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

Trabalho Individual II

DESENVOLVIMENTO DE ARQUITETURAS PARA ESTIMAÇÃO DE MOVIMENTO FRACIONÁRIA SEGUNDO O PADRÃO HEVC

Vladimir Afonso

Pelotas, 2012

VLADIMIR AFONSO

DESENVOLVIMENTO DE ARQUITETURAS PARA ESTIMAÇÃO DE MOVIMENTO FRACIONÁRIA SEGUNDO O PADRÃO HEVC

Trabalho Individual apresentado ao Curso de Mestrado em Ciência da Computação da Universidade Federal de Pelotas.

Orientador: Prof. Dr. Denis T. Franco
Co-Orientador: Prof. Dr. Luciano V. Agostini

PELOTAS
2012

RESUMO

AFONSO, Vladimir. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC**. 2012. 57 f. Trabalho Individual (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

O processamento em tempo real de vídeos digitais de alta resolução está associado a uma elevada complexidade computacional, principalmente devido à necessidade do uso de técnicas de compressão de dados. Dessa forma, o desenvolvimento de circuitos integrados específicos para processamento de vídeo é uma atividade importante na área de pesquisa de sistemas digitais, uma vez que soluções em software geralmente não são satisfatórias. Os sistemas codificadores de vídeo apresentam diversas etapas distintas, como transformadas, quantização, codificação de entropia e estimação de movimento (ME – *Motion Estimation*), entre outras. A etapa de ME é a que mais contribui para a redução das informações a serem transmitidas, sendo que a mesma ainda pode utilizar uma técnica de refinamento chamada estimação de movimento fracionária (FME – *Fractional Motion Estimation*), a qual contribui para melhorar os resultados obtidos. A FME pode ser dividida em duas partes: uma responsável pela interpolação e outra pela busca e comparação dos blocos de vídeo. Inúmeros artigos científicos podem ser encontrados na literatura propondo arquiteturas para a FME do padrão de codificação de vídeo mais eficiente disponível, o padrão H.264/AVC (*Advanced Video Coding*). Porém, não foram encontrados até o momento trabalhos relacionados com a FME do padrão *High Efficiency Video Coding* (HEVC), que ainda está em desenvolvimento e será o sucessor do padrão H.264/AVC. Portanto, se faz necessário o desenvolvimento de arquiteturas eficientes para a etapa de FME do padrão HEVC. Este trabalho apresenta o estudo e o desenvolvimento de sistemas de hardware para a implementação da etapa de interpolação da FME segundo o padrão de codificação de vídeo HEVC. Além da etapa de interpolação, será desenvolvida em trabalhos futuros, a etapa de busca no bloco fracionário, obtendo-se após a integração destas etapas, uma arquitetura completa de FME para o padrão HEVC.

Palavras-chave: Vídeo Digital, Padrão HEVC, Estimação de Movimento, ME, FME.

ABSTRACT

AFONSO, Vladimir. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC**. 2012. 57 f. Trabalho Individual (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

The real time processing of high-resolution digital videos is associated with a high computational complexity, mainly due to the necessity of using data compression techniques. Because of this, the development of specific integrated circuits for video processing is an important area of research in digital systems, since software solutions are not satisfactory. The video coding systems have several distinct stages, as transforms, quantization, entropy coding and motion estimation (ME), among others. The step of ME is the most important for the reduction of information to be transmitted, and can use a refinement technique called fractional motion estimation (FME), which helps to improve the results. The FME can be divided in two parts: one that is responsible for interpolation and the other that is responsible by the search and comparison of video blocks. Many articles can be found in the scientific literature proposing FME architectures for the video encoding standard that is the most effective available, the H.264/AVC (Advanced Video Coding) standard. However, there are no related works with FME for the High Efficiency Video Coding (HEVC) standard, which is under development and will be the successor to the H.264/AVC standard. Therefore, the development of efficient architectures for FME according to the HEVC standard is necessary. This work presents the study and development of hardware implementations developed for the interpolation stage necessary in the FME step defined in the HEVC standard. In addition to the interpolation stage, the search stage in the fractional block will also be developed in future works. After integrating these stages, a complete architecture for FME will be available, according to HEVC standard.

Keywords: Digital Video, HEVC Standard, Motion Estimation, ME, FME.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALF	*Adaptive Loop Filter*
ALUT	*Adaptive Look-Up Table*
AVC	*Advanced Video Coding*
CABAC	*Context-Based Adaptive Binary Arithmetic Coding*
CAVLC	*Context-Based Adaptive Variable Length Coding*
Cb	*Chrominance Blue*
Cr	*Chrominance Red*
CU	*Coding Unit*
DCT	*Discrete Cosine Transform*
DVD	*Digital Versatile Disk*
FIR	*Finite Impulse Response*
FME	*Fractional Motion Estimation*
FPGA	*Field Programmable Gate Array*
HD	*High definition*
HDTV	*High Definition Digital Television*
HEVC	*High Efficiency Video Coding*
HM	*High Efficiency Video Coding Test Model*
HM1	*High Efficiency Video Coding Test Model 1 Encoder Description*
HM6	*High Efficiency Video Coding Test Model 6 Encoder Description*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
ITU-T	*International Telecommunication Union – Telecommunication Standardization Sector*
JCT-VC	*Joint Collaborative Team on Video Coding*
JPEG	*Joint Photographic Experts Group*
JPEG-LS	*Lossless* JPEG
LCU	*Largest Coding Unit*
MC	*Motion Compensation*
ME	*Motion Estimation*

MHz	Mega-Hertz, 10^6 hertz, unidade de frequência
MPEG	*Moving Picture Experts Group*
PIXEL	*Picture Element*, menor unidade finita de uma imagem bidimensional
PSNR	*Peak-to-Signal Noise Ratio*
PU	*Prediction Unit*
Q	*Quantization*
QFHD	*Quad Full* HD
qps	quadros por segundo
RGB	*Red, Green, Blue*
SAD	*Sum of Absolute Differences*
T	*Transform*
TMuC	*Test Model under Consideration*
TU	*Transform Unit*
TV	Televisão
UFPel	Universidade Federal de Pelotas
UHDTV	*Ultra High Definition Television*
VCEG	*Video Coding Experts Group*
VHDL	VHSIC *Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
Y	*Luminance*
YCbCr	*Luminance, Chrominance blue, Chrominance red*

SUMÁRIO

1 INTRODUÇÃO

Existem atualmente diversas aplicações envolvendo vídeo digital, como aparelhos de DVD e Blu-Ray, TV digital, compartilhamento de vídeos online, vídeo-telefonia, entre outras. A elevada complexidade computacional associada ao processamento em tempo real de vídeos digitais de alta definição, pode se beneficiar muito com o desenvolvimento de circuitos integrados específicos, já que de outra forma, com soluções em software, seriam necessários processadores comerciais de alto desempenho. A utilização deste tipo de processadores resultaria em um aumento de consumo energético e inviabilizaria a execução de vídeos digitais em alguns dispositivos portáteis.

O algoritmo de codificação, responsável pela compressão das sequências de vídeo originais (não processadas), apresenta diversas etapas, cada uma das quais com o objetivo de explorar algum tipo de redundância nos dados existentes nessas sequências. Desta forma, é possível reduzir a quantidade de dados a ser armazenada e transferida, com mínimas perdas na qualidade da imagem ou até sem perda alguma.

Dentre os padrões de compressão disponíveis, o padrão H.264/AVC é o mais eficiente em termos de desempenho na compressão de dados, ao custo de uma maior complexidade computacional em relação aos padrões concorrentes (AGOSTINI, 2007). Desde a sua aprovação em 2003, o padrão H.264/AVC vem sendo investigado exaustivamente pelos pesquisadores. Nos últimos anos, com a evolução da capacidade computacional dos equipamentos eletrônicos, foram possíveis inúmeras contribuições para melhorar a eficiência do padrão. Contudo, a demanda por resoluções cada vez maiores, principalmente no que se refere a aplicações destinadas aos dispositivos com recursos computacionais ou energéticos

limitados, como é o caso dos dispositivos portáteis, tornou necessário o desenvolvimento de um novo padrão.

Com o objetivo principal de dobrar a taxa de compressão permitida pelo padrão H.264/AVC, vem sendo desenvolvido o padrão *High Efficiency Video Coding* (HEVC) (SULLIVAN; WIEGAND, 2009). O padrão HEVC está em desenvolvimento desde Janeiro de 2010 através da colaboração de especialistas, reunidos em um grupo intitulado *Joint Collaborative Team on Video Coding* (JCT-VC), para ser o sucessor do padrão H.264/AVC. Mesmo com o padrão HEVC ainda em desenvolvimento, já é possível verificar que novas características foram introduzidas a fim de aumentar as taxas de compressão (McCANN *et al.,* 2012) (BROSS *et al.,* 2012). Algumas destas características estão relacionadas à predição inter-quadros, mais especificamente à etapa de estimação de movimento (ME).

Considerando um codificador de vídeo genérico, a etapa de ME visa à identificação e redução, ou até à eliminação, das redundâncias do tipo temporal, que consistem na grande semelhança que quadros temporalmente próximos costumam apresentar em uma sequência de vídeo. Dessa forma, é possível transmitir somente as informações referentes à diferença entre estes quadros. Esta é a etapa que apresenta a maior complexidade computacional de todo o codificador, mas também é a etapa que permite a maior parte dos ganhos em termos de taxa de compressão (PURI, 2004).

Os movimentos existentes em uma cena normalmente não se limitam a posições inteiras de pixel, e por isso, a ME emprega a técnica de estimação de movimento fracionária (FME) com o objetivo de se obter maior eficiência na codificação, utilizando posições de sub-pixel, além das posições inteiras (RICHARDSON, 2003). Para tal, a etapa de FME é composta basicamente de duas partes, uma etapa de interpolação, na qual são gerados os valores das amostras em posições de sub-pixel, e uma etapa de busca e comparação, onde os valores gerados são comparados com o melhor resultado da ME em posições inteiras. O desenvolvimento de arquiteturas eficientes para FME é muito importante, uma vez que grande parte dos ganhos obtidos na compressão se deve a esta etapa.

1.1 Motivação

Diversas são as motivações para o desenvolvimento de arquiteturas de hardware para FME do padrão HEVC. Sem dúvida, o fato de que o padrão HEVC ainda se encontra em desenvolvimento, e por isso, existem poucos trabalhos relacionados na literatura científica, é extremamente motivador para os pesquisadores da área de codificação de vídeo. Considerando que não foram encontrados até o momento trabalhos relacionados à FME e que grande parte dos ganhos na compressão se deve a esta etapa, a motivação se torna ainda maior. Além disso, os dispositivos eletrônicos que permitem a execução de vídeos digitais são cada vez mais exigidos, pois sistematicamente estes equipamentos têm seus recursos ampliados, como por exemplo, a execução de vídeos digitais com resoluções crescentes em dispositivos portáteis, o que já é uma realidade. Por isso, se faz necessário o desenvolvimento de arquiteturas de hardware cada vez mais eficientes, seja em termos de velocidade, consumo energético ou quantidade de hardware utilizado.

1.2 Objetivos

Este trabalho tem como objetivos principais o estudo e desenvolvimento de arquiteturas de hardware de alto desempenho para a FME do padrão de codificação de vídeo HEVC. A realização de um estudo detalhado das especificações do algoritmo da FME para o padrão HEVC, conforme definido nos *drafts* mais recentes do padrão, e um estudo sobre o estado da arte em termos de arquiteturas para FME do padrão HEVC, possibilitarão o embasamento teórico necessário para o desenvolvimento de arquiteturas de hardware eficientes. A arquitetura desenvolvida para a etapa de interpolação das amostras de luminância será integrada futuramente a uma etapa de busca e comparação de blocos fracionários, obtendo-se dessa forma uma arquitetura completa de FME para o padrão HEVC.

1.3 Organização do Trabalho

Este trabalho está estruturado em seis capítulos, de forma que nos capítulos iniciais são abordados alguns conceitos imprescindíveis, enquanto que nos capítulos

finais são apresentadas as arquiteturas de hardware desenvolvidas, os esquemas propostos e resultados obtidos, assim como as conclusões deste trabalho.

O segundo capítulo apresenta alguns conceitos importantes sobre compressão de vídeos digitais, como compressão com perdas e sem perdas, espaço de cores e subamostragem de cores, redundância na representação de vídeos digitais, codificadores de vídeo, além de outros conceitos básicos relacionados ao assunto.

Em seguida, no terceiro capítulo, são apresentados alguns conceitos referentes ao padrão de codificação de vídeo HEVC, em especial os blocos ME e FME. Além disso, é apresentado o estado da arte em termos de arquiteturas para FME e são realizadas algumas comparações entre os padrões de codificação de vídeo atuais.

No quarto capítulo são apresentados os softwares utilizados para o desenvolvimento das arquiteturas de hardware, bem como a metodologia empregada no desenvolvimento e os diagramas das arquiteturas.

A seguir, no quinto capítulo, são descritos os resultados obtidos com as arquiteturas desenvolvidas.

Enfim, no sexto e último capítulo são apresentadas as conclusões deste trabalho, bem como as oportunidades de estudos em trabalhos futuros.

2 CONCEITOS DE COMPRESSÃO DE VÍDEOS DIGITAIS

Este capítulo tem por objetivo apresentar alguns conceitos importantes sobre compressão de dados em vídeos digitais, como compressão com perdas e sem perdas, espaço de cores e subamostragem de cores, redundância de informações na representação de vídeos digitais, codificadores de vídeo, além de outros conceitos fundamentais relacionados ao assunto. Os conceitos apresentados neste capítulo estão estreitamente relacionados com as arquiteturas de hardware desenvolvidas neste trabalho que serão apresentadas nos próximos capítulos.

2.1 Conceitos Fundamentais de Vídeos Digitais

Um vídeo digital consiste em uma sequência de imagens independentes, captadas com um determinado intervalo de tempo entre as mesmas. Para que seja obtida uma sensação de movimento contínuo, ou seja, de tempo real, a taxa de captura das imagens deve ser de 24 a 30 imagens a cada segundo (GONZALEZ, 2003). Estas imagens independentes, que formam uma sequência de vídeo, são chamadas de quadros (*frames)* e são compostas por pontos, mais comumente chamados de pixels (*picture element*). Os pixels são representados através de números, que correspondem às componentes de brilho ou cor das amostras, de acordo com o espaço de cor utilizado. Os quadros de uma sequência de vídeo podem ser divididos em blocos, de acordo com os padrões de codificação de vídeo, permitindo a aplicação eficiente das técnicas de compressão. Os blocos podem apresentar diferentes tamanhos. Inclusive, em um mesmo padrão de codificação, o tamanho do bloco pode ser variável. Na Figura 2.1 pode ser observada uma sequência de quadros temporalmente próximos e um destes quadros divididos em blocos.

A representação digital de uma cena envolve basicamente dois tipos de amostragem: espacial e temporal. A amostragem espacial consiste de uma matriz de pontos chamada de resolução do vídeo, que normalmente é retangular e tem o objetivo de formar os quadros da sequência de vídeo. Quanto maior for a resolução, melhor será a qualidade da imagem, uma vez que uma maior quantidade de pixels será usada na representação dos quadros. Já a amostragem temporal está relacionada aos intervalos de tempo entre a captura das imagens em uma sequência de vídeo. Quanto maior for a taxa de amostragem, maior será a percepção de movimento no vídeo. A amostragem de forma digital de uma cena de vídeo é o que caracteriza os vídeos digitais.

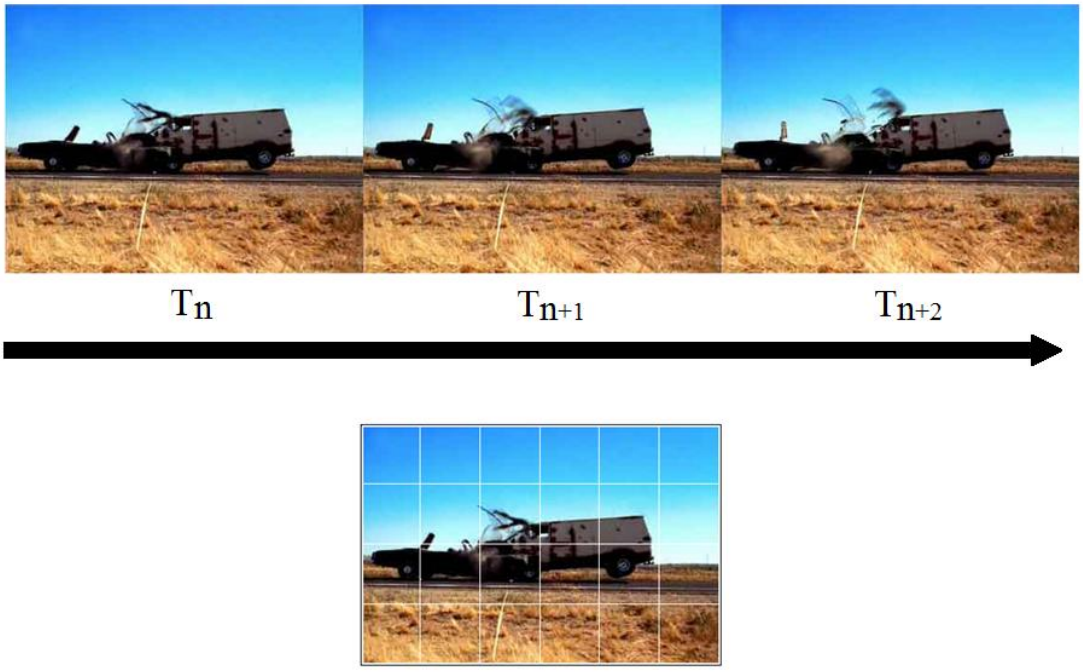

Figura 2.1 – Sequência de quadros e blocos de um vídeo digital.

2.2 Métodos de Compressão Com Perdas e Sem Perdas

O processamento de vídeos digitais envolve uma quantidade muito grande de dados, o que torna inviável a utilização deste tipo de vídeo em aplicações de tempo real, sem o uso de técnicas de compressão. Com a utilização da compressão nos vídeos digitais, através do bloco codificador, é possível reduzir a quantidade de dados a ser armazenada e transferida, com mínimas perdas na qualidade da imagem ou até sem perda alguma.

Considerando as perdas de informação que podem ocorrer no processo de codificação dos codificadores de vídeo, os métodos de compressão são classificados em dois tipos: com perdas (*lossy*) ou sem perdas (*lossless*) (RICHARDSON, 2002). Quando o método de compressão sem perdas é utilizado nos codificadores, as informações do arquivo de saída, após ser realizada a descompressão, são idênticas às informações originais que foram comprimidas, referentes ao arquivo de entrada. Isto é possível devido às técnicas de compressão adotadas e um exemplo de aplicação deste método é o compressor de arquivos ZIP (SALOMON, 2008), o qual reestrutura o arquivo atribuindo códigos menores para os símbolos com maior ocorrência e códigos maiores para símbolos de menor ocorrência. Apesar de compressores sem perdas reduzirem o tamanho dos arquivos, este método de compressão não funciona bem com imagens e vídeos digitais, pois os valores dos pixels, que correspondem aos símbolos, apresentam uma probabilidade de ocorrência muito parecida. Dessa forma, as taxas de compressão obtidas para imagem e vídeo digital utilizando compressão sem perdas são reduzidas. Por exemplo, o compressor sem perdas de imagem JPEG-LS obtém taxas de compressão de 3 a 4 vezes (RICHARDSON, 2003).

A compressão de imagem e vídeo digital requer taxas de compressão elevadas, uma vez que a quantidade de informações a serem processadas, armazenadas e transmitidas é muito grande. Nesse contexto, o método de compressão com perdas é o mais adequado e os compressores de vídeo visam explorar o fato de que algumas informações da imagem não são relevantes para o sistema visual humano, removendo-as (RICHARDSON, 2003). Com este tipo de técnica se consegue obter elevadas taxas de compressão mantendo-se a qualidade da imagem em níveis aceitáveis. Em alguns casos, estas alterações de qualidade sequer são perceptíveis. Utilizando-se técnicas de compressão de vídeo com perdas, o padrão H.264/AVC atinge taxas de compressão de dezenas de vezes, com uma qualidade visual muito próxima a do vídeo original (GHANBARI, 2003).

2.3 Espaço de Cores e Subamostragem de Cores

É chamado de espaço de cores um sistema utilizado para representar cores. A escolha do espaço de cores é muito importante para a eficiência na codificação de vídeos digitais e deve levar em consideração a maneira como o sistema visual

humano interpreta as cores. Entre os espaços de cores utilizados para representação de imagens e vídeos digitais encontram-se os sistemas RGB e YCbCr (SHI, 1999). O sistema RGB é um dos mais conhecidos, uma vez que é utilizado em monitores coloridos e aparelhos de TV. Este espaço de cores consiste de três matrizes, cada uma correspondente a uma das cores primárias captadas pelo sistema visual humano que são o vermelho, o verde e o azul. Através da combinação dessas três cores é possível a obtenção de todas as demais cores percebidas pelo sistema visual humano. O significado da sigla RGB é justamente o nome das cores que compõem o sistema em inglês, *red*, *green* e *blue*. No sistema RGB, cada um dos canais representa a intensidade de sua respectiva cor enquanto que a informação de brilho, ou seja, da intensidade luminosa, está distribuída entre os três canais.

Um outro espaço de cores, bastante utilizado na representação de vídeos digitais, é o sistema YCbCr. O sistema YCbCr também é composto por três componentes: luminância (Y), que define o brilho; crominância azul (Cb); e crominância vermelha (Cr) (MIANO, 1999). A representação de vídeos digitais com o sistema RGB não é adequada, pois os componentes R, G e B apresentam um elevado grau de correlação. Dessa forma, a compressão de vídeos utiliza espaço de cores do tipo luminância e crominância, como o YCbCr (RICHARDSON, 2002). Além disso, no sistema YCbCr a informação de cor é totalmente separada da informação de brilho. Assim, as informações de brilho podem ser tratadas separadamente das informações de cor nos codificadores. Como o sistema visual humano é mais sensível às informações de luminância do que de crominância (GONZALEZ, 2003), os codificadores de vídeo diminuem a taxa de amostragem das informações de crominância em relação às informações de luminância para aumentar a eficiência da codificação (RICHARDSON, 2002). Esta técnica é utilizada no espaço de cores YCbCr nos codificadores de vídeo e é chamada de subamostragem de cores.

A subamostragem de cores pode relacionar componentes de crominância e luminância de várias formas, sendo que os formatos mais comuns são conhecidos como 4:4:4, o 4:2:2 e o 4:2:0. O formato 4:4:4 considera que para cada quatro amostras de luminância, existem quatro amostras de crominância azul e quatro amostras de crominância vermelha. Como as três componentes apresentam a mesma resolução, nesse caso a subamostragem de cores não é empregada e não existe perda na qualidade da imagem. Já no formato 4:2:2, para cada quatro

amostras de Y, existem apenas duas amostras de Cb e duas amostras de Cr. Sendo assim, o tamanho total do vídeo é reduzido em 25% só pela subamostragem de cores, uma vez que metade das informações de crominância são descartadas. O formato 4:2:0 considera que para cada quatro amostras de Y, existe apenas uma amostra de Cb e uma amostra de Cr. Neste caso, apenas um quarto das amostras de crominância é utilizado e o tamanho total do vídeo é reduzido à metade só pela subamostragem de cores quando comparado com um vídeo RGB ou YCbCr no formato 4:4:4. A Figura 2.2 mostra exemplos de vários formatos de subamostragem de crominância.

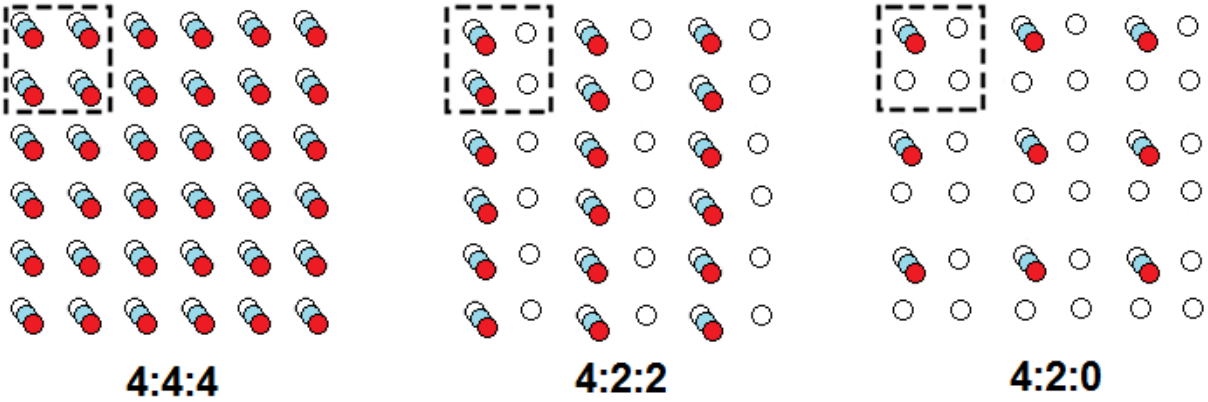

Figura 2.2 – Formatos de subamostragem de crominância.

2.4 Redundância de Informações na Representação de Vídeos Digitais

Em geral, vídeos digitais apresentam redundância nas suas informações, ou seja, alguns dados não apresentam novas informações relevantes para a representação do vídeo. Por isso, os codificadores de vídeo procuram reduzir essas redundâncias de forma a aumentar a eficiência da codificação de vídeo. Existem basicamente três tipos de redundância de dados que podem ser exploradas na compressão de vídeos, que são a redundância espacial, redundância temporal e redundância entrópica.

A redundância espacial também é conhecida como redundância intra-quadro (*intraframe*) (GHANBARI, 2003) e pode ser percebida tanto no domínio espacial como no domínio das frequências. No domínio espacial esse tipo de redundância está relacionado com a tendência que pixels vizinhos têm de apresentar valores semelhantes no mesmo quadro de uma sequência de vídeo e a operação utilizada

para explorar essa redundância nos codificadores de vídeo atuais é chamada de intra-quadro. Contudo, a redundância espacial também pode ser explorada no domínio das frequências através de uma operação chamada de quantização (Q – *Quantization*), cujo princípio de funcionamento está baseado na eliminação de partes das informações da imagem menos relevantes ao sistema visual humano, atenuando ou até eliminando determinadas frequências. Para que a quantização seja utilizada, primeiramente as informações devem ser transformadas do domínio espacial para o domínio das frequências, requerendo outra etapa na codificação, chamada de transformada (T – *Transform*). A quantização, ao contrário da operação intra-quadro, gera perdas no processo de codificação. Porém, as perdas geradas não causam problemas, interferindo de forma pouco significativa na qualidade da imagem.

A redundância temporal, também conhecida como redundância inter-quadros (*interframe*) (GHANBARI, 2003) corresponde à grande semelhança existente entre quadros temporalmente próximos de uma sequência de vídeo. Considerando dois quadros de vídeo temporalmente próximos, muitos blocos de pixels simplesmente não mudam de valores. Isto acontece, por exemplo, em blocos de pixels que compõem o fundo de uma cena em que um determinado objeto encontra-se em movimento. Da mesma forma, o próprio objeto em movimento mantém os valores de seus blocos de pixel, porém com um deslocamento em relação aos demais quadros. A exploração das redundâncias do tipo temporal permite elevadas taxas de compressão e são utilizadas em todos os padrões de codificação de vídeos atuais.

A redundância entrópica está relacionada com a probabilidade de ocorrência dos símbolos codificados. A entropia é uma medida da quantidade média de informação transmitida por símbolo do vídeo (SHI, 1999). Sendo assim, os codificadores que exploram a redundância entrópica visam à transmissão da maior quantidade de informação possível por símbolo codificado, ou seja, com um número menor de bits. Existem diferentes técnicas e algoritmos de compressão sem perdas utilizados com o objetivo de explorar a redundância entrópica.

2.5 Codificadores de Vídeo Digital

Na Figura 2.3 pode ser observado o diagrama em blocos de um modelo genérico de codificador de vídeo que está de acordo com a maioria dos padrões de compressão de vídeo atuais (AGOSTINI, 2007). As principais etapas representadas que compõe o codificador são a predição inter-quadros, a predição intra-quadro, transformada, quantização e codificação de entropia. Cada uma das etapas presentes no codificador tem o propósito de explorar algum tipo de redundância de informações existentes nos vídeos digitais.

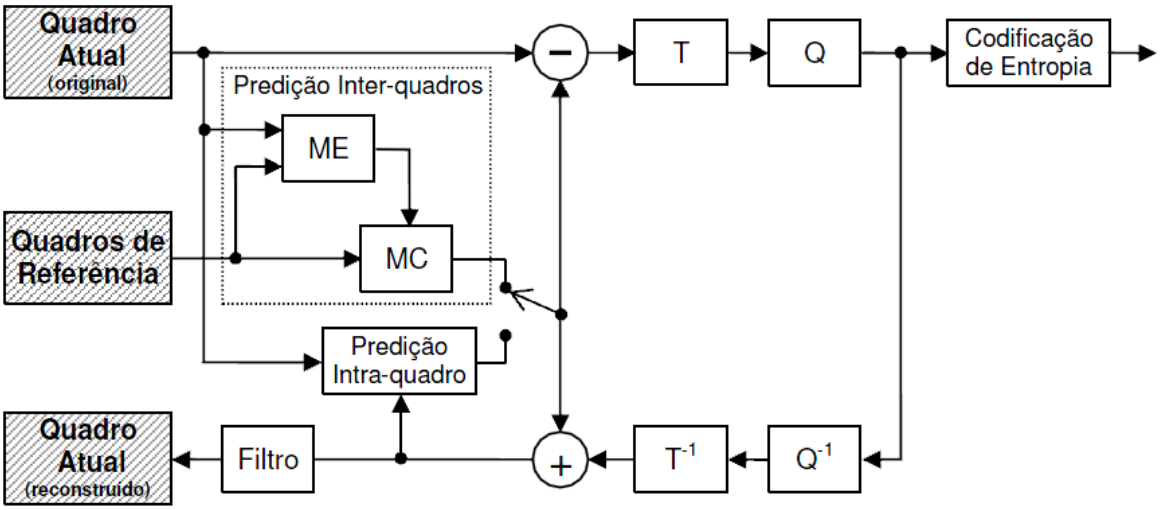

Figura 2.3 – Modelo genérico de codificador de vídeo.

Na predição inter-quadros estão localizadas a ME e a compensação de movimento (MC - *Motion Compensation*). A ME tem como objetivo identificar a redundância temporal para que esse tipo de redundância possa ser eliminado, ou pelo menos atenuado. Isto é feito através de uma comparação entre blocos de vídeo do quadro atual e blocos de quadros processados anteriormente, chamados quadros de referência. Quando o bloco de vídeo mais semelhante ao bloco atual é encontrado, um vetor de movimento é gerado indicando a localização desse bloco. Na etapa seguinte, a MC, são construídos os quadros estimados a partir dos vetores de movimento gerados pela ME.

A etapa de predição intra-quadro permite a redução de redundâncias espaciais no domínio espacial, que estão relacionadas às semelhanças entre os pixels vizinhos de um mesmo quadro. Ao contrário da ME, que necessita das

informações de quadros de referência, a predição intra-quadro precisa apenas das informações do quadro atual. Como pode ser observado na Figura 2.3, a chave seletora representa uma etapa de controle no codificador, a qual é responsável por escolher entre as duas predições, inter-quadros ou intra-quadro, a que será utilizada em cada bloco de vídeo, dependendo das características do vídeo a ser codificado.

Uma operação de subtração entre os valores do quadro atual e do quadro reconstruído obtido através das predições inter-quadros ou intra-quadro, permite a obtenção de uma diferença residual entre os quadros, chamada de resíduo. O resíduo resultante deve ser somado ao quadro reconstruído para que este quadro possa ser utilizado como referência para a codificação de um próximo quadro na predição inter-quadros. No caso da predição intra-quadro ser escolhida, a reconstrução deve ser feita por blocos, pois o bloco recém-predito poderá servir de referência para a predição do próximo bloco. Os valores de resíduo obtidos estão ligados diretamente à eficiência da predição. Enquanto valores baixos de resíduo indicam alta eficiência, valores altos de resíduo indicam baixa eficiência da predição.

Após a obtenção do resíduo e as predições inter-quadros ou intra-quadro, existe a etapa da transformada, cujo objetivo é transformar as informações do domínio espacial para o domínio das frequências. Em seguida, a etapa de quantização é aplicada no domínio das frequências nas amostras transformadas de forma a eliminar as frequências menos relevantes ao sistema visual humano. O processo de quantização gera perdas, uma vez que as informações são eliminadas de forma irreversível.

Finalmente, após a quantização, é realizada a codificação de entropia. A codificação de entropia consiste de uma técnica de compressão sem perdas que visa à alteração na representação dos símbolos com base na probabilidade de ocorrência, sendo que diversos tipos de algoritmos podem ser utilizados. O padrão HEVC, por exemplo, utiliza os algoritmos CAVLC *(Context-Based Adaptive Variable Length Coding*) e CABAC *(Context-Based Adaptive Binary Arithmetic Coding)*, sendo que a utilização de cada um deles depende das configurações do padrão (McCANN *et al.,* 2012).

Ainda se fazem necessárias no codificador, as etapas de transformada e quantização inversas para geração do quadro atual reconstruído. Este quadro é importante, pois serve como quadro de referência para a codificação inter-quadros ou intra-quadro, na codificação dos próximos blocos ou do próximo quadro, já que as

perdas geradas na quantização são irreversíveis. Para obtenção do quadro atual reconstruído ainda é necessária a soma do resíduo com a saída da transformada inversa, para que as referências, usadas na compressão e na descompressão do vídeo, sejam idênticas.

3 PADRÃO HEVC DE CODIFICAÇÃO DE VÍDEO

Neste capítulo é apresentada uma visão geral do padrão de codificação de vídeo HEVC. Algumas características importantes são introduzidas, como o funcionamento da ME, em especial da FME, foco deste trabalho. Também é apresentado o estado da arte em termos de arquiteturas para FME dos padrões de codificação de vídeo atuais e são realizadas algumas comparações entre estes padrões de codificação de vídeo.

3.1 Histórico

O padrão HEVC está em desenvolvimento desde Janeiro de 2010 através da colaboração de especialistas ITU-T, ISO e IEC, reunidos no grupo JCT-VC, para ser o sucessor do padrão H.264/AVC. O objetivo principal é dobrar a taxa de compressão permitida pelo padrão H.264/AVC mantendo a mesma qualidade subjetiva da imagem. Regularmente, ocorrem reuniões do JCT-VC em diferentes locais do mundo e, após estas reuniões, são disponibilizados os resultados obtidos pelo HEVC em termos de melhorias nas taxas de compressão, comparado ao padrão H.264/AVC (LI *et al.,* 2012).

A chamada à apresentação de propostas que iniciou o projeto do padrão HEVC foi emitida conjuntamente pelo ITU-T (VCEG) e ISO/IEC (MPEG) em Janeiro de 2010 (ITU-T/ISO/IEC, 2010), e os trabalhos de normalização subsequentes estão sendo conduzidos pelo JCT-VC. Em Abril do mesmo ano, no primeiro *meeting* realizado, o JCT-VC definiu a primeira versão do software de referência, chamada *Test Model under Consideration* (TMuC), documentada posteriormente (JCTVC-B204, 2010). O objetivo desta documentação era começar a preparação de um modelo de teste formal e a descrição do software de referência, incluindo métodos

de codificação de referência para serem compartilhados pelos pesquisadores. Estes métodos de referência permitem avaliações justas do impacto das novas tecnologias propostas durante o processo de normalização do HEVC. A partir do terceiro *meeting*, em Outubro de 2010, foram estabelecidos os documentos HEVC *Test Model* (HM), modelos de referência que permanecem até o presente momento. O primeiro foi intitulado HM1 e a cada novo *meeting*, uma nova versão é disponibilizada. Atualmente, no documento HM6 – *High Efficiency Video Coding Test Model 6 Encoder Description*, estão definidos dois perfis para o padrão HEVC, o perfil *Main* e o perfil *High Efficiency* 10. O foco de cada um dos perfis é diferente, enquanto um perfil utiliza ferramentas de codificação que possibilitam uma elevada eficiência na compressão em detrimento da complexidade computacional, no outro, o perfil *Main*, as ferramentas selecionadas tem como objetivo manter uma baixa complexidade, com um desempenho de compressão razoavelmente elevado. As principais diferenças entre os perfis estão no fato do perfil *High Efficiency* 10 utilizar um *bit depth* de 10 bits, capacidade completa no processo de *Loop Filtering*, incluindo ALF (*Adaptive Loop Filter*) e a utilização do algoritmo CABAC no codificador de entropia, enquanto que no perfil *Main*, o *bit depth* é de 8 bits, não existe suporte a ALF e a codificação de entropia utiliza CAVLC. As informações fornecidas na seção seguinte, sobre a estrutura geral de codificação do HEVC, são baseadas no perfil *Main*.

3.2 Estrutura Geral de Codificação

O padrão HEVC utiliza um esquema de codificação híbrido bem conhecido, baseado em blocos, que apresenta codificação com predições intra e compensação de movimento, transformadas e codificação de entropia. Em contraste com esquemas convencionais, o HEVC emprega um esquema de compressão de vídeo baseado no particionamento dos blocos codificados em uma hierarquia altamente flexível, que permite o uso de grandes e múltiplas dimensões para blocos de predição e transformadas, além de novas ferramentas de codificação. Estes novos aperfeiçoamentos presentes no HEVC melhoram significativamente sua eficiência na codificação.

Uma das principais inovações do padrão HEVC está no novo esquema de compressão de vídeo baseado em uma hierarquia flexível de representação unitária

que inclui três conceitos de bloco: Unidade de Codificação (CU – *Coding Unit*), Unidade de Predição (PU – *Prediction Unit*) e Unidade de Transformada e Quantização (TU – *Transform Unit*). A separação da estrutura do bloco em três diferentes conceitos permite a cada um ser otimizado de acordo com sua função.

Em padrões de codificação anteriores como MPEG-1 e MPEG-2 é utilizada compensação de movimento com tamanho de bloco fixo, com macrobloco 16 × 16. O Padrão H.264/AVC, por sua vez, utiliza um tamanho de macrobloco de 16 × 16 com a adição de uma *quadtree* de profundidade 2. Mesmo assim, o tamanho de bloco máximo de 16 × 16 é pequeno quando aplicado a vídeos de alta resolução e causa ineficiência. Para resolver este problema, o tamanho máximo permitido passou a ser 64 × 64. No padrão HEVC, a estrutura de compressão de vídeo foi projetada de forma que todos os blocos do codificador podem explorar o tamanho de bloco flexível.

A estrutura geral de codificação do Padrão HEVC permite o particionamento da imagem em LCUs (*Largest Coding Unit*). Uma LCU é composta por um bloco de amostras de luminância juntamente com outros dois blocos correspondentes às amostras de crominância. Os tamanhos dos blocos de crominância, em relação ao bloco de luminância, dependem de qual é a amostragem de cores adotada. O conceito de LCU é análogo ao conceito de macrobloco utilizado em padrões anteriores, como o padrão H.264/AVC. Considerando a versão atual do HEVC, o tamanho máximo permitido para uma LCU é 64x64 amostras de luminância, o que corresponde também ao maior tamanho de CU permitido.

Uma LCU é composta por uma ou mais CUs. As CUs são utilizadas para predição inter-quadros/intra-quadro, são sempre quadradas, com tamanho 2Nx2N, onde N apresenta valores de 4 a 32. Portanto, as CUs assumem tamanhos a partir de 8x8 amostras de luminância até o tamanho da LCU. O conceito de CU permite a divisão recursiva em quatro blocos de tamanho igual, partindo da LCU. Este processo com divisões recursivas permite tamanhos pequenos ou grandes de unidades e forma uma estrutura de codificação em forma de árvore quadrática composta por blocos de CU. As figuras 3.1 e 3.2 mostram uma LCU dividida em CUs e a árvore quadrática que corresponde a esta divisão, respectivamente.

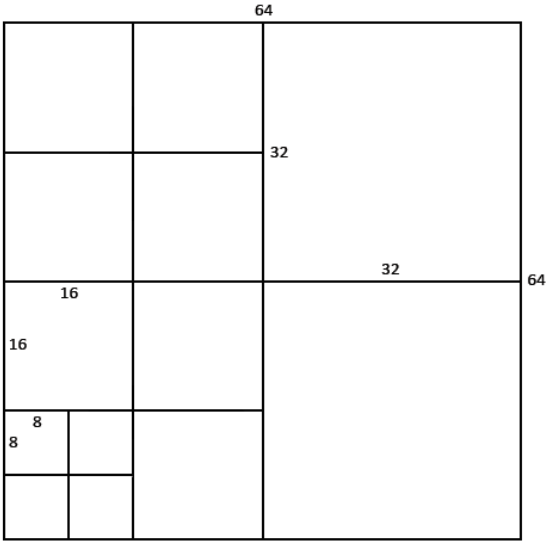

Figura 3.1 – Exemplo de uma LCU dividida em CUs.

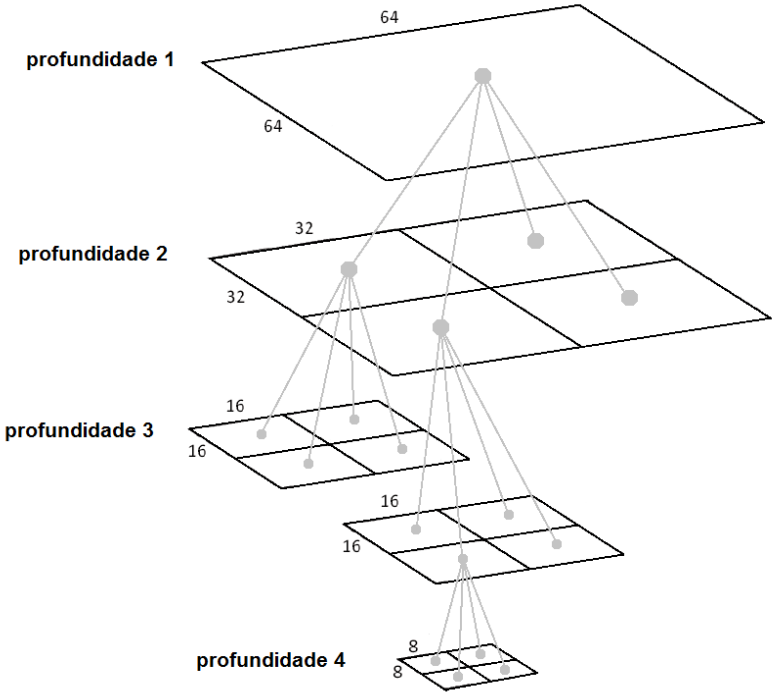

Figura 3.2 – Exemplo de árvore quadrática de uma LCU.

Na Figura 3.2 pode-se observar uma árvore quadrática de uma LCU. Primeiramente a LCU de dimensões 64x64 (profundidade 1) é dividida em quatro blocos 32x32 (profundidade 2). Destes blocos 32x32, dois são divididos em blocos 16x16 (profundidade 3). Os blocos 32x32 não divididos são codificados como CUs 32x32. Na profundidade 3 acontece o mesmo processo, os blocos 16x16 que não

são divididos são codificados como CUs 16x16. Um dos blocos 16x16 é dividido em blocos 8x8 (profundidade 4), chegando na menor dimensão possível para CU, sendo assim codificado.

Cada CU pode conter uma ou mais PUs. A PU é a unidade básica utilizada para os processos de predição, o que contempla a etapa de ME. A PU não se restringe às formas quadradas, ou seja, são permitidas também formas retangulares. Isto é feito com o objetivo de facilitar o particionamento que corresponde a limites de objetos reais na imagem. A Figura 3.3 mostra os diferentes tipos de divisão de uma CU em PUs, considerando o perfil *Main* (McCANN *et al.,* 2012, p. 8). O tipo de divisão NxN só é permitido quando o tamanho correspondente da CU é maior do que o menor tamanho de CU permitido. Desta forma, pode-se concluir que os menores tamanhos possíveis para PUs são 8x4 e 4x8, quando a CU é 8x8, e o maior tamanho é 64x64, o qual pode ocorrer apenas quando a CU é 64x64. Considerando o perfil *High Efficiency* 10 do HEVC, a PU pode assumir outros tipos de particionamento, os quais não serão abordados neste trabalho.

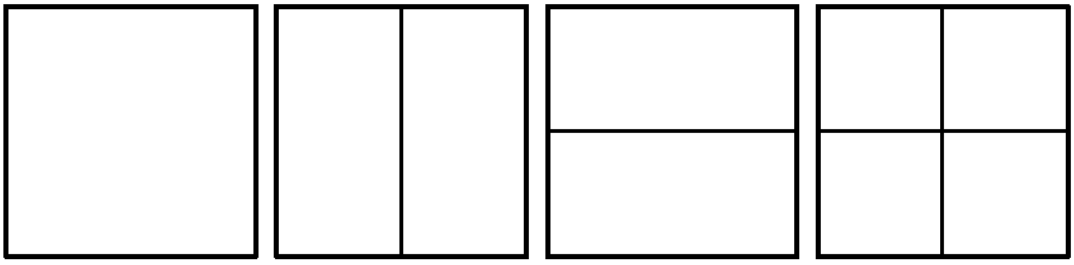

Figura 3.3 – Quatro tipos de divisão de uma CU em PUs.

A TU é a unidade básica para os processos de transformada e quantização e é sempre formada por quadrados, considerando o perfil *Main*. A TU pode apresentar tamanhos de 4x4 até 32x32 amostras de luminância. Cada CU pode conter uma ou mais TUs, onde várias TUs podem estar dispostas em uma estrutura de *quadtree*, como apresentado na Figura 3.4 (McCANN *et al.,* 2012, p. 9). Considerando o perfil *High Efficiency* 10 do HEVC, a TU pode assumir outros tipos de particionamento, inclusive não quadrados, sendo que a forma da TU depende do tipo de particionamento da PU.

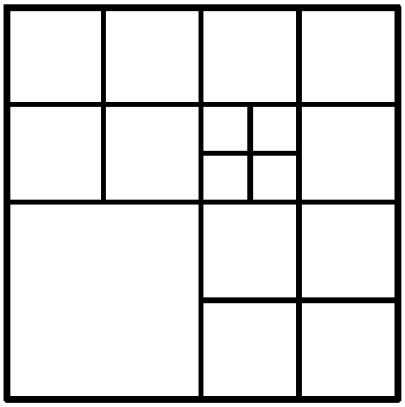

Figura 3.4 – Exemplo de CU 32x32 dividida em TUs.

3.3 Ferramentas de Codificação

No documento HM6 (2012) é possível verificar as diversas ferramentas de codificação disponíveis no padrão HEVC, como as predições intra-quadro e inter-quadros, transformadas e quantização, codificação de entropia e *loop filtering*. A ME faz parte da predição inter-quadros e será abordada nas seções seguintes.

3.3.1 Estimação de Movimento

Considerando o modelo genérico de codificador de vídeo apresentado no capítulo anterior, a ME é a etapa que visa à identificação e redução das redundâncias do tipo temporal, aquelas redundâncias relacionadas com a grande semelhança que quadros temporalmente próximos costumam apresentar em uma sequência de vídeo. Com a etapa de ME é possível transmitir somente as informações referentes à diferença entre estes quadros, chamada de resíduo. Quanto menor for o resíduo gerado na codificação, melhores serão os resultados de compressão.

A ME prediz o quadro atual utilizando quadros de referência, ou seja, quadros anteriormente codificados. O quadro atual é dividido em blocos e estes são comparados aos blocos do quadro de referência, considerando uma determinada área de busca (área de pesquisa). No caso de semelhança entre os blocos dos quadros atual e de referência, é gerado um vetor de movimento que irá permitir a localização do bloco que gerou o melhor casamento. Esta visão geral do processo

de ME pode ser observada na Figura 3.5 (PORTO, 2012, p.37). Na etapa seguinte, a MC, é construído o quadro estimado baseado nas informações dos vetores de movimento gerados pela ME.

Durante a codificação no padrão HEVC, são gerados os resultados do resíduo, baseados em algum critério de similaridade. O HEVC utiliza o SAD (*Sum of Absolute Differences*), um critério que costuma ser bastante utilizado em aplicações com desenvolvimento em hardware, pois pode ser implementado utilizando apenas operações de somas e deslocamentos, ao contrário de outros critérios de similaridade existentes, os quais necessitam de operações de divisão e exponenciação. Estes resultados de resíduo são gerados para todos os tamanhos de PU possíveis de acordo com o perfil do HEVC utilizado, mas apenas o particionamento que apresentar o melhor resultado é selecionado para a codificação.

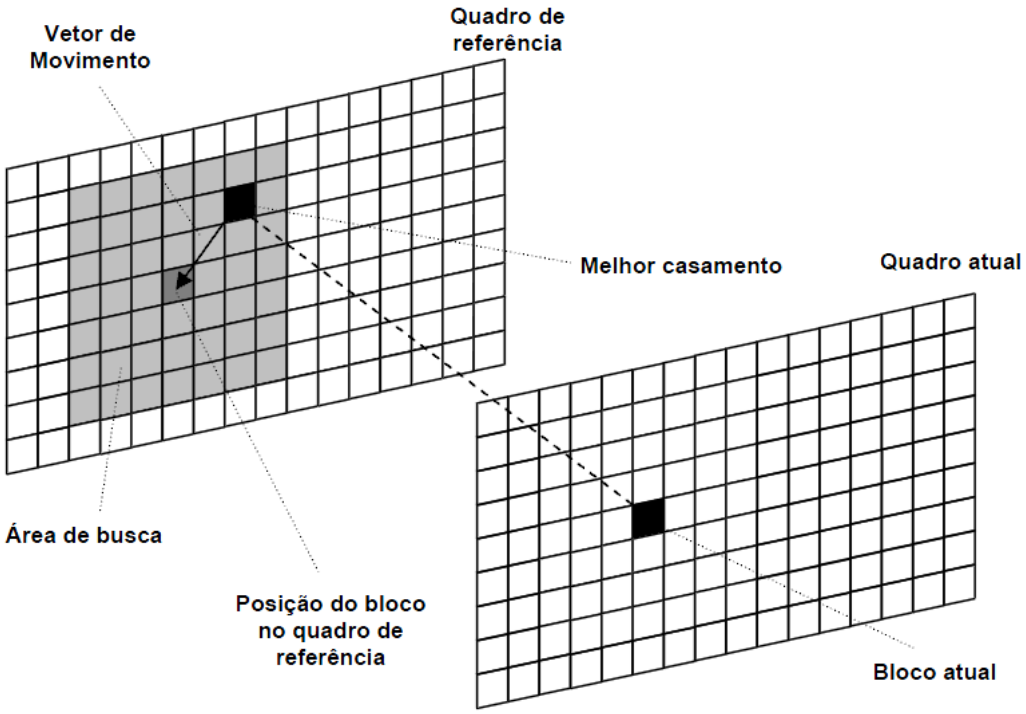

Figura 3.5 – Elementos da Estimação de Movimento

3.3.1.1 Estimação de Movimento Fracionária

A FME é uma técnica importante que pode ser empregada na ME com o objetivo de se obter resíduos de energia menor, e consequentemente, maior

eficiência na codificação. Basicamente, é realizada uma interpolação entre posições de amostras no quadro de referência permitindo a pesquisa de posições interpoladas de sub-pixel, além das posições inteiras de pixel como pode ser observado na Figura 3.6 (CORRÊA; SCHOENKNECHT, 2011a). A FME é utilizada pela maioria dos padrões atuais. O padrão H.264/AVC utiliza FME e prevê a utilização de vetores de movimento com precisão de ½ pixel e ¼ de pixel. No padrão HEVC, o processo de FME sofreu alterações que permitirão melhorias na qualidade da codificação, mas manteve a utilização de vetores de movimento com precisão de ½ pixel e ¼ de pixel.

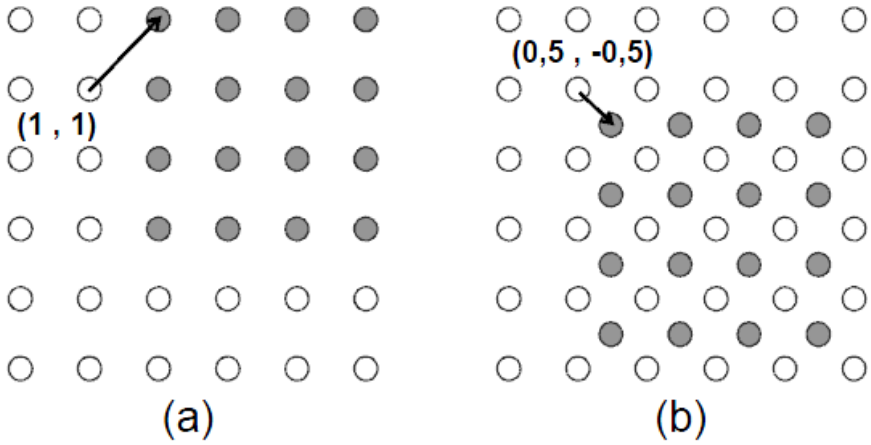

Figura 3.6 – Vetores de Movimento: (a) ME em posições inteiras e (b) FME.

Como mencionado anteriormente, a técnica de FME permite uma maior eficiência na codificação, uma vez que possibilita bons casamentos entre os blocos dos quadros atual e de referência, reduzindo os resíduos, e consequentemente, permitindo o processamento de um número menor de informações. Isto é possível porque os movimentos existentes em uma cena normalmente não se limitam a posições inteiras de pixel. Contudo, à medida que a precisão aumenta na FME, os ganhos obtidos em termos de compressão se tornam menores, já que aumenta também a complexidade computacional. Assim, o ganho de desempenho se reduz em decorrência da necessidade de mais vetores de movimento, e consequentemente, mais bits na representação da FME (CARVALHO, 2007). Desta forma, o desenvolvimento de arquiteturas de alto desempenho para a FME é de extrema importância para o processamento de vídeos digitais em tempo real.

O processo de FME é composto basicamente de duas etapas, uma etapa de interpolação, na qual são gerados os valores das amostras em posições de sub-pixel e uma etapa de busca, onde os valores gerados são comparados com o melhor casamento em posições inteiras (CORRÊA, 2011a) (CORRÊA, 2011b). Esta organização da FME pode ser observada na Figura 3.7.

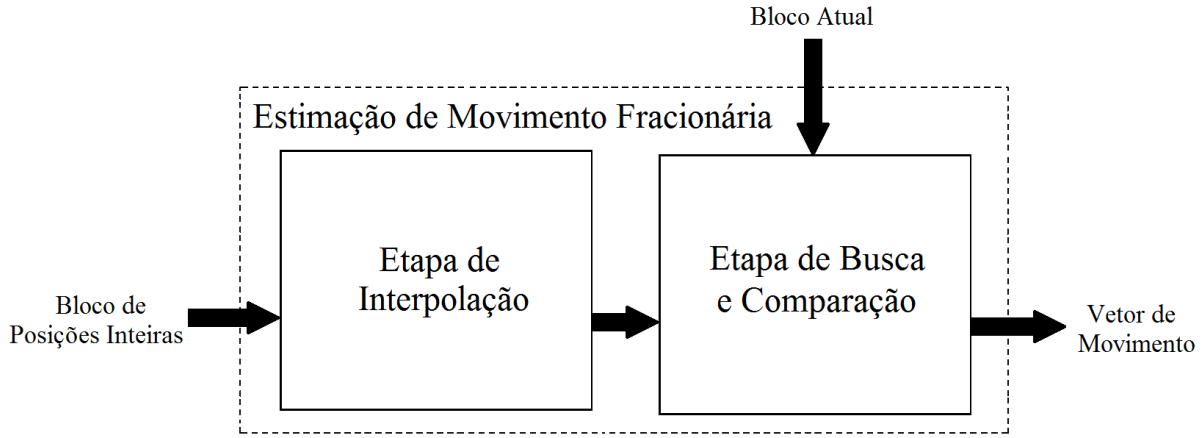

Figura 3.7 – Blocos da FME.

3.3.1.2 Comparação com Padrões de Codificação Anteriores

A interpolação de ¼ de pixel das amostras de luminância no padrão HEVC é realizada de forma diferente da empregada no padrão H.264/AVC. No padrão HEVC é proposta a utilização de um filtro de interpolação separável de 8 *taps*, baseado em DCT (*Discrete Cosine Transform)*, que interpola diretamente as amostras de luminância das posições inteiras de pixel, antes da busca fracionária por um melhor casamento em ¼ de pixel (McCANN *et al.,* 2012). No padrão H.264/AVC, a interpolação de ¼ de pixel das amostras de luminância é realizada em duas etapas. Na primeira destas etapas, as amostras em posições com precisão de ½ pixel são geradas a partir das amostras de luminância em posições inteiras através de um filtro FIR (*Finite Impulse Response)* de 6 *taps*. Ainda nesta etapa, após a interpolação, é realizada uma busca em posições fracionárias de ½ pixel que permita um casamento melhor do que o obtido através da busca em posições inteiras. Caso seja encontrado um melhor casamento em precisão de ½ pixel, é realizada uma segunda etapa. A segunda etapa consiste em gerar as amostras de luminância em posições de ¼ de pixel através de uma média simples entre uma posição de ½ pixel

obtida na etapa anterior e uma posição inteira. Também é feita nesta etapa a busca fracionária por um melhor casamento em ¼ de pixel (AGOSTINI, 2007).

3.3.1.3 Estado da Arte das Arquiteturas para FME

Uma vez que o padrão HEVC ainda se encontra em desenvolvimento, se torna fundamental um acompanhamento dos documentos sobre o padrão disponibilizados a cada *meeting* do JCT-VC, principalmente das versões do HEVC *Draft* e HEVC *Test Model*, os quais permitem identificar qualquer alteração que aconteça nas ferramentas de codificação e no software de referência do padrão. Estes documentos são essenciais para o desenvolvimento de qualquer trabalho envolvendo o padrão HEVC no momento e podem ser encontrados no site *http://phenix.int-evry.fr/jct/*.

Para permitir um estudo sobre o estado da arte das arquiteturas de FME segundo o padrão HEVC, bem como realizar futuras comparações de resultados, pesquisou-se na literatura científica alguns trabalhos relacionados. Contudo, não foram encontrados até a conclusão deste trabalho, artigos científicos que envolvam o desenvolvimento de arquiteturas para a FME com precisão de sub-pixel do HEVC. Desta forma, com o objetivo de identificar estratégias interessantes que poderiam ser utilizadas no projeto das arquiteturas, foram considerados para estudo trabalhos que apresentam arquiteturas para FME segundo o padrão H.264/AVC, do qual é possível encontrar inúmeros trabalhos relacionados na literatura científica (CORRÊA, 2011a) (CORRÊA, 2011b) (OKTEM, 2007) (KAO, 2006) (YALCIN, 2006). Alguns trabalhos para o H.264/AVC abordam o desenvolvimento de arquiteturas para precisão de ¼ de pixel e outros para ½ pixel. Os resultados obtidos, também são bem variados dependendo do foco dos projetos e quais requisitos são relevantes, como taxa de processamento, consumo de energia, qualidade resultante do vídeo, entre outros.

Corrêa (2011a) (2011b) apresenta em dois de seus trabalhos, arquiteturas para FME com abordagem focada em blocos de tamanho fixo 8x8. Em um dos trabalhos, a arquitetura proposta trabalha com precisão de ½ pixel, enquanto que no outro, a arquitetura foi desenvolvida para precisão de ¼ de pixel. Para adotar a decisão de utilizar tamanho de bloco fixo 8x8, Corrêa (2011a) (2011b) realizou avaliações com base no *bit-rate* e no PSNR (*Peak Signal-to-Noise Ratio*) de cinco

vídeos codificados utilizando o software de referência do padrão H.264/AVC. As duas abordagens de Corrêa (2011a) (2011b), as quais consideram tamanho de bloco fixo 8x8, contribuem para o aumento do *bit-rate* e redução do PSNR. Estas alterações têm efeitos importantes em dois aspectos relacionados a vídeo digital: a diminuição da qualidade da imagem e a diminuição do custo de implementação, uma vez que a complexidade computacional é menor e menos hardware é necessário. Contudo, conforme as avaliações de qualidade de imagem, a queda de qualidade não foi muito expressiva, considerando o tamanho de bloco utilizado.

Kao (2006) propõe uma arquitetura que emprega um modelo matemático para estimar SADs em posições de ¼ de pixel ao invés de realizar em sequência etapas de interpolação e busca, conforme o método tradicional. Desta forma, o tempo de computação e os requisitos de acesso à memória são reduzidos, sem prejuízo significativo na qualidade da imagem.

Oktem (2007) apresenta um hardware capaz de realizar a interpolação de ¼ de pixel da FME do padrão H.264/AVC de forma dinâmica, ou seja, apenas são calculadas as posições de ¼ de pixel necessárias para a realização da busca no local apontado pelo vetor de movimento de ½ pixel. Desta forma pode ser reduzida a quantidade de cálculos realizados para a interpolação de ¼ de pixel da FME e, portanto, há uma redução no consumo de energia do hardware.

Como pode ser observado, com escolhas adequadas é possível conseguir um bom equilíbrio entre qualidade de imagem, custo de hardware e desempenho. Na fase de desenvolvimento da arquitetura completa para FME deste trabalho serão realizadas avaliações com base no *bit-rate* e no PSNR de vídeos codificados, utilizando o software de referência do padrão HEVC e considerando alguns tamanhos fixos de bloco. Com base nos resultados, serão analisadas as quedas de qualidade de imagem com diferentes tamanhos de bloco. Estas ações permitirão encontrar um bom equilíbrio entre qualidade de imagem, custo de hardware e taxa de processamento na especificação do projeto de hardware.

4 METODOLOGIA

Neste capítulo são apresentados os softwares utilizados para o desenvolvimento das arquiteturas de hardware, bem como a metodologia empregada no desenvolvimento e os diagramas das arquiteturas.

4.1 Softwares Utilizados

Para que fosse possível o desenvolvimento das arquiteturas de hardware dos filtros de interpolação foram necessários basicamente dois softwares, além do código de referência. Um destes softwares foi utilizado para o projeto do hardware, permitindo a descrição do hardware em linguagem VHDL (VHSIC *Hardware Description Language)*, e o outro para auxiliar no processo de validação de dados. Para a descrição do hardware em VHDL foi utilizado o software Quartus II versão 9.1, da Altera. Por sua vez, o processo de validação foi feito com auxílio do software ModelSim versão 6.5, também da Altera e de informações obtidas a partir do software de referência, o que será melhor explicado nas seções seguintes.

4.1.1 Software de Referência

O software de referência do padrão HEVC é muito importante, pois fornece as condições para a realização de experimentos que possibilitem avaliar o quanto o desempenho de cada ferramenta de codificação é satisfatório. Todas as versões do software de referência do HEVC podem ser encontradas no site *https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/*, sendo possível o download de qualquer versão através do *Apache Subversion*, um sistema de controle de versão. Para fazer o download, utilizando o sistema operacional Linux,

basta utilizar o comando *svn checkout* no terminal do Linux. Após executar o comando, uma pasta HM, contendo diversas outras subpastas, é disponibilizada. De posse dos arquivos, a execução do software de referência ainda pode ser feita de diversas formas. Por exemplo, pode-se executar o bloco codificador ou o bloco decodificador. Se escolhido o bloco codificador diversas configurações para teste estão disponíveis, e cada uma delas conduz a experimentos em um ambiente de teste bem definido. O JCT-VC define oito condições de teste (BOSSEN, 2012), refletindo uma combinação de alta eficiência e baixa complexidade baseadas em configurações denominadas *intra-only*, *random-access* e *low-delay*. Para executar o codificador deve-se primeiramente compilar o código, o que pode ser simplificado através de um arquivo *Makefile* fornecido na pasta *HM/build/linux/app/TAppEncoder/* e um comando *make* no terminal do Linux. Após ser gerado o arquivo executável que ficará na pasta *HM/bin/TAppEncoderStatic* é necessário executá-lo fornecendo dois conjuntos de informações, as configurações do vídeo a ser codificado e as configurações relativas ao funcionamento do codificador. A seguinte linha de comando pode ser utilizada no terminal do Linux a partir da pasta HM para a execução do codificador considerando a sequência *SlideShow*, a configuração *Low-delay* e o perfil *Main*: *./bin/TAppEncoderStatic –c cfg/per-sequence/SlideShow.cfg –c cfg/encoder_lowdelay_main.cfg*.

Os arquivos de configuração das sequências de vídeo de teste definidas pelo JCT-VC (BOSSEN, 2012) podem ser encontrados na pasta *HM/cfg/per-sequence/*. Já os arquivos de configuração do codificador ficam na pasta *HM/cfg/*. Os arquivos de configuração das sequências fornecem, entre outras informações, o endereço do arquivo de vídeo a ser codificado, a taxa de quadros por segundo, a resolução do vídeo, o *bit depth* de entrada e o número de quadros a serem codificados. Os arquivos de configuração do codificador fornecem inúmeras informações, relativas a todas as ferramentas de codificação utilizadas, como predição inter/intra, transformadas, quantização e codificação de entropia. Entre as informações do arquivo, estão o tamanho máximo das unidades de codificação, a área de busca para a estimação de movimento e o número de quadros de referência.

Alguns experimentos estão sendo realizados com o auxílio do software de referência com o objetivo de realizar avaliações importantes relacionadas com este trabalho. Em trabalhos futuros pretende-se, através da variação de configurações e

alterações no código do software, extrair informações como: o impacto da ME e da FME nos resultados de compressão; os tamanhos de PU mais selecionados, ou seja, que apresentam os melhores resultados na codificação; resultados de qualidade de imagem e *bit-rate* fixando o tamanho dos blocos. Conhecendo-se os tamanhos de blocos mais importantes para a predição inter-quadros, poderão ser adotadas estratégias para o desenvolvimento das arquiteturas de hardware que permitam um equilíbrio entre desempenho, qualidade resultante de imagem e custo de hardware. Outro importante papel do software de referência é a contribuição para validação das arquiteturas propostas. Com alterações no código de referência se pode obter, por exemplo, os valores das amostras de luminância antes e após a interpolação realizada na FME, possibilitando a validação com o auxílio da ferramenta ModelSim.

4.2 Arquiteturas de Hardware Desenvolvidas e Metodologia Utilizada

Este trabalho apresenta o desenvolvimento de arquiteturas de hardware para FME capazes de realizar a interpolação das amostras de luminância com precisão de ¼ de pixel do padrão HEVC. Basicamente, a FME pode ser dividida em duas etapas: interpolação e busca. Por sua vez, a etapa de interpolação, também pode ser dividida em partes, como memória de entrada, filtros de interpolação e memória de saída. O foco principal deste trabalho está no desenvolvimento da etapa de interpolação, na qual os filtros que geram as amostras em posições de sub-pixel têm papel fundamental. Portanto, o desenvolvimento da etapa de interpolação passou primeiramente pelo desenvolvimento dos filtros, cuja localização pode ser observada na Figura 4.1. Para o projeto foi considerado o perfil *Main* do HEVC, com amostras de luminância com 8 bits, do tipo *unsigned* em que a interpolação será realizada a partir de blocos de PU com tamanho fixo 8x8, o que reduz a complexidade computacional consideravelmente.

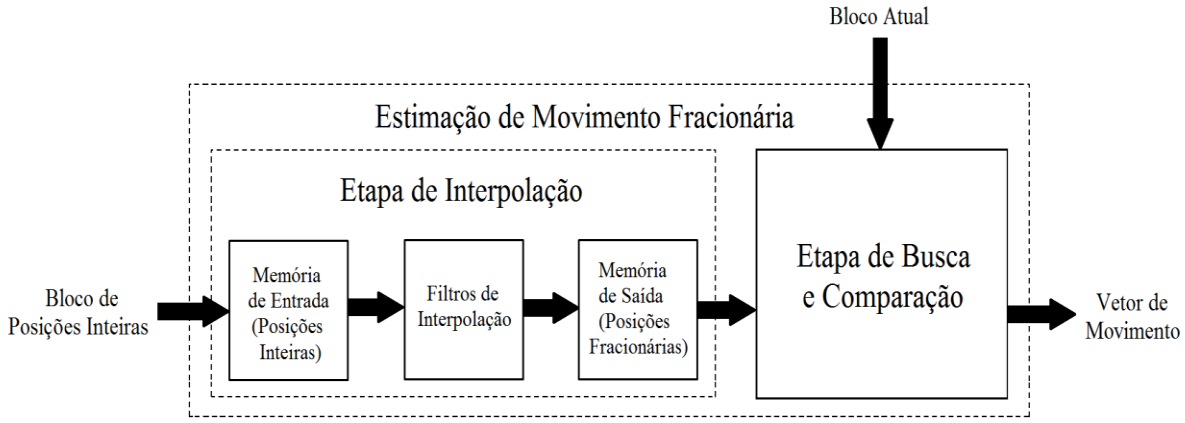

Figura 4.1 – Diagrama da FME com detalhamento da etapa de interpolação.

Desde a chamada de propostas para o padrão HEVC, a normatização da FME tem sofrido algumas alterações na medida em que avançam as contribuições com relação à ferramenta, como pode ser observado nos *drafts* do padrão ou em documentos específicos do JCT-VC (ALSHINA *et al.*, 2011). As arquiteturas de hardware desenvolvidas foram baseadas em dois dos mais recentes documentos elaborados pelo JCT-VC até o término deste trabalho: HM6 – *High Efficiency Video Coding Test Model 6 Encoder Description* (McCANN *et al.,* 2012) e *High Efficiency Video Coding text specification draft* 7 (BROSS *et al.,* 2012). Nestes documentos é possível encontrar em detalhes o processo para a geração das posições de sub-pixel segundo o padrão HEVC. A Figura 4.2 representa as amostras em posições inteiras, bem como as amostras em posições fracionárias para interpolação das amostras de luminância com precisão de ¼ de pixel do padrão HEVC (BROSS *et al.,* 2012, p. 131). Já a Figura 4.3 permite a visualização de todas as posições, sejam inteiras ou fracionárias de um bloco 8x8.

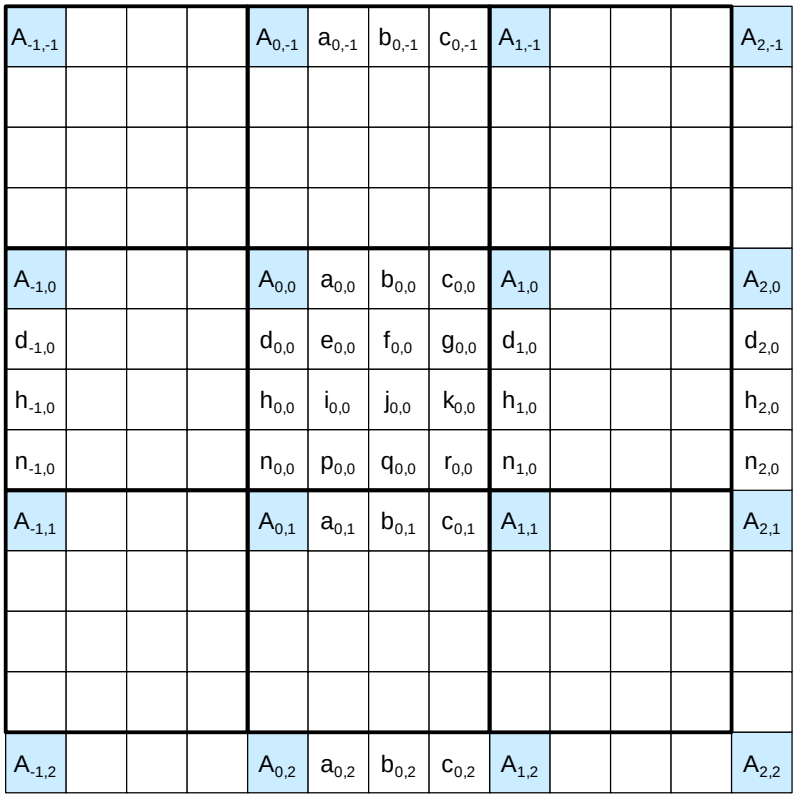

Figura 4.2 – Amostras em posições inteiras (quadrados sombreados e com letras maiúsculas) e amostras em posições fracionárias (quadrados não sombreados e com letras minúsculas) para interpolação das amostras de luminância com precisão de ¼ de pixel do padrão HEVC.

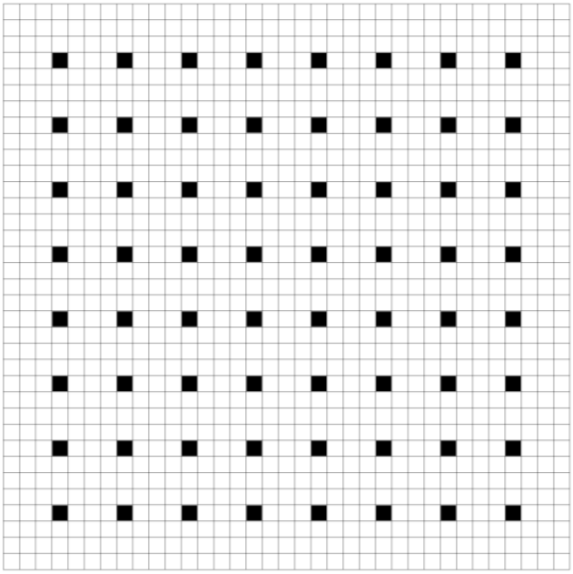

Figura 4.3 – Amostras em posições inteiras (quadrados sombreados) e amostras em posições fracionárias (quadrados não sombreados) para interpolação das amostras de luminância considerando um bloco de tamanho 8x8.

A partir das amostras de luminância em posições inteiras e de filtros FIR de 8 *taps*, são obtidos os valores das posições fracionárias $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$ e $n_{0,0}$, como pode ser observado nas equações 1-6. Estas equações nada mais são do que os algoritmos retirados do documento HEVC *draft* 7 (BROSS *et al.,* 2012, p. 132).

$$a_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 10*A_{-1,0} + 58*A_{0,0} + 17*A_{1,0} - 5*A_{2,0} + A_{3,0}) >> shift1 \quad (1)$$

$$b_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 11*A_{-1,0} + 40*A_{0,0} + 40*A_{1,0} - 11*A_{2,0} + 4*A_{3,0} - A_{4,0}) >> shift1 \quad (2)$$

$$c_{0,0} = (A_{-2,0} - 5*A_{-1,0} + 17*A_{0,0} + 58*A_{1,0} - 10*A_{2,0} + 4*A_{3,0} - A_{4,0}) >> shift1 \quad (3)$$

$$d_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 10*A_{0,-1} + 58*A_{0,0} + 17*A_{0,1} - 5*A_{0,2} + A_{0,3}) >> shift1 \quad (4)$$

$$h_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 11*A_{0,-1} + 40*A_{0,0} + 40*A_{0,1} - 11*A_{0,2} + 4*A_{0,3} - A_{0,4}) >> shift1 \quad (5)$$

$$n_{0,0} = (A_{0,-2} - 5*A_{0,-1} + 17*A_{0,0} + 58*A_{0,1} - 10*A_{0,2} + 4*A_{0,3} - A_{0,4}) >> shift1 \quad (6)$$

O cálculo dos valores das posições fracionárias $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$ requerem primeiramente o cálculo dos valores das posições fracionárias $a_{0,i}$, $b_{0,i}$ e $c_{0,i}$, onde i varia de -3 a 4 na direção vertical. Em seguida, através de filtros FIR de 8 *taps*, são obtidos os valores das demais posições fracionárias, como pode ser observado nas equações 7-15.

$$e_{0,0} = (-a_{0,-3} + 4*a_{0,-2} - 10*a_{0,-1} + 58*a_{0,0} + 17*a_{0,1} - 5*a_{0,2} + a_{0,3}) >> shift2 \quad (7)$$

$$f_{0,0} = (-b_{0,-3} + 4*b_{0,-2} - 10*b_{0,-1} + 58*b_{0,0} + 17*b_{0,1} - 5*b_{0,2} + b_{0,3}) >> shift2 \quad (8)$$

$$g_{0,0} = (-c_{0,-3} + 4*c_{0,-2} - 10*c_{0,-1} + 58*c_{0,0} + 17*c_{0,1} - 5*c_{0,2} + c_{0,3}) >> shift2 \quad (9)$$

$$i_{0,0} = (-a_{0,-3} + 4*a_{0,-2} - 11*a_{0,-1} + 40*a_{0,0} + 40*a_{0,1} - 11*a_{0,2} + 4*a_{0,3} - a_{0,4}) >> shift2 \quad (10)$$

$$j_{0,0} = (-b_{0,-3} + 4*b_{0,-2} - 11*b_{0,-1} + 40*b_{0,0} + 40*b_{0,1} - 11*b_{0,2} + 4*b_{0,3} - b_{0,4}) >> shift2 \quad (11)$$

$$k_{0,0} = (-c_{0,-3} + 4*c_{0,-2} - 11*c_{0,-1} + 40*c_{0,0} + 40*c_{0,1} - 11*c_{0,2} + 4*c_{0,3} - c_{0,4}) >> shift2 \quad (12)$$

$$p_{0,0} = (a_{0,-2} - 5*a_{0,-1} + 17*a_{0,0} + 58*a_{0,1} - 10*a_{0,2} + 4*a_{0,3} - a_{0,4}) >> shift2 \quad (13)$$

$$q_{0,0} = (b_{0,-2} - 5*b_{0,-1} + 17*b_{0,0} + 58*b_{0,1} - 10*b_{0,2} + 4*b_{0,3} - b_{0,4}) >> shift2 \quad (14)$$

$$r_{0,0} = (c_{0,-2} - 5*c_{0,-1} + 17*c_{0,0} + 58*c_{0,1} - 10*c_{0,2} + 4*c_{0,3} - c_{0,4}) >> shift2 \quad (15)$$

Com o objetivo de reduzir a quantidade de arquiteturas de hardware diferentes necessárias para implementação dos filtros de interpolação do padrão HEVC, foi feita uma análise dos cálculos a serem realizados para determinação de cada um dos valores das posições fracionárias. Em um primeiro momento se observou uma similaridade entre algumas posições, como pode ser observado na Tabela 4.1. Inicialmente, foram identificados 4 grupos diferentes de hardware para construção da arquitetura dos filtros de interpolação. As posições $a_{0,0}$, $c_{0,0}$, $d_{0,0}$, e

$n_{0,0}$, por exemplo, podem utilizar o mesmo hardware, uma vez que apesar das posições inteiras utilizadas para os cálculos serem diferentes, as constantes utilizadas nas multiplicações são as mesmas para as 4 posições.

Tabela 4.1 – Posições fracionárias e similaridades entre as posições nos algoritmos.

Posições Fracionárias	**Similaridade**
$a_{0,0}$, $c_{0,0}$, $d_{0,0}$ e $n_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –5, –10, 1, 4, 17 e 58) e *shift1*.
$b_{0,0}$ e $h_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40) e *shift1*.
$e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –5, –10, 1, 4, 17 e 58) e *shift2*.
$i_{0,0}$, $j_{0,0}$ e $k_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40) e *shift2*.

Contudo, uma análise mais detalhada a respeito das variáveis utilizadas para o cálculo dos valores de luminância com precisão de sub-pixel, permitiu uma redução na quantidade de filtros diferentes a serem desenvolvidos em hardware. As informações sobre essas variáveis são definidas segundo o documento HEVC *draft* 7 (BROSS *et al.,* 2012) pelas seguintes expressões.

- *bit_depth_luma_minus8* pode ter valores de 0 a 6
- $BitDepth_Y = 8 + bit\_depth\_luma\_minus8$
- $shift1 = BitDepth_Y - 8$
- $shift2 = 6$

Analisando o software de referência do HEVC (2012), e o documento observou-se que a variável $BitDepth_Y$ está definida como uma constante cujo valor é 14. De posse do valor de $BitDepth_Y$ e de acordo com as expressões definidas no documento HEVC *draft* 7 (BROSS *et al.,* 2012), é possível calcular os valores das demais variáveis, como segue abaixo.

- *bit_depth_luma_minus8* = 6
- $BitDepth_Y = 14$
- *shift1* = 6
- *shift2* = 6

Através dos resultados é possível observar que os valores de *shift1* e *shift2* são idênticos. Portanto, a fim de diminuir o número de arquiteturas a serem desenvolvidas, as posições de sub-pixel foram separadas em dois grupos de acordo com as constantes utilizadas nas multiplicações. A Tabela 4.2 apresenta os dois tipos de filtros que foram implementados de acordo com as constantes utilizadas nas multiplicações. Foi chamado de filtro tipo *Up/Down*, o filtro utilizado para o cálculo das posições fracionárias $a_{0,0}$, $c_{0,0}$, $d_{0,0}$, $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $n_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$. Por sua vez, o filtro tipo *Middle* é utilizado para o cálculo das posições fracionárias $b_{0,0}$, $h_{0,0}$, $i_{0,0}$, $j_{0,0}$ e $k_{0,0}$.

Tabela 4.2 – Arranjo final das posições fracionárias para o desenvolvimento das arquiteturas.

Filtro	**Posições Fracionárias**	**Similaridade**
Tipo *Up/Down*	$a_{0,0}$, $c_{0,0}$, $d_{0,0}$, $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $n_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –5, –10, 1, 4, 17 e 58).
Tipo *Middle*	$b_{0,0}$, $h_{0,0}$, $i_{0,0}$, $j_{0,0}$ e $k_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40).

A decisão de chamar os filtros de tipo *Up*, tipo *Middle* e tipo *Down* se deve a localização das posições fracionárias calculadas por cada tipo de filtro, como pode ser observado na Tabela 4.3 e na Figura 4.4. Na Figura 4.4 são destacados em (a), os três tipos de posições fracionárias, em (b), as posições fracionárias para o filtro tipo *Up*, em (c), as posições fracionárias para o filtro tipo *Middle* e em (d), as posições fracionárias para o filtro tipo *Down*.

Tabela 4.3 – Tipos de filtros e suas respectivas posições fracionárias.

Filtro	**Posições Fracionárias**
Tipo *Up*	$a_{0,0}$, $d_{0,0}$, $e_{0,0}$, $f_{0,0}$ e $g_{0,0}$
Tipo *Middle*	$b_{0,0}$, $h_{0,0}$, $i_{0,0}$, $j_{0,0}$ e $k_{0,0}$
Tipo *Down*	$c_{0,0}$, $n_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$

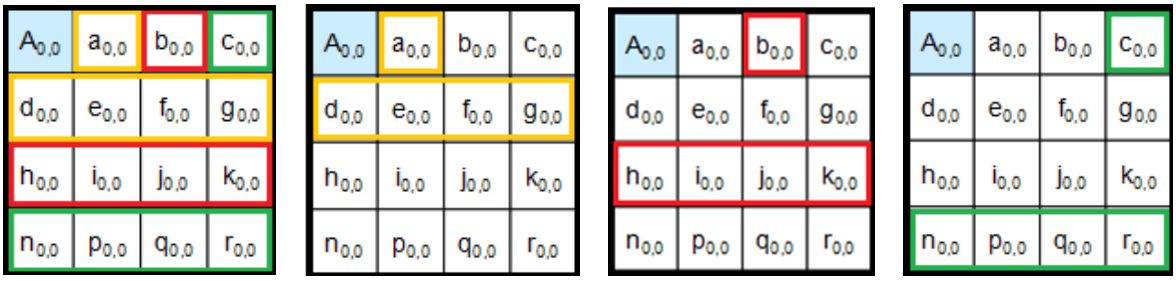

Figura 4.4 – Localização das posições fracionárias de acordo com o tipo do filtro.

Antes da implementação em hardware, cada tipo de filtro passou por uma análise a fim de se localizar outras possibilidades de otimização. Para o filtro tipo *Up/Down*, por exemplo, foram realizadas várias otimizações. Com o objetivo de dar maior clareza às otimizações realizadas, as representações das amostras inteiras e fracionárias no documento HEVC *draft* 7 (BROSS *et al.,* 2012) foram substituídas por a_0-a_6. Primeiramente, as multiplicações e as constantes utilizadas foram reescritas na forma de soma/subtração de multiplicações cujas constantes são números na base 2, como pode ser observado na Equação 17. Esta estratégia permite a realização de multiplicações utilizando somente operações de soma/subtração e deslocamentos, diminuindo consideravelmente o custo de hardware dos multiplicadores. Uma vez que cada constante poderia ser substituída por mais de um conjunto de soma/subtração de multiplicações de constantes base 2, foram escolhidos os conjuntos que utilizaram um número menor de operações de soma/subtração, e consequentemente, menor custo de hardware. Por exemplo, – $10*a_2$ pode ser reescrito como – $(8*a_2 + 2*a_2)$, mas também poderia ser reescrito como – $(16*a_2 - 4*a_2 - 2*a_2)$. Porém, neste último caso a escolha acarretaria em um maior custo de hardware e desempenho. Em seguida, foram feitos agrupamentos de forma que as multiplicações por constantes na base 2 fossem realizadas apenas uma vez, como pode ser observado na Equação 18. Isto equivale a dizer que os deslocamentos necessários podem ser realizados apenas uma vez. Finalmente, pode ser visto na Equação 19 o resultado final das otimizações, já com os deslocamentos a serem realizados. Por exemplo, multiplicar por 2 é equivalente a um deslocamento à esquerda de um bit, o que apresenta um custo muito menor do que uma multiplicação convencional.

$$\text{Sub-pixel} = [-a_0 + 4*a_1 - 10*a_2 + 58*a_3 + 17*a_4 - 5*a_5 + a_6] >> 6 \quad (16)$$

$$\text{Sub-pixel} = [-a_0 + 4*a_1 - (8*a_2 + 2*a_2) + (64*a_3 - 4*a_3 - 2*a_3) + (16*a_4 + a_4) - (4*a_5 + a_5) + a_6] >> 6 \quad (17)$$

$$\text{Sub-pixel} = [1*(-a_0 + a_4 - a_5 + a_6) - 2*(a_2 + a_3) + 4*(a_1 - a_3 - a_5) - 8*(a_2) + 16*(a_4) + 64*(a_3)\ ] >> 6 \quad (18)$$

$$\text{Sub-pixel} = [(-a_0 + a_4 - a_5 + a_6) - (a_2 + a_3) << 1 + (a_1 - a_3 - a_5) << 2 - (a_2) << 3 + (a_4) << 4 + (a_3) << 6] >> 6 \quad (19)$$

No filtro tipo *Middle* também foram realizadas várias otimizações similares às do filtro anterior. Foram utilizadas estratégias de reaproveitamento de hardware e multiplicações baseadas em soma/subtração e deslocamentos. Nas equações 20-24 pode ser observado todo o processo de otimização para o segundo filtro. Uma vantagem que pode ser observada na otimização do filtro tipo *Middle* foi a possibilidade de reaproveitar hardware nas operações de soma/subtração, como pode ser observado em negrito na Equação 23.

$$\text{Sub-pixel} = [-a_0 + 4*a_1 - 11*a_2 + 40*a_3 + 40*a_4 - 11*a_5 + 4*a_6 - a_7] >> 6 \quad (20)$$

$$\text{Sub-pixel} = [-a_0 + 4*a_1 - (8*a_2 + 2*a_2 + a_2) + (32*a_3 + 8*a_3) + (32*a_4 + 8*a_4) - (8*a_5 + 2*a_5 + a_5) + 4*a_6 - a_7] >> 6 \quad (21)$$

$$\text{Sub-pixel} = [1*(-a_0 - a_2 - a_5 - a_7) + 2*(-a_2 - a_5) + 4*(a_1 + a_6) + 8*(-a_2 + a_3 + a_4 - a_5) + 32*(a_3 + a_4)] >> 6 \quad (22)$$

$$\text{Sub-pixel} = [1*(-a_0 - a_7 - \mathbf{(a_2 + a_5)}) - 2*\mathbf{(a_2 + a_5)} + 4*(a_1 + a_6) + 8*(\mathbf{(a_3 + a_4)} - \mathbf{(a_2 + a_5)}) + 32*\mathbf{(a_3 + a_4)}] >> 6 \quad (23)$$

$$\text{Sub-pixel} = [(-a_0 - a_7 - \mathbf{(a_2 + a_5)}) - \mathbf{(a_2 + a_5)} << 1 + (a_1 + a_6) << 2 + (\mathbf{(a_3 + a_4)} - \mathbf{(a_2 + a_5)}) << 3 + \mathbf{(a_3 + a_4)} << 5] >> 6 \quad (24)$$

Após a realização das otimizações, os filtros de interpolação foram descritos em VHDL através da ferramenta Quartus II da Altera. No desenvolvimento do hardware, primeiramente foram descritas em VHDL duas arquiteturas puramente combinacionais, uma para cada tipo de filtro. Com a finalidade de obter melhores resultados em termos de desempenho, na sequência, foram desenvolvidas versões com dois e quatro estágios de *pipeline* destes filtros. Os tamanhos das entradas a_0-a_7 utilizadas nas equações 19 e 24 poderiam ser de 8 bits, considerando como entradas valores de amostras em posições inteiras, uma vez que o tamanho adotado no projeto para as amostras de luminância foi de 8 bits, sem sinal. Nesse caso, as saídas dos filtros apresentariam valores entre -64 e 319 ou -96 e 351 para algumas das posições fracionárias, dependendo do tipo de filtro. Contudo, os filtros das equações 19 e 24 podem ser utilizados considerando como entradas valores de amostras de posições fracionárias, requerendo que o tamanho das entradas seja de 10 bits, como pode se observado nas tabelas 4.4 e 4.5. Dessa forma, como o filtro funcionará com *pipeline* e um mesmo filtro poderá receber na entrada 8 ou 10 bits, dependendo das posições a serem calculadas, o tamanho adotado para as entradas dos filtros foi de 10 bits.

Tabela 4.4 – Faixa de valores possíveis nas entradas e saída do filtro tipo *Up/Down*.

Filtro Tipo *Up/Down*		
Posições Fracionárias	$a_{0,0}$, $c_{0,0}$, $d_{0,0}$ e $n_{0,0}$	$e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $p_{0,0}$, $q_{0,0}$ e $r_{0,0}$
Valores possíveis nas entradas a_0 a a_6	0 a 255	-64 a 319
Número de bits necessários nas entradas	8	10
Menor valor possível na saída	-64	-160
Maior valor possível na saída	319	415
Número de bits necessários na saída	10	10

Tabela 4.5 – Faixa de valores possíveis nas entradas e saída do filtro tipo *Middle*.

Filtro Tipo *Middle*		
Posições Fracionárias	$b_{0,0}$ e $h_{0,0}$	$i_{0,0}$, $j_{0,0}$ e $k_{0,0}$
Valores possíveis nas entradas a_0 a a_7	0 a 255	-96 a 351
Número de bits necessários nas entradas	8	10
Menor valor possível na saída	-96	-264
Maior valor possível na saída	351	519
Número de bits necessários na saída	10	11

As duas arquiteturas com 4 estágios de *pipeline* desenvolvidas podem ser observadas nas figuras 4.5 e 4.6, onde são apresentados os esquemas para os filtros tipo *Up/Down* e tipo *Middle*, respectivamente.

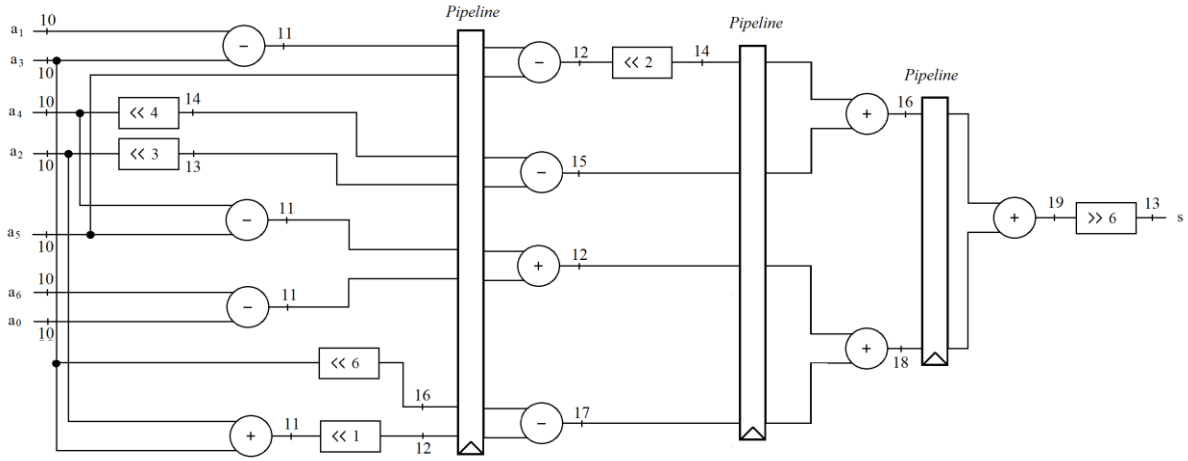

Figura 4.5 – Arquitetura de hardware para o filtro tipo *Up/Down*.

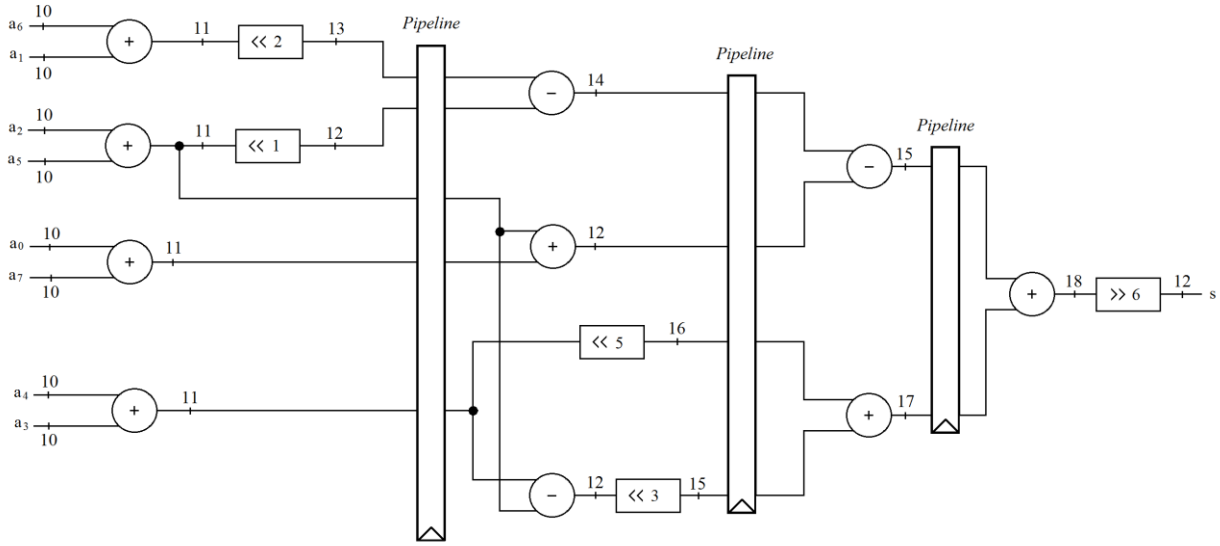

Figura 4.6 – Arquitetura de hardware para o filtro tipo *Middle*.

Como pode ser observado nos esquemas das figuras 4.5 e 4.6, os somadores e subtratores podem apresentar diferentes tamanhos em cada uma de suas entradas. Dessa forma, é necessária a utilização de concatenação à esquerda a fim de que as entradas de um mesmo somador ou subtrator tenham o mesmo número de bits. A concatenação consiste em copiar o bit mais significativo, quantas vezes sejam necessárias à esquerda até que as entradas tenham o mesmo tamanho. As operações de concatenação não foram representadas nos esquemas. Contudo, estas operações estão presentes em todos os somadores e subtratores cujas entradas não apresentam o mesmo tamanho. Outra característica das operações de

soma e subtração é que a saída apresenta um bit a mais em relação às entradas devido ao *carry* de cada somador e subtrator.

Para o desenvolvimento da arquitetura completa da etapa de interpolação ainda são necessárias uma memória de entrada, para o armazenamento das amostras de luminância em posições inteiras, e uma memória de saída, para conter os valores das amostras em posições fracionárias. Inicialmente, estas memórias foram implementadas em nível de registradores. As informações armazenadas por estes registradores serão necessárias na busca e comparação de blocos fracionários com o bloco que apresentou o melhor resultado na ME inteira. A Figura 4.7 mostra a arquitetura proposta para toda a etapa de interpolação.

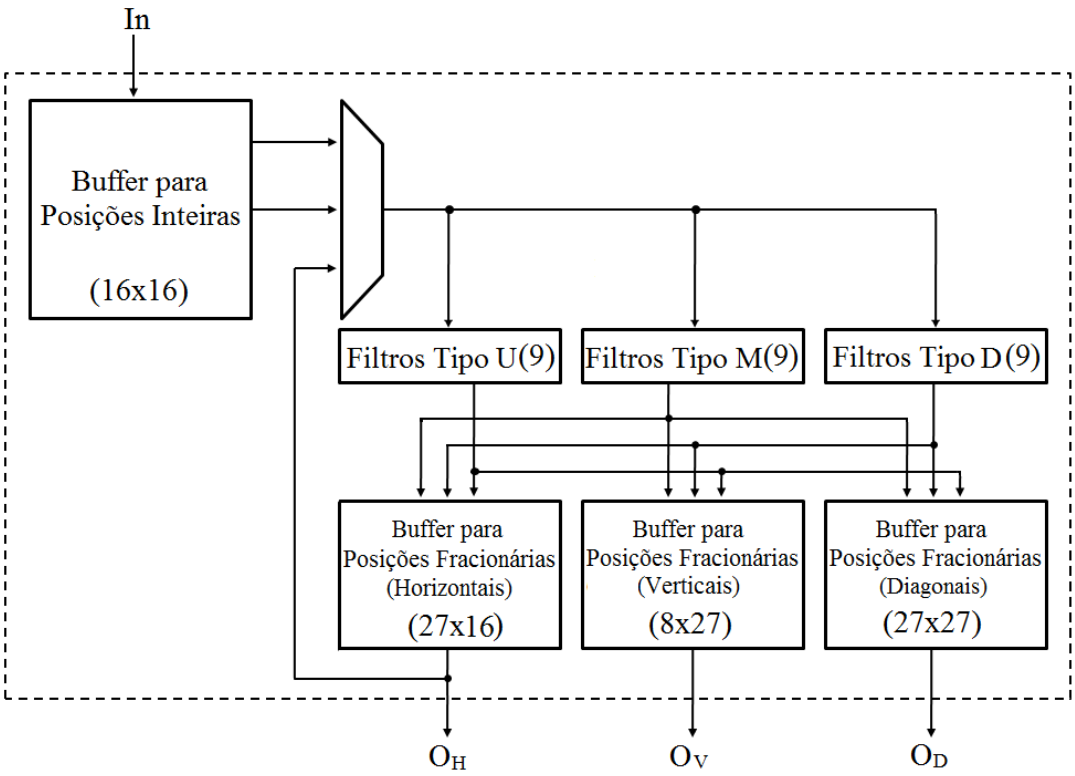

Figura 4.7 – Arquitetura de hardware para a etapa de interpolação.

A arquitetura completa para a etapa de interpolação apresenta um buffer para posições inteiras que deve armazenar 256 posições de 8 bits, uma vez que devem ser armazenadas, além das amostras em posições inteiras do bloco 8x8, uma borda de mais 4 amostras ao redor das amostras do bloco. A borda é necessária para o cálculo das posições fracionárias que não estão no centro do bloco. A Figura 4.8 ilustra como a borda é utilizada, onde as amostras inteiras ou fracionárias dentro do

quadro vermelho representam as amostras do bloco 8x8, e as amostras fora do quadrado vermelho, a borda. Em verde podemos observar a posição fracionária ‘b’ cujo valor deve ser calculado, e em azul, as amostras necessárias para o cálculo de acordo com o algoritmo da posição ‘b’.

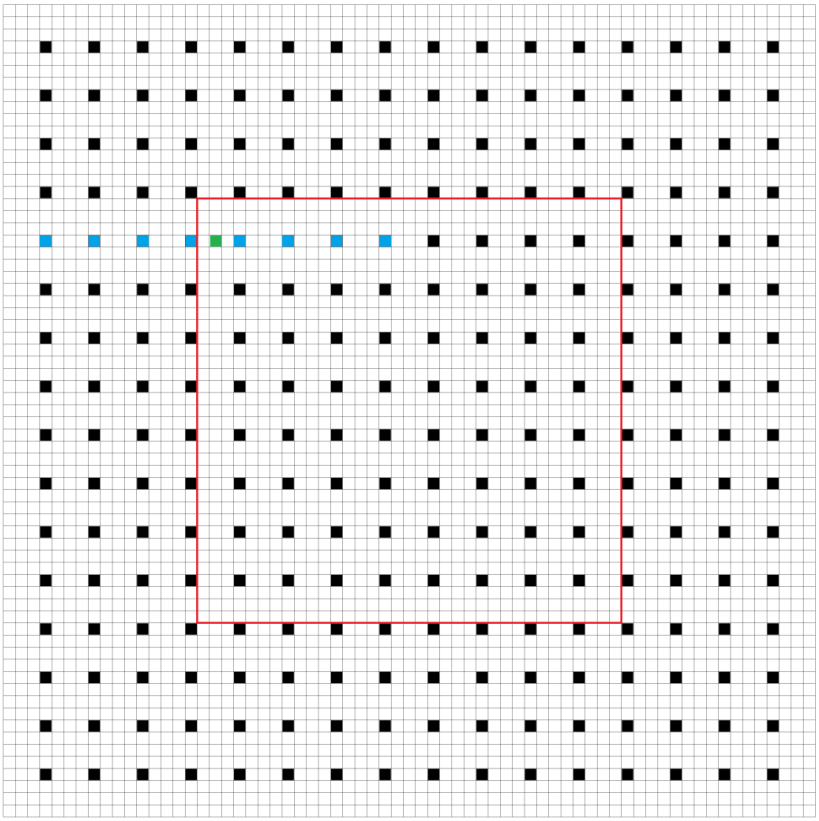

Figura 4.8 – Representação de bloco 8x8, com amostras em posições inteiras e fracionárias junto à borda necessária para a interpolação das amostras.

Para interpolação das amostras em posições inteiras optou-se por um esquema capaz de realizar o cálculo de uma linha ou coluna inteira de amostras em posições fracionárias a cada ciclo. Portanto, foram necessários 3 conjuntos de filtros (*Up*, *Middle* e *Down*), cada um com 9 unidades, permitindo o cálculo de 9 posições fracionárias por conjunto, ou 27 posições por ciclo. No total, para um bloco de tamanho 8x8 devem ser calculadas 432 posições fracionárias horizontais em relação as posições inteiras, 216 posições fracionárias verticais e 729 posições fracionárias diagonais, como pode ser observado em verde na Figura 4.9. Na Figura 4.10 são destacados em (a), os três tipos de posições fracionárias de acordo com os *buffers*, em (b), as posições fracionárias horizontais, em (c), as posições fracionárias verticais e em (d), as posições fracionárias diagonais. As posições fracionárias na

borda que devem ser calculadas são necessárias devido ao fato de algumas posições fracionárias requererem outras posições fracionárias (horizontais) como entrada, o que pode ser observado na Tabela 4.6. Os *buffers* para armazenamento de posições horizontais e verticais trabalham com amostras de 10 bits, enquanto o *buffer* para posições diagonais trabalha com amostras de 11 bits, conforme explicado anteriormente. O multiplexador tem a função de selecionar as amostras de entrada dos conjuntos de filtros, vindas do *buffer* de posições inteiras ou de posições fracionárias horizontais, de acordo com uma etapa de controle.

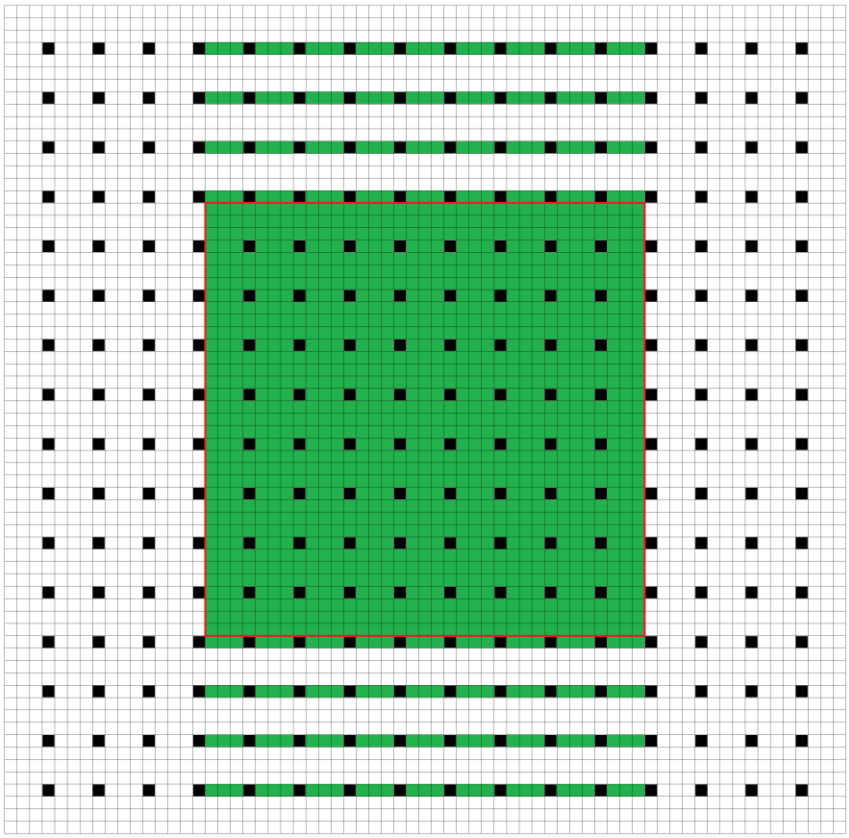

Figura 4.9 – Posições fracionárias que devem ser calculadas.

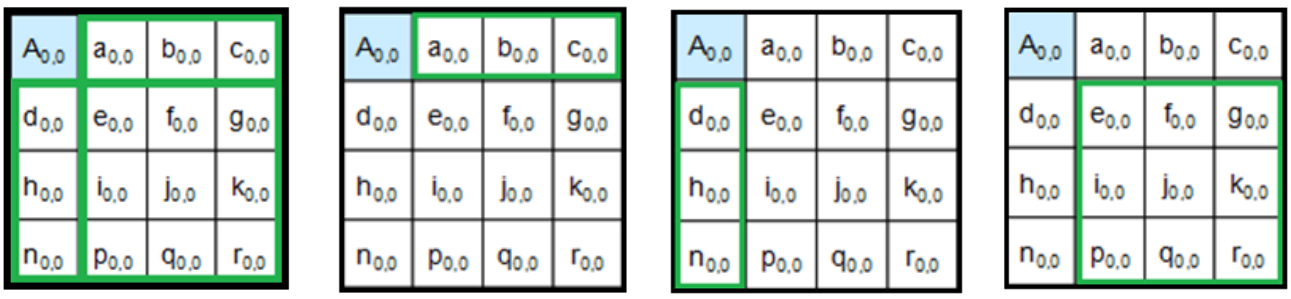

Figura 4.10 – Relação das posições fracionárias de acordo com os *buffers*.

A Tabela 4.6 também permite entender como se dão os cálculos das amostras em posições fracionárias. Primeiramente, a partir das amostras em posições inteiras horizontais de uma linha são calculadas todas as posições fracionárias horizontais dessa linha, ou seja, todas as posições ‘a’, ‘b’ e ‘c’. Esse processo é feito para as 16 linhas. Em seguida, utilizando amostras em posições verticais inteiras de uma coluna são calculadas as amostras em posições fracionárias verticais da coluna, ou seja, ‘d’, ‘h’ e ‘n’. Este processo deve ser repetido 8 vezes, pois apenas 8 das colunas precisam ser processadas. Por fim, a partir das amostras em posições fracionárias ‘a’, ‘b’ e ‘c’ recém-calculadas são calculadas as amostras das demais posições fracionárias. São calculadas todas as colunas que contém as posições ‘e’, ‘i’ e ‘p’ utilizando posições ‘a’, todas as colunas de posições ‘f’, ‘j’ e ‘q’ a partir das posições ‘b’ e todas as colunas de posições ‘q’, ‘k’ e ‘r’ usando posições ‘c’, o que totaliza mais 27 colunas. De acordo com o número de linhas e colunas a serem processadas pode-se estimar um total de 52 ciclos para a completa interpolação de um bloco 8x8.

Tabela 4.6 – Posições de entrada para o cálculo das posições fracionárias.

Posições Fracionárias	**Posições utilizadas para o cálculo**	
$a_{0,0}$	$A_{-3,0}$ a $A_{3,0}$	$A_{-3,0}$ a $A_{4,0}$
$b_{0,0}$	$A_{-3,0}$ a $A_{4,0}$	
$c_{0,0}$	$A_{-2,0}$ a $A_{4,0}$	
$d_{0,0}$	$A_{0,-3}$ a $A_{0,3}$	$A_{0,-3}$ a $A_{0,4}$
$h_{0,0}$	$A_{0,-3}$ a $A_{0,4}$	
$n_{0,0}$	$A_{0,-2}$ a $A_{0,4}$	
$e_{0,0}$	$a_{0,-3}$ a $a_{0,3}$	$a_{0,-3}$ a $a_{0,4}$
$i_{0,0}$	$a_{0,-3}$ a $a_{0,4}$	
$p_{0,0}$	$a_{0,-2}$ a $a_{0,4}$	
$f_{0,0}$	$b_{0,-3}$ a $b_{0,3}$	$b_{0,-3}$ a $b_{0,4}$
$j_{0,0}$	$b_{0,-3}$ a $b_{0,4}$	
$q_{0,0}$	$b_{0,-2}$ a $b_{0,4}$	
$g_{0,0}$	$c_{0,-3}$ a $c_{0,3}$	$c_{0,-3}$ a $c_{0,4}$
$k_{0,0}$	$c_{0,-3}$ a $c_{0,4}$	
$r_{0,0}$	$c_{0,-2}$ a $c_{0,4}$	

5 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos com as arquiteturas de hardware desenvolvidas, bem como o processo de validação realizado com os dados de saída obtidos.

5.1 Resultados Obtidos

As arquiteturas de hardware desenvolvidas foram descritas em VHDL através da ferramenta Quartus II da Altera. Os resultados obtidos para os filtros de interpolação, através de simulação com a ferramenta, podem ser observados nas tabelas 5.1 e 5.2. A Tabela 5.1 apresenta os resultados referentes às versões do filtro tipo *Up/Down*, enquanto que os resultados das versões do filtro tipo *Middle* podem ser vistos na Tabela 5.2.

Tabela 5.1 – Resultados obtidos com as versões do filtro tipo *Up/Down*.

Arquitetura do Filtro *Up/Down*	**ALUTs Combinacionais**	**Total de registradores**	**Frequência Máxima (MHz)**
Sem otimizações	141	0	166,75
Combinacional	152	0	184,40
Pipeline e 2 estágios	152	48	295,25
Pipeline e 4 estágios	152	157	436,30
Dispositivo FPGA Altera Stratix III EP3SE50F484C2			

Tabela 5.2 – Resultados obtidos com as versões do filtro tipo *Middle*.

Arquitetura do Filtro *Middle*	**ALUTs Combinacionais**	**Total de registradores**	**Frequência Máxima (MHz)**
Sem otimizações	173	0	127,71
Combinacional	149	0	173,37
Pipeline e 2 estágios	149	46	274,42
Pipeline e 4 estágios	149	115	403,06
Dispositivo FPGA Altera Stratix III EP3SE50F484C2			

Todos os resultados foram obtidos considerando o dispositivo FPGA Stratix III, modelo EP3SE50F484C2. Com o objetivo de verificar os ganhos obtidos com as otimizações propostas, foram descritas versões sem qualquer tipo de otimização diretamente no ambiente do Quartus, utilizando macrofunções fornecidas pelo software. Estas versões para comparação foram descritas para cada tipo dos filtros, de forma combinacional.

5.2 Validação

O processo de validação das arquiteturas de hardware desenvolvidas foi feito utilizando a ferramenta ModelSim. A partir da versão mais recente disponível do software de referência do padrão HEVC (*HM-7.1rc1*), e também do ModelSim, foram gerados diferentes arquivos de saída na etapa de interpolação para uma mesma entrada no processo de codificação. Com estes arquivos de saída e o auxílio da ferramenta ModelSim foram realizadas comparações entre os resultados obtidos com a arquitetura e com o software de referência, permitindo a validação do hardware proposto.

5.3 Comentários sobre os Resultados

De posse dos resultados obtidos com as arquiteturas desenvolvidas para os filtros e uma análise do funcionamento da arquitetura completa para a etapa de interpolação é possível verificar o quanto as arquiteturas são satisfatórias para o

processamento de vídeos de alta definição em tempo real. Existem inúmeras resoluções de vídeo digital de alta definição atualmente. Na Tabela 5.3 estão representadas diversas informações importantes e necessárias para analisar a eficiência das arquiteturas desenvolvidas em relação a algumas destas resoluções. Considerando que é necessário o processamento de 30 qps para se obter a sensação de movimento contínuo, vídeos *Full* HD necessitam de uma taxa de processamento de 62,208 MAmostras/s, para amostras de luminância. Como na especificação do projeto foram considerados blocos de tamanho fixo 8x8, podemos concluir que devem ser processados 972 kBlocos/s. Sabendo que a arquitetura desenvolvida necessita de 52 ciclos para processar por completo um bloco 8x8, a frequência mínima é de 50,544 MHz, frequência inferior à obtida por qualquer uma das versões de qualquer um dos tipos de filtros.

Tabela 5.3 – Taxas de processamento necessárias para processamento em tempo real em diferentes resoluções de vídeo.

Resolução	***Full HD* (1920x1080)**	**QFHD (3840x2160)**	**UHDTV (7680x4320)**
Taxa de Processamento (qps)	30	30	30
Taxa de Processamento (MAmostras/s)	62,208	248,83	995,33
Número de blocos 8x8	32400	129600	518400
Taxa de Processamento (MBlocos/s) Considerando blocos 8x8.	0,972	3,888	15,552
Frequência Mínima (MHz) Considerando 52 ciclos para 1 bloco.	50,544	202,18	808,7

Baseado nos resultados de Frequência Máxima obtidos pelas diferentes versões dos filtros desenvolvidos e os dados da Tabela 5.3, é possível estimar o número de quadros por segundo que podem ser processados por cada uma das versões dos filtros otimizados, considerando diferentes resoluções. Foram consideradas para esta análise as resoluções *Full* HD, QFHD e UHDTV. A

estimativa de quadros por segundo das arquiteturas pode ser visualizada na Tabela 5.4. Todas as versões de filtros desenvolvidas podem processar vídeos *Full* HD. Para vídeos QFHD, apenas as versões combinacionais não se mostraram adequadas. Este tipo de verificação dos resultados é muito importante, pois permite tomar decisões de projeto de acordo com o objetivo da arquitetura. Por exemplo, se o alvo é o processamento de vídeos *Full* HD, a versão combinacional pode ser considerada a melhor solução, pois pode funcionar com uma frequência menor, reduzindo o consumo de energia, além de utilizar uma quantidade menor de hardware. Por outro lado, nenhuma arquitetura se mostrou eficaz para o processamento em tempo real de vídeos UHDTV. Sendo assim, se este fosse o alvo, seriam necessárias outras estratégias, além das otimizações realizadas, para atingir as taxas de processamento necessárias.

Tabela 5.4 – Estimativa de quadros por segundo utilizando as arquiteturas desenvolvidas.

Arquitetura Desenvolvida	**Taxa de processamento obtida (qps)**		
	***Full HD* (1920x1080)**	**QFHD (3840x2160)**	**UHDTV (7680x4320)**
Filtro *Up/Down* Combinacional	109	27	6
Filtro *Up/Down* com *Pipeline* e 2 estágios	175	43	10
Filtro *Up/Down* com *Pipeline* e 4 estágios	258	64	16
Filtro *Middle* Combinacional	102	25	6
Filtro *Middle* com *Pipeline* e 2 estágios	162	40	10
Filtro *Middle* com *Pipeline* e 4 estágios	239	59	14

6 CONCLUSÕES

Este trabalho apresentou o desenvolvimento em hardware da etapa de interpolação da FME do padrão HEVC responsável pela interpolação das amostras de luminância em posições inteiras. Com as arquiteturas desenvolvidas foi possível atingir resultados expressivos em termos de desempenho, uma vez que os filtros projetados e a arquitetura para interpolação em si se mostrou capaz de processar vídeos na resolução QFHD (3840 x 2160) em tempo real. Um estudo detalhado dos *drafts* e do software de referência do padrão HEVC também foi realizado, permitindo um embasamento teórico fundamental para o desenvolvimento de arquiteturas de hardware eficientes.

Como o padrão HEVC ainda se encontra em desenvolvimento, não foram encontradas na literatura científica arquiteturas de FME desenvolvidas, permitindo a realização de comparações. Com base em um estudo sobre o estado da arte das arquiteturas de FME segundo o padrão H.264/AVC, foi possível analisar o impacto da utilização de algumas estratégias durante a especificação da arquitetura. Para permitir uma contribuição ainda mais relevante para a FME do padrão HEVC, uma série de avaliações serão realizadas, utilizando o software de referência do HEVC, as quais permitirão saber, entre outras informações, o impacto da ME e da FME nos resultados de compressão obtidos pelo padrão, os tamanhos de blocos PU mais selecionados durante o processo de codificação e as perdas/ganhos na qualidade da imagem e no *bit-rate*, limitando ou mesmo fixando os tamanhos dos blocos.

Além das avaliações, será desenvolvida em trabalhos futuros, a etapa de busca e comparação de blocos fracionários, obtendo-se após a integração destas etapas, uma arquitetura completa de FME para o padrão HEVC.

REFERÊNCIAS

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas a Compressão de Vídeo Segundo o Padrão H.264/AVC.** 2007. 172f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

ALSHINA, E., ALSHIN, A. e PARK, JH. **CE3: 7 Taps Interpolation Filters for Quarter Pel Position MC from Samsung and Motorola Mobility (JCTVC-G778)**, [S.l.], 2011.

BOSSEN, F. **Common Test Conditions and Software Reference Configurations (JCTVC-I1100)**, [S.l.], 2012.

BROSS, B., HAN, W., OHM, J., SULLIVAN, G. e WIEGAND, T. **High Efficiency Video Coding text specification draft 7 (JCTVC-I1003)**, [S.l.], 2012.

CARVALHO, M. G. **Implementação Hardware/Software da Estimação de Movimento Segundo o Padrão H.264.** 2007. 102f. Dissertação (Mestrado em Sistemas e Computação) – Departamento de Informática e Matemática Aplicada, UFRN, Natal.

CORRÊA, M. M. e SCHOENKNECHT, M. T. A H.264/AVC quarter-pixel motion estimation refinement architecture targeting high resolution videos. In: VII Southern Conference on Programmable Logic, SPL**,** 2011. **Proceedings...** Cordoba: IEEE, 2011a. p. 131-136.

CORRÊA, M. M., SCHOENKNECHT, M. T. e DORNELLES, R. S. A high-throughput hardware architecture for the H.264/AVC half-pixel motion estimation targeting high-definition videos. **International Journal of Reconfigurable Computing**, [S.l.], v. 2011, 2011b.

GHANBARI, M. **Standard Codecs: Image Compression to Advanced Video Coding.** United Kingdom: The Institution of Electrical Engineers, 2003.

GONZALEZ, R.; WOODS, R. **Processamento de Imagens Digitais.** São Paulo: Edgard Blücher, 2003.

HEVC Reference Software. Disponível em: <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/> Acesso em: junho 2012.

ITU-T; ISO/IEC. **Joint Call for Proposals on Video Compression Technology (VCEG-AM91)**, [S.l.], 2010.

JCT-VC. **Encoder-side Description of Test Model under Consideration (JCTVC-B204)**, [S.l.], 2010.

KAO, C., KUO, H. e LIN, Y. High performance fractional motion estimation and mode decision for the H.264/AVC. In: IEEE International Conference on Multimedia and Expo, ICME, 2006. **Proceedings...** [S.l.]: IEEE, 2006. p. 1241-1244.

LI, B., SULLIVAN, G. e XU, J. **Comparison of Compression Performance of HEVC Draft 6 with AVC High Profile (JCTVC-I0409)**, [S.l.], 2012.

McCANN, K., BROSS, B., KIM, I., SUGIMOTO, K. e HAN, W. **HM6: High Efficiency Video Coding Test Model 6 Encoder Description (JCTVC-H1002)**, [S.l.], 2012.

MIANO, J. **Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP.** Reading, MA: Addison Wesley, 1999.

OKTEM, S., e HAMZAOGLU, I. An efficient hardware architecture for quarter-pixel accurate H.264 motion estimation. In: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD '07**,** 2007. **Proceedings...** Lubeck: IEEE, 2007. p. 444–447.

PORTO, M. **Desenvolvimento Algorítmico e Arquitetural para a Estimação de Movimento na Compressão de Vídeo de Alta Definição.** 2012. 166f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

PURI, A.; et al. Video Coding Using the H.264/MPEG-4 AVC Compression Standard. **Elsevier Signal Processing: Image Communication.** [S.l.], n. 19, p.793–849, 2004.

RICHARDSON, I. **Video Codec Design : Developing Image and Video Compression Systems.** Chichester: John Wiley and Sons, 2002.

RICHARDSON, I. **H.264 and MPEG-4 Video Compression : Video Coding for Next-Generation Multimedia.** Chichester: John Wiley and Sons, 2003.

SALOMON, D. **A Concise Introduction to Data Compression.** New York: Springer, 2008.

SHI, Y.; SUN, H. **Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards.** Boca Raton: CRC Press, 1999.

SULLIVAN, G. e WIEGAND, T. **Draft requirements for next-generation video coding project (VCEG-AL96)**, [S.l.], 2009.

YALCIN, S. e HAMZAOGLU, I. A high performance hardware architecture for half-pixel accurate H.264 motion estimation. In: 14th International Conference on Very Large Scale Integration and System-on-Chip, VLSI-SoC '06, 2006. **Proceedings...** Nice: IEEE, 2006. p. 63–67.