by composing the SAD from all PU sizes based on the SAD value from smaller PU sizes. Also, the FME solution integrated into this ME hardware was designed to avoid redundancy in the operations when generating the fractional candidates, that can occur if two or more IME motion vectors are similar. In this FME the SA was interpolated once, and all fractional blocks that depend on the interpolated samples were computed in parallel.

Figure 20 presents the complete architecture. It was divided into three main modules, being two modules for the IME, and one module for the FME. The first module applies the Test Zone Search algorithm by selecting the candidates from the SA to be evaluated. It is also responsible for computing the SAD value from sub-blocks of the candidate being evaluated.

The second one is a SAD Table module, which is responsible for accumulating the SAD of the sub-blocks to obtain the SAD value of higher PUs, so it computes the SAD from all PUs sizes supported by the 3D-HEVC. This module also stores the SAD value and the respective motion vector from the candidates with the smallest SAD value of each partition scheme. The SAD Table feeds the IME Unit with the SAD value from the $64x64$ PU, so the TZS control can decide the next candidate to be evaluated.

The last module is the FME module, which implements the interpolation filters defined by the 3D-HEVC to generate the Fractional blocks. It generates and evaluates the fractional blocks indicated by the motion vectors from the best partition scheme.

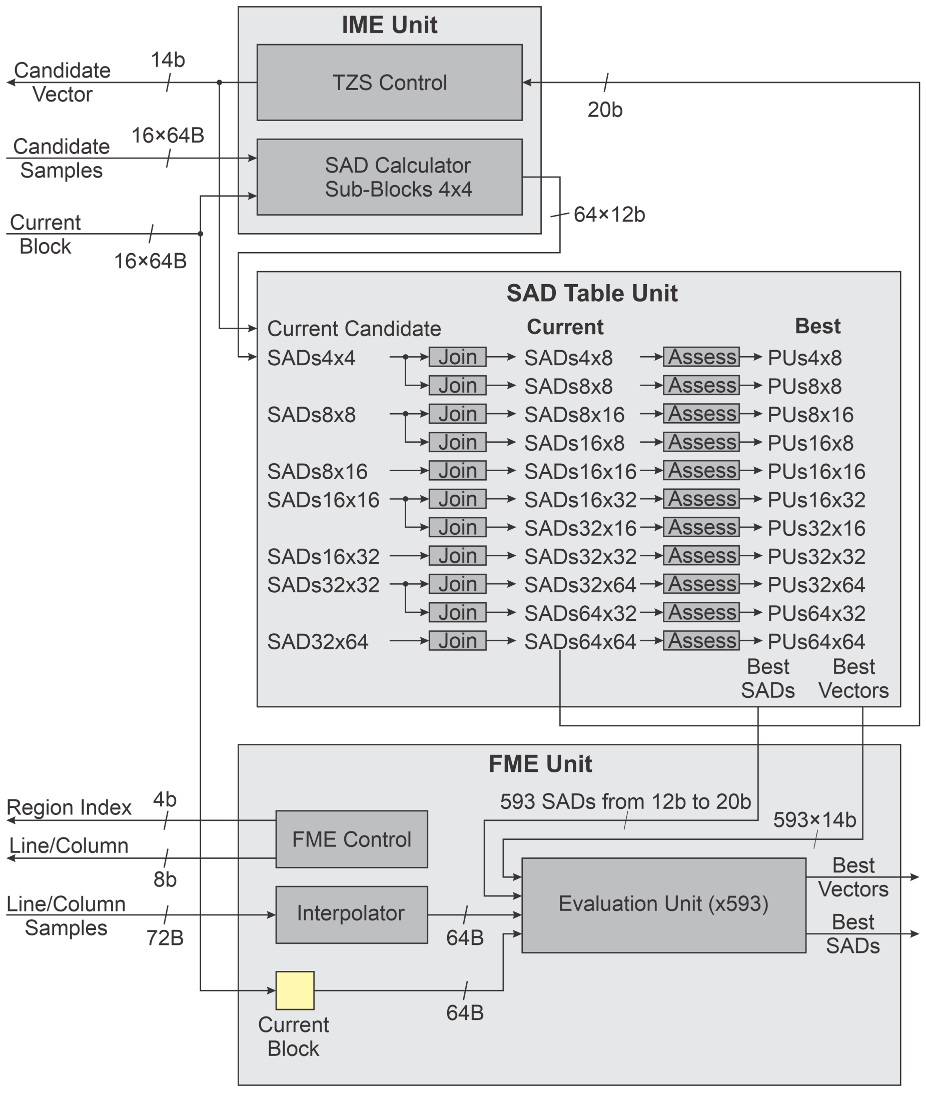

Figure 20 – The complete ME architecture

The next two sections explain in detail these modules, while the third section presents the timing analysis and operation considering all these three modules. Finally, the fourth section presents the memory requirements of the designed hardware.

4.1 Integer Motion Estimation

The IME module is composed of two Units. The first one is the TZS Unit, which is responsible for the decisions related to the TZS algorithm and for computing the SAD value of small $4x4$ sub-blocks. The second unit is the SAD Tables, which is responsible for composing the SAD value of all block sizes supported by the 3D-HEVC by accumulating the SAD value from the small sub-blocks and also for evaluating the best candidate to each block size. The next two subsections explain these two units in detail.

4.1.1 Test Zone Search Implementation

The TZS adopts the same modifications presented in Chapter 3. In other words, TZS uses only the zero predictor, the first search always expands to the whole search range, the search range in the raster step was reduced from the half, and also the consecutive refinements were limited to seven.

The control unit is mainly composed of a Finite State Machine (FSM) that adopts the TZS algorithm to individually selects the candidate blocks to be evaluated. This state machine is similar to the one presented in Figure 21, where each square represents a different state. In this figure, the states represented by red squares are responsible for individually selects all candidate blocks relative to each TZS step before passing to the next state. The states represented by blue dotted squares are responsible for waiting for the results of all candidates of its relative step (due to the pipeline stages of the architecture, as will be better explained in the sequence), and decides the next step to be applied based on the vector relative of the candidate with the smallest SAD value.

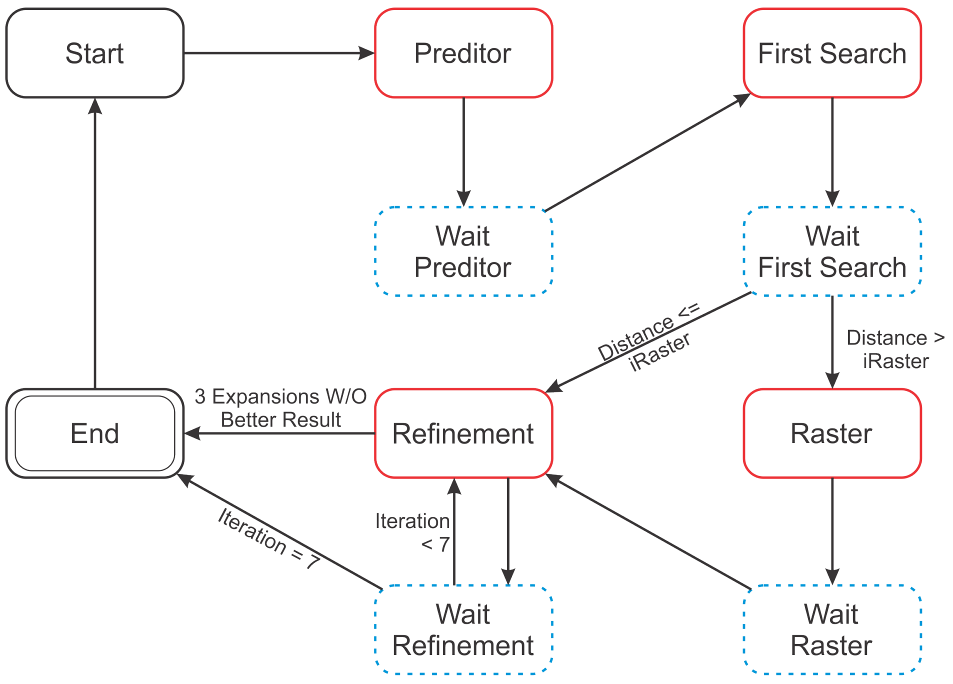

Figure 21 – Finite State Machine that process the TZS algorithm

The samples of the selected candidate are passed from the memory to the processing unit. The samples of the whole candidate could be passed to the processing unit at once. However, the architecture was developed to receives the samples of 1/4 of the candidate block at each clock cycle. This means that the control unit selects a new candidate to be processed at every four clock cycles.

The Sub-block Calculator starts by computing the absolute difference between the samples of the current block and the input samples of the candidate block being processed, as represented in Figure 22.

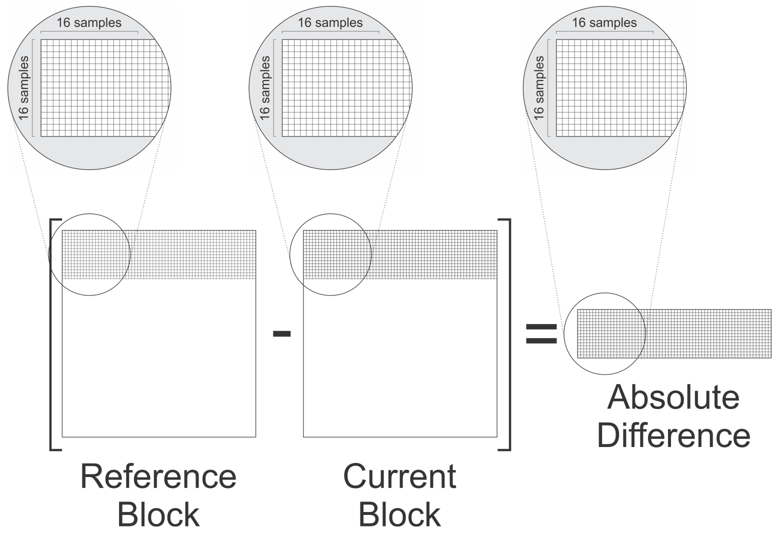

Figure 22 – Scheme for obtaining the Absolute Difference between Reference and Current Blocks

After obtaining the absolute differences, the values are accumulated together with the differences of the neighboring samples until it results in the SAD value of $4x4$ sub-blocks. This can be better understand in Figure 23, that presents the absolute difference between each sample of the candidate block, obtained by subtracting each sample of the current block by the correlated sample of the candidate block.

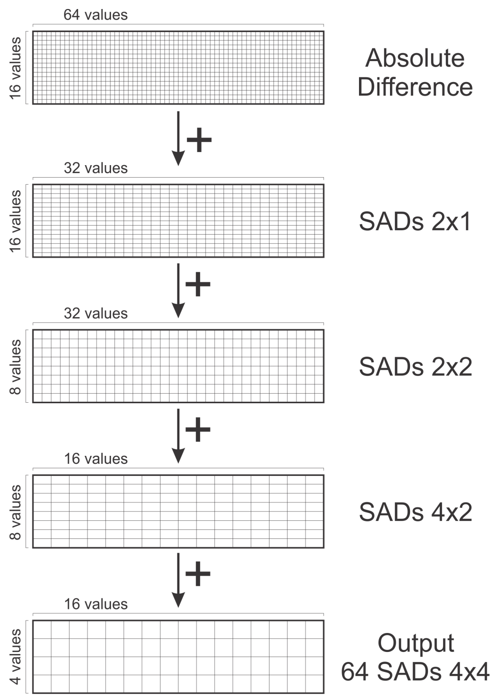

Figure 23 – Scheme that join the Absolute Difference between two neighboring samples to obtain the SAD of $4x4$ sub-blocks

After that, the results of the 64 sub-blocks of $4x4$ samples are passed to the SAD tables, which will accumulate the value of the several sub-blocks to compute the SAD value of the whole $64x64$ candidate block, as will be presented in the next subsection. In the sequence, the SAD of the whole $64x64$ block is passed to the control unit, that will process this result to take the decisions of the algorithm, as can be seen in Figure 20. In other words, this means that the control unit stores the vector related to the candidate block with the smallest SAD value of each TZS step. So, after the IME Unit processes all candidate blocks from a TZS step, the vector of the best candidate block will decide the next TZS step to be applied, and the center location of that TZS step (if applicable).

4.1.2 SAD Tables

The SAD Tables are responsible for accumulating the SAD value from smaller $4x4$ sub-blocks to compose the SAD value from all PU sizes. A diagram with an example of the processing of this unit can be seen in Figure 24. The left side of the Figure represents the accumulation from the SADs of neighboring $4x4$ PUs to compose the SADs of $8x4$ and $4x8$ PUs. A similar accumulation occurs to compose the SADs of larger PU sizes. These accumulations were divided into several pipeline stages, so only one adder operation was performed sequentially in each stage. Also, the processing of PUs with height higher than 16 samples requires the adoption of some registers to store the partial SAD value. These registers were necessary since the processing is performed on 16 lines per clock cycle, so partial SAD values need to be stored until computing the SAD of the rest of the block.

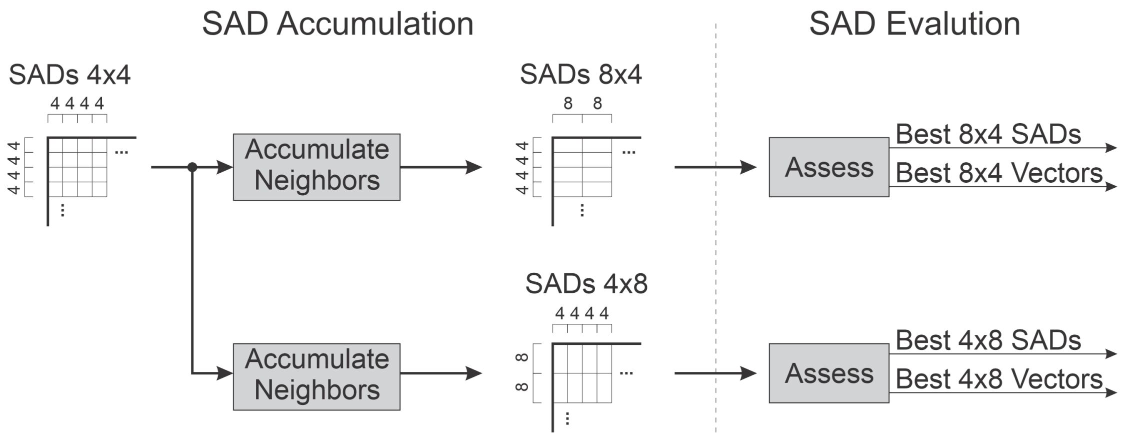

Figure 24 – Composing diagram of the SAD values from larger PU sizes

In the right side of the diagram from Figure 24, it is possible to observe the adoption of the Assess Unit, which is responsible for comparing the result from all candidates, while storing the Vector and SAD from the best candidate. The architecture of this Assess Unit can be seen in Figure 25. As can be seen, the SAD from the current candidate compared with the best SAD from previous candidates. The signal of this comparison is used to define which value will be stored. So, after the TZS algorithm ends its processing, the register from the several Assess Units has stored the best vector related to the candidate with the smallest SAD to all possible partitions. Thus, these vectors are then passed by the next Unit, the FME Unit, to be processed.

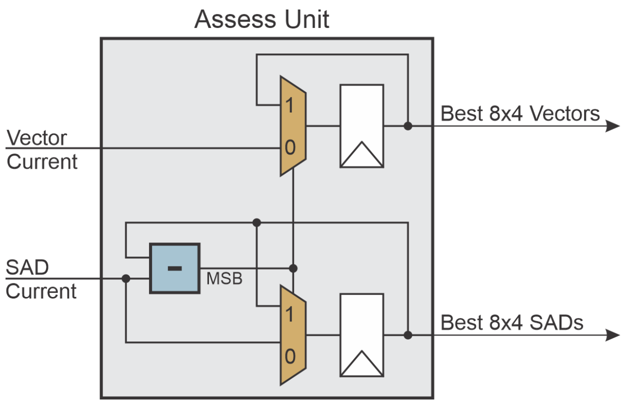

Figure 25 – Block diagram of one Assess Unit

In Figure 26, it is possible to observe a timeline from the processing of several PU sizes supported by SAD Table based on accumulating the SADs from the $4x4$ sub-blocks given as input. Although Figure 26 shows only the partitions required to obtain the SAD from the $64x64$ block, any other partition size not presented in this figure can also be computed following the same scheme.

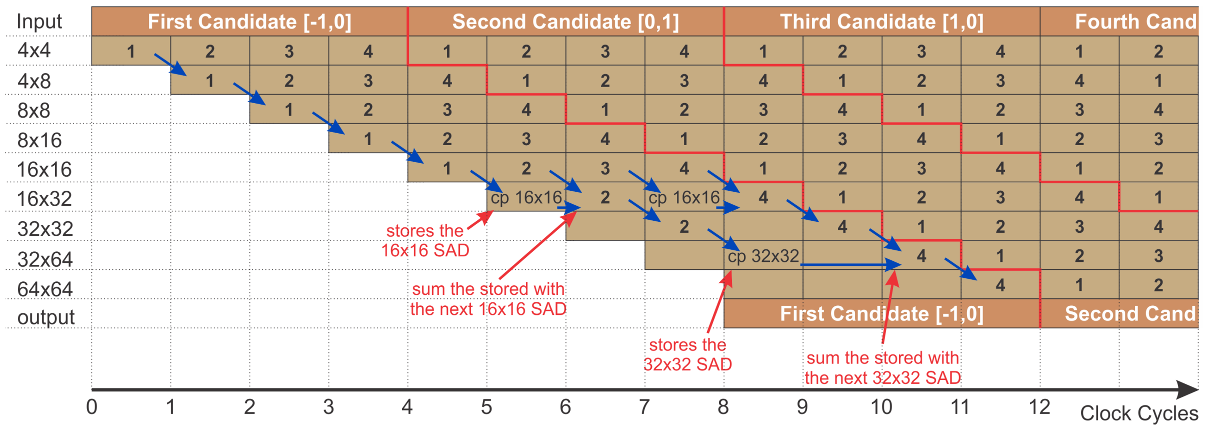

Figure 26 – Schedule from the processing of SAD values

As can be seen in Figure 26 it takes four clock cycles to obtain the SADs from $4x4$ sub-blocks of the first candidate (cycles 0 to 3). Then, by accumulating neighboring SADs from a $4x4$ sub-block, the $4x8$ SADs from the same candidate could be computed (cycles 1 to 4). By accumulating the neighboring $4x8$ SADs, the $8x8$ SADs from the same candidate could be computed (cycles 2 to 5). And so on, until computing the $16x16$ SADs (cycles 4 to 7).

However, since the IME process 16 lines per clock cycle, PUs with height higher than 16 samples requires the storing of the SAD from a smaller PU, to be further added with the SAD from another region of the smaller PU. For example, as represented by the red arrows in Figure 26, the $16x32$ PU requires the storage of the $16x16$ SAD from the first region on cycle 5, to be further added with the $16x16$ SAD from the second region on cycle 6. Similarly, the $32x64$ PU requires the storage of the $32x32$ SAD from the second region on cycle 8, to be further added with the $32x32$ SAD from the fourth region on cycle 10. Finally, the SAD from the $64x64$ block size is only computed after the $32x64$ block size is computed, eight cycles after the last region of the candidate are given as input.

The SADs computed for all PU sizes are processed by the Access Units as previously represented in Figure 24, which will result in the SAD and the Vector related of the best candidate evaluated to each partition. These results of the Assess Units are ready for being used by the RD cost processing in the next steps of the encoder. However, this work also aims an integration to an FME, since the 3D-HEVC has support for fractional motion vectors through the FME processing. Therefore, the next section presents a possible FME unit to be integrated with the IME presented above. This FME will process those results given by the Access Units, performing a refinement around the region pointed by each vector as already explained in Section 2.3.2.

4.2 Fractional Motion Estimation

The FME architecture is divided into two units. The first one is the Interpolation Unit, which is responsible for generating the new samples at fractional positions using the integer samples from the SA. The second unit is the Evaluation Unit, which is responsible for evaluating all fractional blocks generated around the IME vectors. These fractional blocks are composed of the new samples at fractional positions from the Interpolation Unit. The next two topics from this subsection explain in detail the architecture of these two units.

Different strategies could be adopted for FME processing: 1) Interpolate only the locations pointed by an IME result vector, which can result in several redundancies in the interpolation process if two IME vectors are pointed to similar regions of the SA. This redundancy was represented in Figure 27, where two $8x8$ blocks to be interpolated are represented by blue and yellow squares, while the redundancy is represented by green squares, since those samples will be interpolated for both blocks. 2) Firstly interpolate the whole SA and store the fractional blocks interpolated, and after evaluates the fractional blocks composed by the interpolated samples. This solves the redundancy problem of FME, but it requires a huge number of registers to store the interpolated samples of the whole SA.

Although, the adopted strategy for the FME of this work is to evaluate all fractional blocks to all 24 PU sizes that are dependent on a specific fractional sample, in parallel, as soon as this specific sample is interpolated. This strategy of evaluating all fractional blocks to all PU sizes is based on the idea of obtaining the best encoding efficiency of

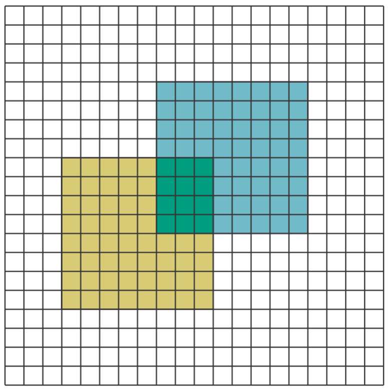

Figure 27 – Redundancy that can occur in the interpolation when two IME vectors are similar

the ME tool since the developed IME hardware will result in the motion vectors to all 24 PU sizes. However, the adopted strategy is not a practical solution for real applications due to the high number of hardware resources needed to evaluate all fractional blocks in parallel. Therefore, in the next works in the future, the adopted FME strategy will be better explored, as by adopting some registers to reduce the parallelism of the FME, or by limiting the PU sizes supported for fractional blocks.

4.2.1 Interpolation Unit

As mentioned before, the Interpolation Unit is responsible for generating the new samples at fractional positions by using the Integer samples from the SA. For this, the SA from $192x192$ samples was divided into nine regions from $64x64$ samples, so each region is processed sequentially. Thus, the interpolation filters were reused to perform the interpolation of the whole SA.

From each region of the SA, a set of samples composed of a whole line or column from the current region is selected to be interpolated. As previously explained in Section 2.3.2, three fractional samples could be generated between each two integer neighboring samples. Thus, to process all 64 input samples in parallel, the interpolation is performed by $64 + 1$ sets of filters, similar to the ones developed by AFONSO et al. (2016b). Each set of filters was composed of three types of filters, depending on the position of the fractional sample to be computed. One filter computes the sample at half position, and the other two filters compute the samples at quarter positions. This results in a total of $3 * (64 + 1)$ interpolation filters. Specific details of the development of those filters could be found on AFONSO et al. (2016b).

Figure 28 represents the position of the new fractional samples to be computed. As can be seen, these new samples can be categorized into horizontal, vertical, and even diagonal samples, related to the position of the integer sample. So, the processing was also divided into these three categories.

Figure 29 presents the architecture of the Interpolation Unit, where the selection of

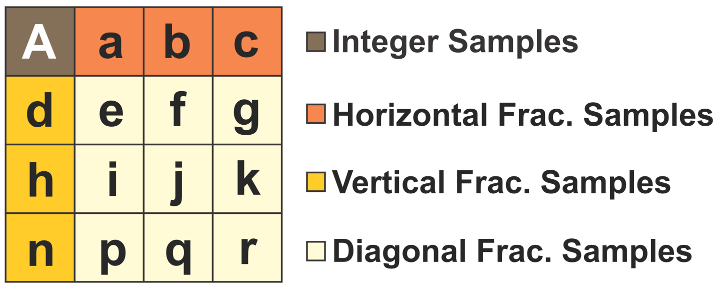

Figure 28 – Fractional sample distribution around the integer sample
Source: Adapted from AFONSO et al. (2016a).

the input samples to be interpolated, the horizontal buffer, and also a Clip operation can be seen. Firstly, the interpolation starts by processing the horizontal fractional samples. So, all samples from the current region of the SA were selected as input, one line at each clock cycle. Then, the selected line is passed to the $64 + 1$ sets of interpolation filters, to generate the new fractional samples.

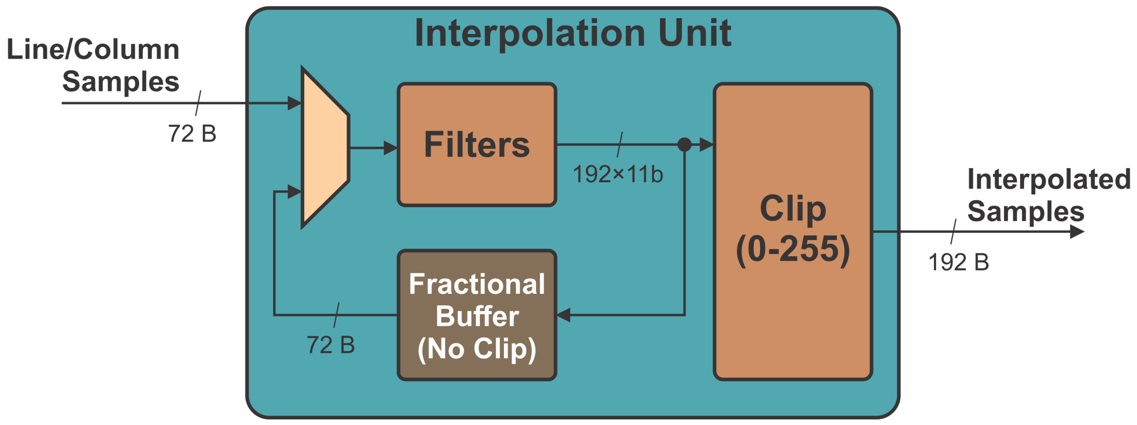

Figure 29 – Architecture of the Interpolation Unit
Source: Adapted from AFONSO et al. (2016a).

For the diagonal processing, the horizontal buffer must store the horizontal fractional samples, and also the horizontal fractional samples of upper and bottom edges (see section 2.3.2). Thus, for the horizontal buffer, an internal buffer with $3 * (64 + 1)x(64 + 8)$ registers are required to store all these horizontal samples.

Also, the outputs of the filters are connected to the Clip operation, the last step of the Interpolation Unit. This Clip operation ensures that all samples that compose the new fractional blocks have 8 bits, that is, samples with value less than zero are converted to zero, while samples with values higher than 255 are converted to 255. The values between 0 and 255 are unchanged. The architecture from the Clip operation is presented in Figure 30, where the results from the three higher significant bits of the input define the conversion. Finally, after the Clip operation, the new interpolated samples are used to compose the new fractional blocks to be compared with the current block in the Evaluation Unit, as will be explained in the next subsection.

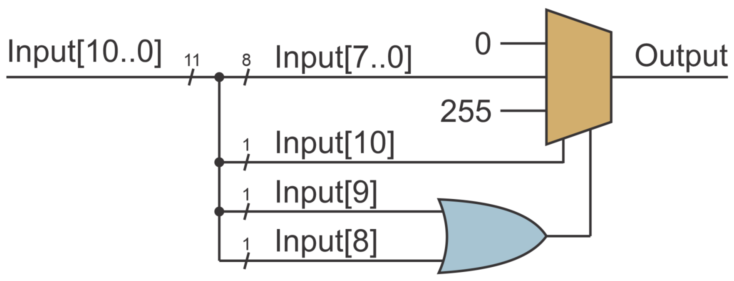

Figure 30 – Architecture of the Clip operation

After interpolating the horizontal fractional samples from all lines of the current region of the SA, the processing passes to the vertical fractional samples. The processing of the vertical samples is similar to the processing of the horizontal samples. Although, the region of the SA is processed column by column. Every selected column is interpolated using the interpolation filters and, after, the new samples are connected to the clip operation, to be processed by the Evaluation Unit.

Finally, after the processing of all columns from the current region of the SA, the processing of the diagonal fractional samples is performed. For this, the horizontal samples stored on the horizontal buffer are selected from the internal multiplexer to be used from the interpolation filters. Since there are three categories of diagonal samples (as can be seen in Figure 28), the processing of the diagonal samples is done three times, but selecting different samples from the horizontal buffer.

After the diagonal fractional samples are processed, the Interpolation Unit ends the processing of the current region. Then, it idles some clock cycles (exact value in the sequence). Only after the Evaluation Unit ends the generation and comparison of the new fractional blocks, the Interpolation Unit starts the processing of the next region from the SA by repeating its processing.

4.2.2 Evaluation Unit

The Evaluation Unit is responsible for processing the IME vectors and the interpolated samples. It composes the fractional blocks related to the IME vector and, then, applies the similarity criterion over these fractional blocks.

Naturally, it is expected that one Evaluation Unit could process more than one IME vector, mainly if those vectors are related to different regions of the SA. However, as previously mentioned, the ME architecture was developed with data reuse techniques to avoid redundancy in the processing. By the exception of the horizontal samples, any other interpolated sample is stored in order to improve the processing. So, as soon as the Interpolation Unit processes a line or column of fractional samples, the similarity value from all dependent blocks (fractional blocks composed by the generated fractional samples) must be processed at the same time.

Moreover, it is impossible to implement an Evaluation Unit able to be reused for processing different blocks. Considering that all IME vectors can be extremely similar to each other, all fractional blocks must be composed and evaluated at the same time when the interpolation filters finish the interpolation of the associated samples. Therefore, the FME Unit requires a total of 593 Evaluation Units from different sizes, being that each Evaluation Unit is responsible for processing only one IME vector.

The architecture from the Evaluation Unit is presented in Figure 31, where each unit has 12 SAD Processing Units, 48 registers to store the SAD from all 48 fractional blocks, and a comparator that results in the vector relative to the block with the smallest SAD value. The internal buffer that stores the current block can also be seen in Figure.

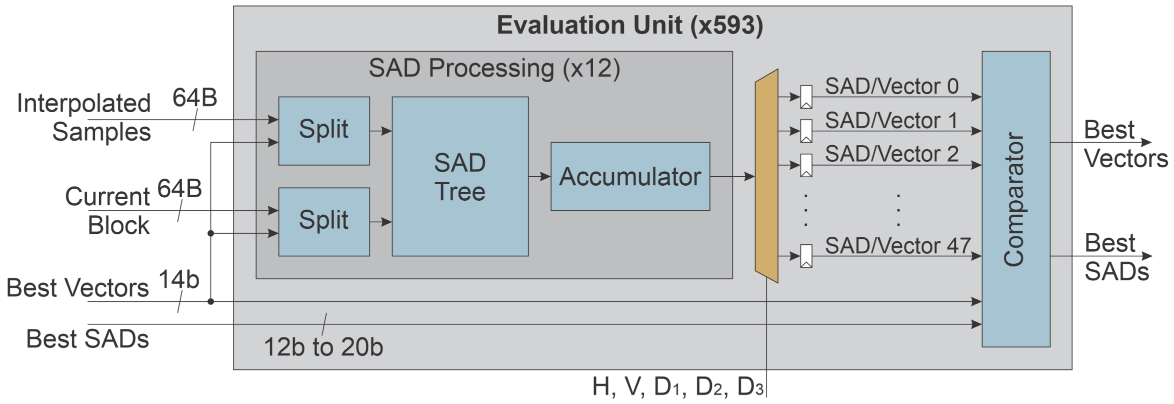

Figure 31 – Block diagram of the Evaluation Unit

A total of 12 SAD Processing Units are required due to a total of 48 fractional blocks that can be processed. Although the Processing of the Horizontal and Vertical fractional blocks requires six SAD Processing Units each, the processing of the Diagonal fractional blocks requires twice the SAD Processing Units. This occurs because the diagonal fractional samples require to be evaluated considering two side-neighboring integer samples.

As can be seen in Figure 31, the SAD Processing Unit has three main modules, which were the Block Split, the SAD Tree, and the Accumulator. The SAD Processing Unit receives as input the line or column composed by the interpolated fractional samples, and it has also access to the internal buffer to obtain the samples from the current block. So, these input samples pass by the Block Split, responsible for selecting only the interpolated samples and current block samples related to the block pointed by the IME vector. The Split was necessary since the interpolated samples received by this unit are a line or column from a complete region of the SA, while the block pointed by the IME vector can be related to a block that occupies two different regions of the SA, so only the samples related of this block will be selected for comparison with the samples of the current block.

After the selection of the samples to be compared, the SAD value is computed. As can be seen in Figure 31, the SAD Tree is responsible for computing the SAD value from the line or column being interpolated firstly. After the SAD Tree computation, an Accumulator stores the SAD value from the current line and accumulates it with the SAD value from the next lines. After processing all lines, the SAD value is stored to be further compared with the SAD from other fractional blocks.

Figure 32 presents the architecture of a SAD Tree with four samples range, which can obtain the SAD value of a line or column with a maximum of four samples. The SAD Tree of Figure has two pipeline registers, that perform only one arithmetic operation in each pipeline stage. The SAD Trees with a higher sample range, which can process higher lines or columns, have more input samples and consequently more consecutive arithmetic operators. Therefore, it requires the adoption of more pipeline registers to maintain a high throughput.

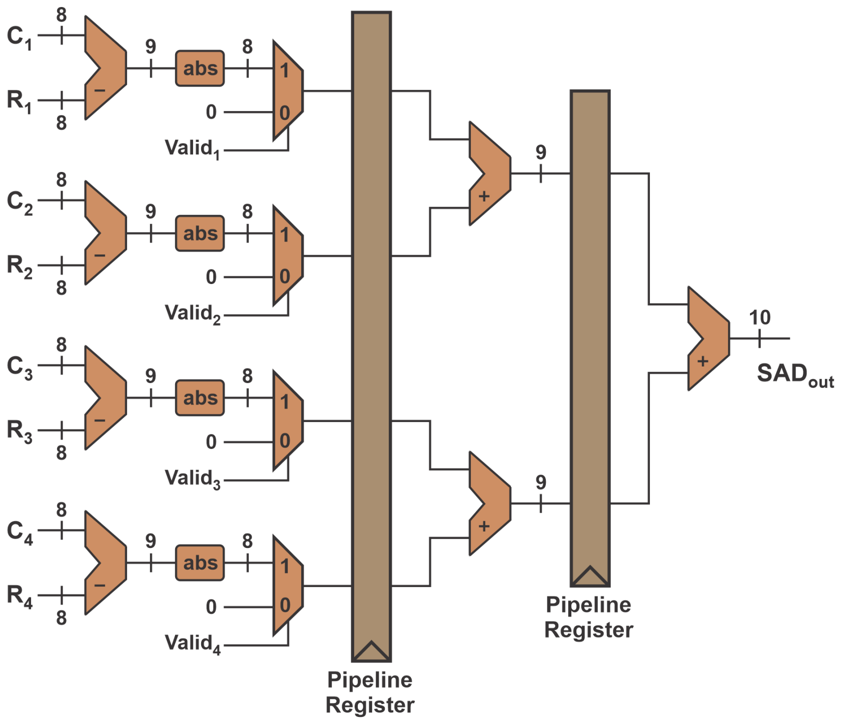

Figure 32 – The architecture of a SAD Tree with four samples range

As will be mentioned before, there are some cases when the SAD Trees do not receive the total number of samples. In this case, some SAD Tree inputs must not be considered in the output value. So, an array containing several valid bits were required as input by the SAD Tree, which can also be seen in Figure 32. This array is used to indicate what input samples are the valid ones and must be considered in the output value.

Since the SAD Trees compute only the SAD value of one line or column at each clock cycle, each Evaluation Unit requires an accumulator to store the SAD Tree output until all fractional block is processed. The architecture of one Accumulator is presented in Figure 33, where each Accumulator has a multiplexer, an adder, and a register. The multiplexer is responsible for resetting the accumulated value, or even, it allows the sum of the stored value with the SAD Tree output value. The registers store the adder output value. So, the Accumulator is restarted whenever a new block starts to be processed, and it keeps accumulating its input values until the processing of all lines of the fractional block being processed.

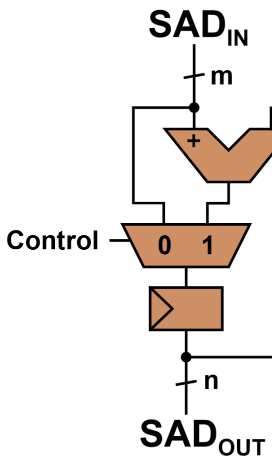

Figure 33 – The architecture of one Accumulator Unit
Source: Adapted from AFONSO et al. (2016a).

When the Interpolation Unit ends the processing of any region, the Accumulator output has the SAD value from the block pointed to the IME vector, if the IME vector was pointed to that region. Thus, the Accumulator output is stored in one of the 48 registers from the Evaluation Unit. An exception can occur in this processing when the block pointed by the IME vector belongs to more than one region from the SA. In this case, the result from the Accumulator is the SAD value from the part of the IME vector block that belongs to the current region being processed. This way, the Accumulator output must be added with the respective value from one of the 48 registers at the end of the processing of each region. Therefore, these registers have the complete SAD value from the IME vector block at the end of the processing of all SA.

As can be seen in Figure 31, all 48 registers are connected to the SAD Comparator, which is enabled after the interpolation of all regions from the SA. This SAD Comparator is responsible for comparing all SAD values, aiming to result in the motion vector of the fractional block with the smallest SAD value. This unit also receives the SAD value and its motion vector from the best IME candidate. Thus, the SAD Comparator can also compare the best IME block with the best fractional block. The architecture of a SAD Comparator is presented in Figure 34. It has several Decide Two Units that compares the SAD values two by two. Between each sequential comparison, pipeline registers are adopted, aiming to reach higher throughput.

The Decide Two Units were adopted to evaluate two SAD values and decide what is the smallest one. The architecture of this Decide Two Unit can be seen in Figure 35.

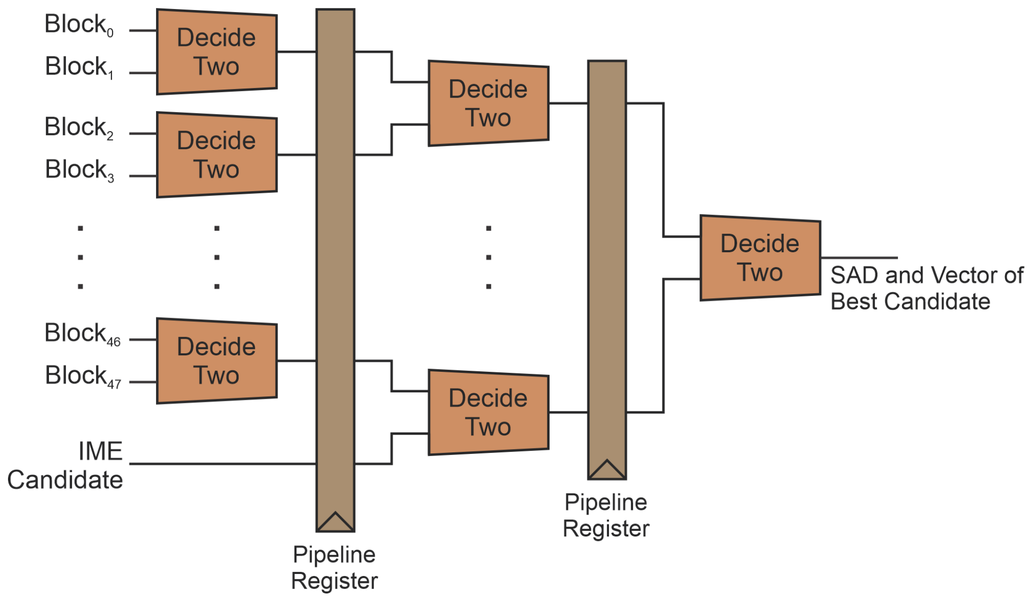

Figure 34 – The architecture to process 49 SAD values and its respective motion vectors

This architecture performs a subtraction between the two SAD values, and so the Most Significant Bit (MSB) of the result indicates what is the smallest one. Then, the MSB is used as a control signal of two multiplexers: one selects the smallest SAD value, while the other selects the motion vector associated of the smallest SAD value.

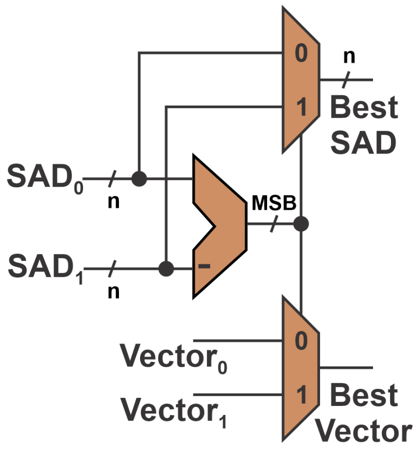

Figure 35 – Architecture of the Decide Two Unit
Source: Adapted from AFONSO et al. (2016a).

As mentioned before, the Evaluation Unit was specialized in processing only one PU size. The main difference between the units from different PU sizes is the bit width of the connections between the units, and also the SAD Tree Units, which must process a different number of samples in parallel. For the square-shaped PU sizes, the range of the SAD trees is equal to both width and height. However, for the non-square-shaped PU sizes, the range of the SAD trees are defined by its highest side, aka, the highest value between the width value and the height value.

This way, for processing a PU where the width is higher than the height, as the processing of an $8x4$ PU, an SAD Tree with eight samples range is required. When processing this PU line by line, the complete SAD Tree is used, while when processing this PU column by column, half of the SAD Tree is disabled using clock-gating techniques, which also helps to save power dissipation.

4.3 IME and FME Synchronism

This subsection explains in details the synchronism between the IME and the FME modules, and how the several CTUs from a video sequence are processed.

Figure 36 presents the timeline of the IME Unit schedule considering its best scenario in a SA of $192x192$ samples and a CTU size of $64x64$ samples. In this case, the TZS processing requires 337 clock cycles, assuming that although the First Search is performed, the best result is the co-located block evaluated at the Predictor step. Thus, the Raster and the Refinement steps won't have been executed.

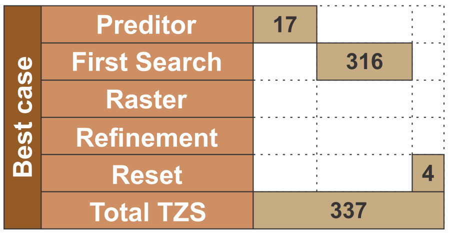

Figure 36 – Timeline of the best case of the IME Processing in a SA of $192x192$ samples

By another side, the Figure 37 presents the timeline of the IME Unit considering its worst scenario, where the TZS processing requires 3237 clock cycles to process a SA of $192x192$ samples and a CTU size of $64x64$ samples. This case assumes that the Raster step is performed after the First Search, and seven Refinements are performed so that every Refinement has found a better result than the previous Refinement.

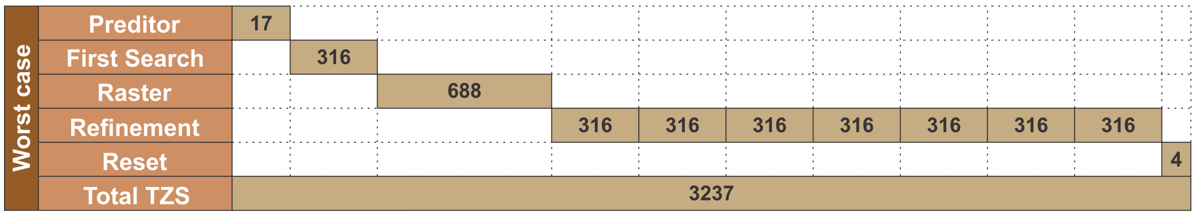

Figure 37 – Timeline of the worst case of the IME Processing in a SA of $192x192$ samples

Figure 38 shows the timeline from the FME processing. As can be seen, the FME always requires 3680 clock cycles to process the whole SA, where FME spends 408 clock cycles to process each region of the SA, plus one cycle between regions to reset the necessary modules from the FME. On the bottom of the Figure, the distribution of these 408 clock cycles between the FME Units among the five groups of fractional samples can be seen. To interpolate the group of Horizontal, Vertical and Diagonal samples, the Interpolation Unit requires 72, 64, and 65 clock cycles, respectively. Since the interpolation filters have three pipeline registers, the Evaluation Unit idles by three clock cycles before the processing starts. Moreover, the processing of the Horizontal samples requires more clock cycles than other samples due to the edge required to process the Vertical samples, as mentioned in subsection 2.3.2. Also, although the Interpolation Unit process 72 lines of horizontal samples, only 64 lines are considered by the Evaluation Unit, being the first four lines ignored by the Evaluation Unit. Finally, after processing any group of fractional samples, some controls of the Evaluation Unit was restarted, which requires one clock cycle per reset.

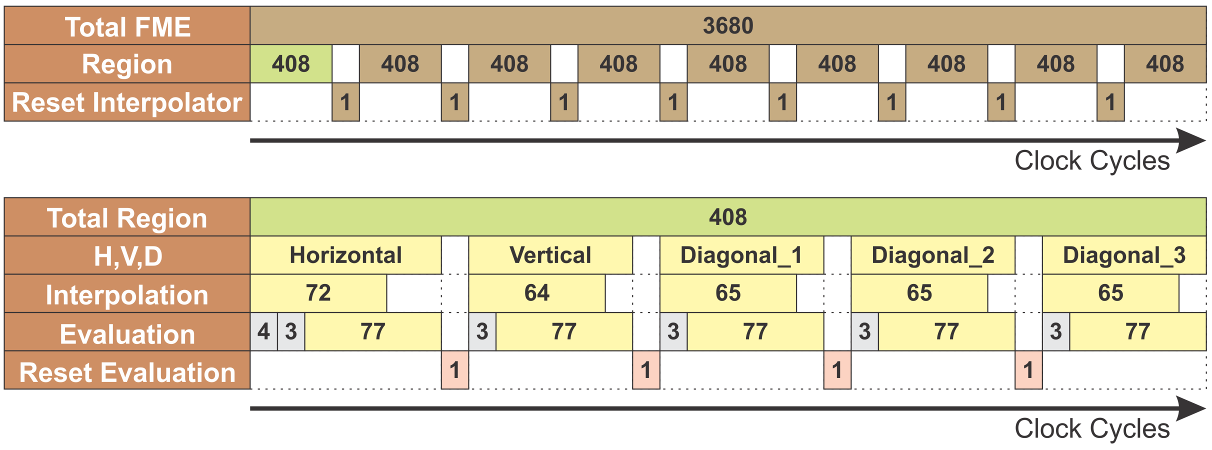

Figure 38 – Timeline from the FME Processing in a SA of $192x192$ samples

Figure 39 presents a timeline from the processing of both IME and FME modules considering the processing of three views of an FHD video at 30 frames per second. As can be seen, the processing of each CTU requires from 4017 to 6917 clock cycles, according to the best and worst scenario of the IME Unit. Considering all 510 CTUs from an FHD frame, the architecture requires from 2.05M to 3.5M clock cycles to process each FHD frame. Moreover, considering that the ME encoding tool was used in both channels (in Texture and Depth Maps), and also the processing of three views, the complete ME architecture requires from 368.7M to 634.9M clock cycles per second, which demands a frequency of 634.98 MHz, considering the worst-case scenario of the IME Unit.

When considering only the IME unit, disregarding the FME processing, the processing of each CTU requires up to 3237 clock cycles. Figure 40 present the timeline of the developed IME Unit for the processing of three views of an FHD video at 30 frames per second. As can be seen in Figure 40, considering the 510 CTUs of each frame, the processing of 30 frames of the temporal resolution, the two channels, and the three views, the IME unit requires the frequency of 297.15 MHz, considering the worst-case

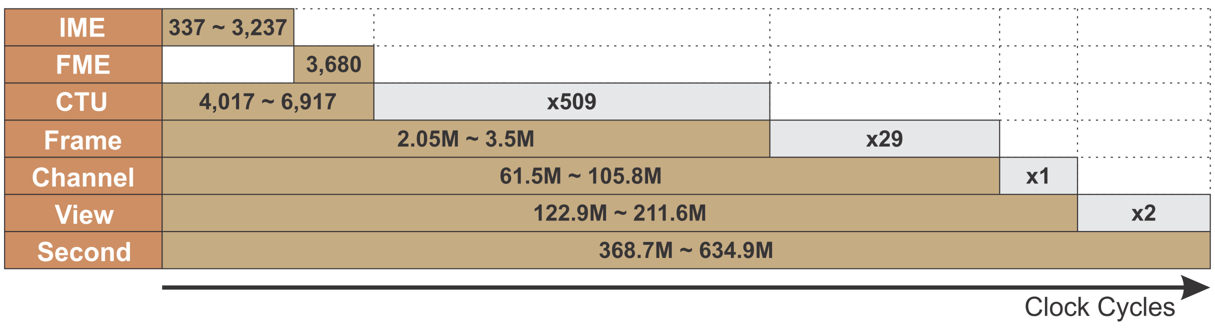

Figure 39 – Timeline from the Complete ME Architecture when processing FHD 1080p videos at 30 fps considering 3 views in the MVD format

scenario of the TZS algorithm.

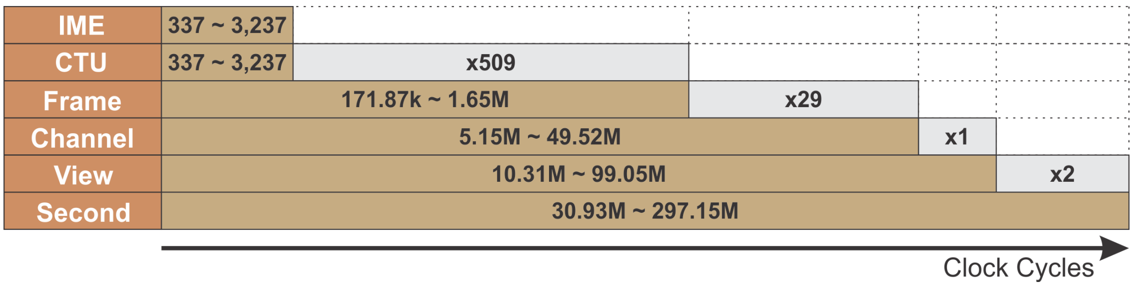

Figure 40 – Timeline from the IME Unit disregarding the FME processing when processing FHD 1080p videos at 30 fps considering 3 views in the MVD format

4.4 Memory Management

This work was focused on the development of the processing unit from the hardware architecture, as presented above. Details of the memory communication of this processing unit were presented below. Although, it should be noted that this memory communication was superficially evaluated among its viability in a real application. In future works, this memory will be better explored and optimized, if necessary.

As previously mentioned, the ME encoding tool requires a high memory communication to process 3D videos with high-resolution, since it requires the samples from several candidates from the Search Area. This was a problem for many hardware designs that evaluate the several block sizes allowed by the 3D-HEVC independently. However, since this work adopted a scheme that all block sizes are processed together, the memory communication can be highly reduced than if processing all block sizes independently, as presented in section 2.6. The next subsection presents the memory communication necessary for each unit from the developed architecture.

4.4.1 Integer ME Unit

The IME Unit processes the higher block sizes inheriting the result of smaller sub-blocks. As mentioned before, the IME Unit requires the data from 64 sub-blocks of $4x4$ samples at each clock cycle for obtaining the result of the smaller sub-blocks. Considering that the required data comprises the samples of both the Search Area and the Current Block being processed, and also that each sample has 8 bits width (1 Byte), this results that the IME Unit requires 2,048 Bytes of data per cycle.

4.4.2 Fractional ME Unit

The adopted scheme for the FME Unit interpolates a set of samples from a region of the SA and evaluates all dependent block sizes in parallel. Considering that the set of samples from a region to be interpolated contemplates the total of 72 samples (64 + 8 due to the edge of samples required for the interpolation), the Interpolation Unit requires 72 Bytes of data per clock cycle.

As previously mentioned, 593 Evaluation Units were adopted in the FME architecture due that each CTU has a total of 593 PUs from different sizes. If each Evaluation Unit adopts a different connection to the external storage for obtaining the samples of the current block, a high bandwidth will be required to obtain all these samples. As an example, considering only the square-shaped PU sizes, this represents 64 samples per cycle for processing the $64x64$ Evaluation Unit, $2*2*32$ samples for processing the $32x32$ Evaluation Units, $4*4*16$ samples for processing the $16x16$ Units, and $8*8*8$ samples for processing the $8x8$ Units, resulting in the total of 960 samples per clock cycle. Considering all PU sizes supported by the architecture, this strategy may result in a huge requirement of at 8,384 samples per clock cycle. Therefore, for reducing this huge memory communication of obtaining the samples of the current block to be used in the Evaluation Unit, an internal buffer was adopted to store the current block, being that this internal buffer is filled by the samples requested in the TZS Predictor step. Thus, any memory communication with the SRAM is necessary for obtaining the samples of the current block.

4.4.3 Complete ME Unit

As previously mentioned, the IME Unit requires 2,048 samples per clock cycle, while the FME Unit requires only 72 samples per clock cycle. Figure 41 presents the memory scheme required for the developed ME architecture, considering this bandwidth required for the IME and FME Units and considering that each sample has 8 bits (MÜLLER; VETRO, 2014).

When considering only the IME Unit, disregarding the FME, 3,237 clock cycles were necessary to process each CTU, thus requiring 297.15MHz of operational frequency to process 3 views of an FHD 1080p video with 30 fps, as presented in section 4.3.

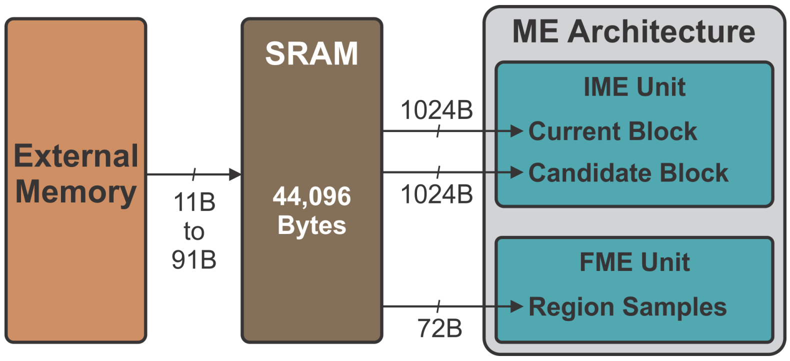

Figure 41 – Memory scheme required for the developed architecture

Moreover, in each of those clock cycles, 2,048 Bytes of data are required for the IME processing. Thus, for the frequency of 297.15MHz, the IME Unit requires a sustained memory bandwidth of 608.57 GB/s.

Now, considering both IME and FME units, the ME architecture was scheduled so that the FME was performed sequentially after the IME, while the IME is idle, thus requiring a total of 6,917 clock cycles to performs the ME to each CTU, as presented in Section 4.3. So, the developed architecture requires the maximum storage access of 2,048 Bytes per cycle (given by IME processing). Considering that the developed ME architecture requires an operating frequency of 634.98 MHz, it requires a memory throughput of 1300.440 GB/s to process three views of FHD 1080p videos at 30 fps.

Therefore, to supply this high memory communication and feed the samples required when adopting both IME and FME Units, or even adopting only the IME Unit, an SRAM memory was required between the ME architecture and the External Memory, as can be seen on Figure 41. These SRAM stores a total of 44,096 Bytes since it was used to buffer the samples of both the current block and from the SA.

The complete 44,096 Bytes of the SRAM buffer must be updated with the samples of the new current block and its related SA at each new CTU to be processed. Although, neighboring CTUs share up to 2/3 of the samples from the Search Area. Therefore, considering the Level-C scheme (Ching-Yeh Chen et al., 2006), only the 13,333 new samples of the Search Area, and the 4,096 samples of the current block, needs to be updated.

When considering only the IME unit, disregarding the FME, the update of the 17,429 new Bytes of the SRAM could occur in two forms: 1) By dedicating some clock cycles to perform the update between the end of the processing of one CTU and the start of the processing of a new CTU, which may require a small increase in the operational frequency, or 2) By performing the update while the TZS performs the lasts iteration of the Refinement Step when the samples of the SA can be discarded. Thus, in both forms, considering updating all 17,429 new Bytes in 192 clock cycles, a bandwidth fewer than 91 Bytes per cycle are sufficient for supply the SRAM buffer with its new samples.

When considering both IME and FME Units presented, this data update can be performed in parallel to the FME processing, due to the adoption of the internal buffer to store the current block. This way, after the FME process each region, its related samples of the region can be discarded and updated with its new 4,096 samples. Since each region was processed in 408 clock cycles, these 4,096 samples can be updated in less than 11 Bytes per cycle.

After presenting the developed ME architecture, the next chapter presents the results of the developed architecture and compares it with the results of related works from the literature.

5 RESULTS AND COMPARISONS

The developed architecture was described in VHDL and the ASIC was synthesized for 40nm TSMC standard-cells technology using the Cadence RTL Compiler. The obtained power results presented below considers the total power dissipation, composed by the sum of leakage and dynamic power. The dynamic power results were computed considering the default switching activity of the Cadence RTL Compiler tool, which corresponds to a transition of 20% in each input of architecture (PERLEBERG et al., 2018). The area results presented are equivalent to the number of NAND-2 gates. For that, it was computed the quotient between the cell area and the size of one NAND gate with two inputs. Thus, the number of NAND-2 gates can be fairly compared with the results of other works that adopt different technologies.

Two synthesis processes were performed. The first process considers only the IME Unit, disregarding the FME, while the second process considers both IME and FME. The next two subsections discuss, in detail, the results of each synthesis process.

5.1 IME Unit Results and Comparisons

The first process considers the adoption of only the IME Unit. By disregarding the FME, the proposed IME algorithm results in an impact in the encoding efficiency of 17.93% and 17.712% considering the BD-rate metric for NCO and FCO schemes, respectively, as evaluated in Chapter 3. When adopting an FME Unit, like the one presented in this paper, this BD-rate impact can be reduced to up to 8.945% and 8.766% considering NCO and FCO schemes, respectively.

The synthesis results show that the developed architecture can reach a maximum frequency of 2.377 GHz. Considering that the IME requires a maximum of 3237 clock cycles to process each CTU, the developed IME can process up to three views of a UHD 2160p video at 60 frames per second.

Therefore, the synthesis was performed considering three different target frequency: 1) at the maximum frequency of 2.377GHz, with which the architecture can process up to three views of a UHD 2160p video at 60 frames per second. 2) at the frequency of 297.15MHz, with which the architecture can process three views of an FHD 1080p video at 30 frames per second, the major frequency of the 3D-HEVC Common Test Conditions (MÜLLER; VETRO, 2014). 3) at the frequency of 990.52MHz, an average between the two frequencies above, with which the IME architecture can process five views of an FHD 1080p video at 60 fps. The results of these three synthesis are presented in Table 8, which shows the target frequency and its respective target resolution, and the results among the Cell Area measured in the equivalent number of NAND-2 gates, and the total power dissipation.

Table 8 – Synthesis results according to each synthesis performed and target frequency

Target Frequency	Target Resolution	Cell Area (k gates)	Total Power (mW)
297.15MHz	3 Views 1080p @30fps	269	108.48
990.52MHz	5 Views 1080p @60fps	269	247.89
2.377GHz	3 Views 2160p @60fps	280	312.81

As can be seen in Table 8, the developed 3D-HEVC IME Unit requires 269k gates, with a total power dissipation of 108.48mW when running at 297.15 MHz. When running at 990.52MHz, the IME architecture has a total power dissipation of 247.89mW. Yet, when running at its maximum frequency of 2.377GHz, the developed IME dissipates 312.81mW. Moreover, can be seen by the results in Table 8 that for reaching its maximum frequency, the developed IME architecture requires 4.09% more NAND-2 gates.

Table 9 presents the results of the developed hardware along with the results of other works proposing hardware architectures only for the IME encoding tool, disregarding the FME tool. It should be noted that all these related works were focused on the HEVC encoding standard. Thus, for a fairer comparison, the throughput of the developed architecture considering the processing of only the texture channel of one view.

As can be seen, MEDHAT; SHALABY; SAYED (2015) and LIAO; SHEN; TSENG (2019) have proposed architectures that adopt an algorithm that shares similarities to the Full Search algorithm. Both these works propose a search range reduction, thus requiring a small SRAM size than the developed hardware. However, the developed hardware requires fewer area resources than these two works, even compared

Table 9 – Comparative results for IME encoding tool considering only the processing of texture channel

Related Works		MEDHAT (2015)	LIAO (2019)	GU (2019)	FAN (2018)	Developed Hardware
Video Coding Standard		HEVC	HEVC	HEVC	HEVC	3D-HEVC
BMA Algorithm		Full Search	Full Search	Diamond	Modified TZS	Master-Slave TZS
Search Range		-27+27	-32+32	-16+16	-64+64	-64+64
SRAM size		10kB	8kB	6.75kB	147.20kB	40.96kB
ASIC Technology		65nm TSMC	90nm TSMC	65nm TSMC	65nm[1]	40nm TSMC
Supported PU Sizes		All	Square & Symetric	Square	All	All
Area (Gates)		454k	274.5k	225.7k	489.4k	269k
Maximum Throughput		2160p 30fps	2160p 30fps	2160p 139fps	2160p 30fps	2160p 150fps
2160p 30fps	Frequency (MHz)	720	100.5	500	500	198.1[2]
	Power (mW)	N.A.	N.A.	N.A.	128.5	81.74

[1] Standard-cells library was not mentioned in the paper

[2] The MVD format was disabled

N.A. - Not Available

with LIAO; SHEN; TSENG (2019), which disregards the processing of Asymmetric PU sizes. Both MEDHAT; SHALABY; SAYED (2015) and LIAO; SHEN; TSENG (2019) do not present the power dissipation results of its proposed architectures.

The work GU et al. (2019) has presented an architecture that adopts a BMA algorithm following a Diamond scheme. It adopts a Search Range of only 16 samples around the center of the SA, thus it requires a smaller SRAM than the developed hardware. It requires a smaller gate count than the developed hardware, and it processes only Square-shaped PUs. Moreover, GU et al. (2019) can process 139 fps of UHD 2160p videos, while the developed hardware can process 150 fps for the same resolution. Also, GU et al. (2019) does not present the power dissipation results of its proposed hardware.

The hardware proposed by FAN et al. (2018) adopts an algorithm similar to the expansive search of the TZS algorithm, requiring an SRAM three times higher than the developed hardware. It also supports the processing of all PU sizes of HEVC. However, it requires more area resources, and it dissipates a power 57,2% higher than the developed hardware when processing videos with the same resolution. Moreover, the developed hardware reaches a maximum throughput 5 times higher than FAN et al. (2018).

5.2 Complete ME Results and Comparisons

In the second synthesis process, the complete ME architecture presented was considered. The architecture was divided into six parts to perform the synthesis. The first part is composed by the IME Unit and the Interpolation Unit, since these units define the bottleneck of processing each CTU, while the other parts are a set of Evaluation Units according to the type of the PU supported by the Evaluation Unit, i.e., the other parts were divided into the Evaluation Units: Square-shaped, Symmetrical-Horizontal, Symmetrical-Vertical, Asymmetrical-Horizontal, and Asymmetrical-Vertical. The synthesis results of these two processes were presented in Table 10, which shows the Area results, measured in the number of NAND-2 gates, and the total Power dissipation obtained in each synthesis, according to the target frequency.

Table 10 – Synthesis results of the complete ME architecture to the target resolution

Target Resolution	**Synthesized Unit**	**Cell Area (k gates)**	**Total Power (mW)**
3 Views 1080p @30fps (634.98MHz)	IME Unit & Interpolation Filters	1,699	358.55
	Square	15,743	1,577.32
	Symmetrical Horizontal	29,176	2,824.78
	Symmetrical Vertical	30,480	3,164.01
	Asymmetrical Horizontal	24,786	2,345.39
	Asymmetrical Vertical	26,882	2,604.78
	Complete (IME & FME)	128,766	12,874.83

Considering the complete ME architecture running at 634.98MHz, frequency necessary to process 3 views of an FHD 1080p video at 30 fps, the IME Unit and the Interpolation Filters require 1,699k gates and dissipates 358mW, as can be seen in Table 10. Table 10 also shows that the set of Square Evaluation Units require 15,743k gates and dissipates 1.577W, while the other sets of Evaluation Units require between

24,786k to 30,480k gates, with a total power dissipation between 2.345W to 3.164W. Therefore, the complete IME and FME architecture requires 128,766k gates, with a total power dissipation of 12.873W.

Table 11 presents the results of the complete ME architecture, along with the results of other works for the HEVC ME encoding tool. Again, the throughput of the developed architecture was normalized to the processing of only the texture channel and only one view to allow a fair comparison with the related works. It should be emphasized that the presented FME is a primary architecture, implemented only to the ME reaches its best result considering the compression efficiency. The FME results shouldn't be compared with the related works since the results of Table 10 already shows that the presented FME is not practical for real applications. Thus, in future works, a strategy to reduce the complexity of the FME will be explored.

Table 11 – Comparative results for ME encoding tool when processing only the Texture channel

Related Works		**PERLEBERG (2018)**	**XU (2018)**	**PASTUSZAK (2016)**	**Developed Hardware**
Video Coding Standard		HEVC	HEVC	HEVC	3D-HEVC
BMA Algorithm		Modified TZS	Sub-sampled Full Search	Modified TZS	Master-Slave TZS
Search Range		-64+64	H: -80+80 V: -48+48	-64+64	-64+64
SRAM size		no	75.5kB	76kB	44.1kB
ASIC Technology		45nm Nangate	28nm TSMC	90nm TSMC	40nm TSMC
Supported PU Sizes		Squared	32x32, 16x16 32x16, 16x32	All	All
Area (Gates)		18,103.0k	1,094.0k	422.6k	128,766.0k
2160p 30fps	Frequency (MHz)	144.92	350	400	634.98
	Power (mW)	140.84	47	293	12,874.83

The work XU et al. (2018) presents a ME architecture that adopts a sub-sampled FS algorithm in IME with a reduced search range of ±80 samples for Horizontal and ±48 samples for vertical positions. Moreover, the interpolation filters adopted in the FME evaluation of XU et al. (2018) was also simplified to reduce the hardware complexity. It supports the processing of only four PU sizes. Due to all these complexity reduction strategies, it reaches an architecture that requires a small area and small power dissipation than the developed hardware. Although, all these strategies can result in a high impact on encoding efficiency.

In the previous work (PERLEBERG et al., 2018), focused on HEVC, a hardware architecture adopting the TZS algorithm with several constraints was proposed. However, in PERLEBERG et al. (2018) no data reuse strategy was adopted, thus only the four squared PU sizes were processed to reduce complexity. The several PUs of a CTU were processed in part sequentially, thus it has fewer units when compared with processing all PUs in parallel. Therefore, as expected, the hardware of PERLEBERG et al. (2018) requires fewer area resources while it dissipates less power than the developed hardware.

The algorithm adopted by PASTUSZAK; TROCHIMIUK (2016) applies the TZS algorithm to $8x8$ PUs, and higher PU sizes are predicted based on the results of those $8x8$ PUs. Its FME Unit was also designed to process four $8x8$ PUs in parallel. Therefore, due to its processing is being performed from $8x8$ PUs, its architecture has reached a small area and power dissipation.

Table 12 presents the results of the related work that proposes hardware solutions for 3D-HEVC, along with the results of the developed hardware. The work AFONSO et al. (2019) proposes a hardware architecture for both ME and DE encoding tools. The hardware architecture proposed in our work can also be used to perform the DE tool only by selecting as reference frame a frame from a neighboring view. Although, the evaluations presented in Section 3 were only measured in the ME tool, being its impact in the encoding efficiency not evaluated in the DE context.

As in the developed hardware, AFONSO et al. (2019) performed several constraints in the TZS algorithm to reduce the complexity of those tools. It processes the two encoding tools in parallel, thus it requires a larger SRAM than the required by the developed hardware. Since it processes only two PU sizes, and due to all constraints applied in its algorithm, it reached a BD-rate impact of 23.3%. This impact is higher than the developed hardware disregarding the FME tool, which has reached 17.7% of BD-rate in the same conditions.

By disregarding the FME tool the developed hardware results in an area resource 99% smaller and a power dissipation 94.8% smaller than the required by the ME of AFONSO et al. (2019). These expressive results are obtained mainly since the hardware of AFONSO et al. (2019) was developed to evaluates the 2 channels of all 3 views in parallel, thus having several similar instances operating in parallel, and also since its results considers the implementation of the SRAM memory. Although, when considering both IME and FME Units, huge area resources are required to process

Table 12 – Results and related works for encoding tools of 3D-HEVC

Related Works		**Afonso (2009)**	**Developed Hardware** IME Only	**Developed Hardware** IME & FME
Encoding Tool		ME & DE	ME	
BMA Algorithm		Modified TZS	Master-Slave TZS	
SRAM size		ME: 165.87kB DE: 38.02kB	44.1kB	
ASIC Technology		45nm Nangate	40nm TSMC	
Supported PU Sizes		16x16, 32x32	All	
BD-rate impact		FCO: 23.2%	NCO: 17.9% FCO: 17.7%	NCO: 8.9% FCO: 8.8%
Area (Gates)		ME: 27,147k DE: 13,740k	269k	128,766.0k
1080p 30fps	Frequency (MHz)	100	634.98	634.98
	Power (W)	ME: 4.800 DE: 1.648	0.247	12.873

the fractional candidates, which also results in an increase in power dissipation when compared with the ME of AFONSO et al. (2019) with a smaller BD-rate impact.

As mentioned, the results of Table 10 shows that the presented FME is not practical for real applications, since it was implemented to evaluate all fractional blocks considering all PU sizes supported on HEVC, thus reaching the best encoding efficiency of ME. There are different strategies that could be adopted in this FME Unit to reduce its high resources requirement and also its power dissipation, as 1) Reduce the amount of PU sizes supported for fractional blocks, since traditionally some specific PU sizes are more common than others (AFONSO et al., 2016a). 2) Evaluating only fractional blocks at the half position to specific block sizes, thus reducing the size of each Evaluation Unit. 3) Introduce an internal buffer to store the fractional samples, thus enabling that one Evaluation Unit can process more than one IME vector. Therefore, those strategies will be better explored in future work.

6 CONCLUSION

This work presented an energy-efficient hardware design for the Integer Motion Estimation encoding tool adopting reuse strategies for data and operations, which allows the processing of all PU sizes supported on 3D-HEVC with a reduced memory communication. The developed hardware adopts the TZS algorithm, the default BMA algorithm used in 3D-HEVC Reference Software, with some hardware constraints to reduce its high complexity. The TZS was applied over the $64x64$ PUs, and all smaller PUs evaluate the same candidates evaluated by the $64x64$ PUs. Therefore the developed hardware explores operations reuse strategies to evaluate all possible partitions of each candidate in parallel, by computing the similarity value to small blocks and accumulating these similarity values to compose the similarity values of higher PUS. A Fractional Motion Estimation unit was also presented, with adopts some data reuse strategies to reduce the memory communication to process all PU sizes. The FME performs the interpolation of each region of the Search Area one time, and all candidate blocks that depend on the fractional sample interpolated are processed in parallel. The architecture was described in VHDL and synthesized for ASIC using the 40nm TSMC standard-cells library. The synthesis results show that the developed IME hardware can reach the processing of up to 5 views of UHD 2160p videos at 60 frames per second while dissipating 312.81mW and requiring 280k gates. The IME architecture requires the lowest gate count among the IME related works that process all PU sizes supported on HEVC. The developed IME also results in a hardware 99% smaller with a power dissipation 94.8% smaller than the unique work proposing hardware implementation for the 3D-HEVC Motion Estimation. The presented FME unit was not practical for real applications, since it adopts high parallelism to reach its best encoding efficiency, thus resulting in requiring a huge amount of hardware resources with high power dissipation. Therefore, in future works, strategies of reducing the FME complexity will be explored, mainly adopting some strategies on FME of specific PU sizes, but also enabling that one Evaluation Unit can process more than one IME vector, thus better using the hardware resources.

REFERENCES

AFONSO, V. **High-Throughput Dedicated Hardware Design Targeting the 3D-HEVC-Prediction Coding Tools**. 2019. 204p. Tese de Doutorado — Universidade Federal do Rio Grande do Sul, Porto Alegre.

AFONSO, V.; CONCEIÇÃO, R.; SALDANHA, M.; BRAATZ, L.; PERLEBERG, M.; CORRÊA, G.; PORTO, M.; AGOSTINI, L.; ZATT, B.; SUSIN, A. Energy-Aware Motion and Disparity Estimation System for 3D-HEVC With Run-Time Adaptive Memory Hierarchy. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.29, n.6, p.1878–1892, June 2019.

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. **Journal of Integrated Circuits and Systems**, [S.l.], v.11, n.2, p.106–120, 2016.

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. **Journal of Integrated Circuits and Systems**, [S.l.], v.11, n.2, p.106–120, 2016.

AFONSO, V.; SUSIN, A.; PERLEBERG, M.; CONCEIÇÃO, R.; CORRÊA, G.; AGOSTINI, L.; ZATT, B.; PORTO, M. Hardware-Friendly Unidirectional Disparity-Search Algorithm for 3D-HEVC. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018.

AGOSTINI, L. V. **Desenvolvimento de Arquiteturas de Alto Desempenho dedicadas à compressão de vídeo segundo o Padrão H.264/AVC**. 2007. 173p. Tese de Doutorado — Universidade Federal do Rio Grande do Sul, Porto Alegre.

AMISH, F.; BOURENNANE, E.-B. An efficient hardware solution for 3D-HEVC intra-prediction. **Journal of Real-Time Image Processing**, [S.l.], v.16, p.1559–1571, 2019.

BJONTEGAARD, G. Improvements of the BD-PSNR model. In: VCEG MEETING, 35., 2008, Berlin. **Anais...** ITU-T SG16, 2008.

Ching-Yeh Chen; Chao-Tsung Huang; Yi-Hau Chen; Liang-Gee Chen. Level C+ data reuse scheme for motion estimation with corresponding coding orders. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.16, n.4, p.553–558, April 2006.

FAN, Y.; HUANG, L.; HAO, B.; ZENG, X. A Hardware-Oriented IME Algorithm for HEVC and Its Hardware Implementation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.28, n.8, p.2048–2057, Aug 2018.

FEHN, C. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In: STEREOSCOPIC DISPLAYS AND VIRTUAL REALITY SYSTEMS XI, 2004. **Anais. . .** SPIE, 2004.

Goncalves, P.; Porto, M.; Zatt, B.; Agostini, L.; Correa, G. Octagonal-Axis Raster Pattern for Improved Test Zone Search Motion Estimation. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018. p.1763–1767.

GOPALAKRISHNA, S.; HANNUKSELA, M. M.; GABBOUJ, M. Flexible Coding Order for 3D video extension of H.265/HEVC. In: PICTURE CODING SYMPOSIUM (PCS), 2013., 2013. **Anais. . .** [S.l.: s.n.], 2013. p.253–256.

GU, C.; HUANG, L.; ZENG, X.; FAN, Y. A Micro-Code-Based Hardware Architecture of Integer Motion Estimation for HEVC. In: IFIP/IEEE 27TH INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION (VLSI-SOC), 2019., 2019. **Anais. . .** [S.l.: s.n.], 2019. p.269–274.

HEVC. **3D High Efficiency Video Coding**. Disponível em: <https://hevc.hhi.fraunhofer.de/3dhevc>. Acesso em: 2020-02-24.

ITU-T. **H.265 High efficiency video coding**. [S.l.]: Recommendation ITU-T, 2013.

JCT-3V. **3D-HEVC Reference Software**. Disponível em: <https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/>. Acesso em: 2020-02-24.

JCT-VC. **HEVC Reference Software**. Disponível em: <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>. Acesso em: 2020-02-24.

JCT-VC. **High Efficiency Video Coding**. Disponível em: <https://hevc.hhi.fraunhofer.de>. Acesso em: 2020-02-24.

KAUFF, P.; ATZPADIN, N.; FEHN, C.; MüLLER, M.; SCHREER, O.; SMOLIC, A.; TANGER, R. Depth Map Creation and Image-Based Rendering for Advanced 3DTV Services Providing Interoperability and Scalability. **Image Commun.**, USA, v.22, n.2, p.217–234, Feb. 2007.

LI, X.; WANG, R.; WANG, W.; WANG, Z.; DONG, S. Fast motion estimation methods for HEVC. In: IEEE INTERNATIONAL SYMPOSIUM ON BROADBAND MULTIMEDIA SYSTEMS AND BROADCASTING, 2014., 2014. **Anais. . .** [S.l.: s.n.], 2014. p.1–4.

LIAO, T.-T.; SHEN, C.-A.; TSENG, Y.-H. The algorithm and VLSI architecture of a high efficient motion estimation with adaptive search range for HEVC systems. **Journal of Real-Time Image Processing**, [S.l.], Jun 2017.

LIAO, T.-T.; SHEN, C.-A.; TSENG, Y.-H. The algorithm and VLSI architecture of a high efficient motion estimation with adaptive search range for HEVC systems. **Journal of Real-Time Image Processing**, [S.l.], v.16, p.1943–1958, June 2019.

LIN, J.; CHEN, Y.; HUANG, Y.; LEI, S. Motion Vector Coding in the HEVC Standard. **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v.7, n.6, p.957–968, Dec 2013.

MEDHAT, A.; SHALABY, A.; SAYED, M. S. High-throughput hardware implementation for motion estimation in HEVC encoder. In: IEEE 58TH INTERNATIONAL MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS (MWSCAS), 2015., 2015. **Anais. . .** [S.l.: s.n.], 2015.

MIANO, J. **Compressed Image File Formats**: JPEG, PNG, GIF, XBM, BMP. [S.l.]: Addison-Wesley Professional, 1999.

MÜLLER, K.; VETRO, A. **Common Test Conditions of 3DV Core Experiments**. San Jose: Standard JCT3V-G1100, 2014.

PASTUSZAK, G.; TROCHIMIUK, M. Algorithm and architecture design of the motion estimation for the H. 265/HEVC 4K-UHD encoder. **Journal of Real-Time Image Processing**, [S.l.], v.12, n.2, p.517–529, Ago 2016.

PERLEBERG, M.; AFONSO, V.; CONCEIÇÃO, R.; SUSIN, A.; AGOSTINI, L.; ZATT, B.; PORTO, M. A Power-Efficient and High-Throughput Hardware Design for 3D-HEVC Disparity Estimation. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018.

PERLEBERG, M.; AFONSO, V.; CONCEIÇÃO, R.; SUSIN, A.; AGOSTINI, L.; ZATT, B.; PORTO, M. High-Throughput Hardware Design for 3D-HEVC Disparity Estimation. **IEEE Design Test**, [S.l.], 2019.

PERLEBERG, M. R.; AFONSO, V.; CONCEICÃO, R.; SUSIN, A.; AGOSTINI, L.; PORTO, M.; ZATT, B. Energy and Rate-Aware Design for HEVC Motion Estimation Based on Pareto Efficiency. **Journal of Integrated Circuits and Systems**, [S.l.], v.13, n.1, Aug. 2018.

PERLEBERG, M. R.; GOEBEL, J. W.; MELO, M. S.; AFONSO, V.; AGOSTINI, L. V.; ZATT, B.; PORTO, M. ASIC power-estimation accuracy evaluation: A case study using video-coding architectures. In: IEEE 9TH LATIN AMERICAN SYMPOSIUM ON CIRCUITS SYSTEMS (LASCAS), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018.

PORTO, M. S. **Desenvolvimento algorítmico e arquitetural para a estimação de movimento na compressão de vídeo de alta definição**. 2012. 166p. Tese de Doutorado — Universidade Federal do Rio Grande do Sul, Porto Alegre.

SANCHEZ, G.; FERNANDES, R.; CATALDO, R.; AGOSTINI, L.; MARCON, C. Low Area Reconfigurable Architecture for 3D-HEVC DMMs Decoder Targeting 1080p Videos. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2018., 2018. **Anais. . .** [S.l.: s.n.], 2018. p.201–204.

SANCHEZ, G.; MARCON, C.; AGOSTINI, L. Real-time scalable hardware architecture for 3D-HEVC bipartition modes. **Journal of Real-Time Image Processing**, [S.l.], v.13, p.71–83, 2017.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. J. (Ed.). **High Efficiency Video Coding (HEVC).** [S.l.]: Springer International Publishing, 2014.

TECH, G.; CHEN, Y.; MÜLLER, K.; OHM, J.; VETRO, A.; WANG, Y. Overview of the Multiview and 3D Extensions of High Efficiency Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.26, n.1, p.35–49, Jan 2016.

TOSHIBA. **ZL2* Digital Series**. Disponível em: <http://www.toshiba-om.net/pdf/manuals/lcdtv/English/Country_Specific/ZL2-55-English-Specific.pdf>. Acesso em: 2020-03-15.

WIEGAND, T.; SULLIVAN, G. J.; BJONTEGAARD, G.; LUTHRA, A. Overview of the H.264/AVC video coding standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.13, n.7, p.560–576, July 2003.

XU, K.; HUANG, B.; LIU, X.; TU, X.; WU, Z.; YAN, Z.; LIU, P.; HAN, B.; LI, Y. A Low-power Pyramid Motion Estimation Engine for 4K@30fps Realtime HEVC Video Encoding. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.1–4.

ÜCKER, M.; AFONSO, V.; AUDIBERT, L.; SUSIN, A.; ZATT, B.; PORTO, M.; AGOSTINI, L. Low-Power and High-Throughput Architecture for 3D-HEVC Depth Modeling Mode 4. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2018., 2018. **Anais...** [S.l.: s.n.], 2018.

APPENDIX A – LIST OF PUBLICATIONS DURING THIS MASTERS DEGREE

During this masters degree, one work was accepted for publication on a conference, and two works were accepted for publication in relevant periodic:

- **Title:** A Power-Efficient and High-Throughput Hardware Design for 3D-HEVC Disparity Estimation.
  Conference: 2018 Symposium on Integrated Circuits and Systems Design (SBCCI)
  Authors: Murilo Perleberg, Vladimir Afonso, Ruhan Conceição, Altamiro Susin, Luciano Agostini, Bruno Zatt, and Marcelo Porto
  Qualis: B2

- **Title:** High-Throughput Hardware Design for 3D-HEVC Disparity Estimation.
  Periodic: 2019 IEEE Design & Test
  Authors: Murilo Perleberg, Vladimir Afonso, Ruhan Conceição, Altamiro Susin, Luciano Agostini, Bruno Zatt, and Marcelo Porto
  Qualis: A1

- **Title:** 6WR: a Hardware Friendly 3D-HEVC DMM-1 Algorithm and its Energy-Aware and High-Throughput Design.
  Periodic: 2020 IEEE Transactions on Circuits and Systems II: Express Briefs
  Authors: Murilo Perleberg, Vinicius Borges, Vladimir Afonso, Daniel Palomino, Luciano Agostini, and Marcelo Porto
  Qualis: A2