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

Trabalho Individual I

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, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Denis T. Franco

PELOTAS
2012

RESUMO

AFONSO, Vladimir. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC**. 2012. 53 f. Trabalho Individual (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

Este trabalho apresenta arquiteturas de hardware desenvolvidas para a implementação dos filtros de interpolação da Estimação de Movimento Fracionária (FME – *Fractional Motion Estimation*) do padrão de codificação de vídeo *High Efficiency Video Coding* (HEVC). Foi realizado um estudo a respeito do estado da arte em termos de arquiteturas para a implementação da FME dos padrões de codificação de vídeo H.264/AVC (*Advanced Video Coding*) e HEVC. Com base em comparações entre os trabalhos relacionados foi possível analisar o impacto da utilização de algumas estratégias durante a especificação da arquitetura, bem como perceber algumas diferenças entre a FME desses dois padrões de codificação de vídeo. As arquiteturas de hardware desenvolvidas fazem parte da etapa de interpolação da FME, que será completamente desenvolvida em trabalhos futuros. Além da etapa de interpolação, será desenvolvida uma etapa de busca de ¼ de pixel para, após a integração destas etapas, se obter uma arquitetura completa de FME para o padrão HEVC.

Palavras-chave: Vídeo Digital, Padrão HEVC, Estimação de Movimento, Filtros de Interpolação.

ABSTRACT

AFONSO, Vladimir. **Desenvolvimento de Arquiteturas para Estimação de Movimento Fracionária Segundo o Padrão HEVC**. 2012. 53 f. Trabalho Individual (Mestrado em Ciência da Computação). Universidade Federal de Pelotas, Pelotas.

This work presents hardware architectures developed for the interpolation filters necessary for the Fractional Motion Estimation (FME) step defined in the High Efficiency Video Coding (HEVC) standard. We conducted a study about the state of the art in terms of FME hardware architectures according to video coding standards H.264/AVC (Advanced Video Coding) and HEVC. Based on comparisons of available works, it has been possible to analyze the impact of some strategies in architecture specification, and evaluate the differences in FME between these two video coding standards. The proposed interpolation filters are part of the FME interpolation stage, which will be fully developed in future works. In addition to the interpolation stage, the quarter-pixel search stage will also be developed. After integrating these stages, a complete architecture for FME will be available, according to HEVC standard.

Keywords: Digital Video, HEVC Standard, Motion Estimation, Interpolation Filters.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

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*
HM3	*High Efficiency Video Coding Test Model 3 Encoder Description*
HSI	*Hue, Saturation, Intensity*
JCT-VC	*Joint Collaborative Team on Video Coding*
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	Quantização
QFHD	*Quad Full* HD
QHDTV	*Quad High Definition Digital Television*
QUXGA	*Quad Ultra eXtended Graphics Array*

RGB	*Red, Green, Blue*
SAD	*Sum of Absolute Differences*
T	Transformada
tco	*clock-to-output delay*
tpd	*propagation delay*
TU	*Transform Unit*
TV	Televisão
UFPel	Universidade Federal de Pelotas
VGA	*Video Graphics Adapter*
VHDL	VHSIC *Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
WD3	*Working Draft 3 of High-Efficiency Video Coding*
Y	*Luminance*
YCbCr	*Luminance, Chrominance Blue, Chrominance Red*

SUMÁRIO

1 INTRODUÇÃO

Existem atualmente diversas aplicações envolvendo vídeo digital, como aparelhos de DVD, TV digital e dispositivos móveis, entre outras. Devido à elevada complexidade computacional que é demandada para o processamento de vídeos digitais em tempo real, se faz necessário o desenvolvimento de circuitos integrados específicos, já que de outra forma, com soluções em software, seriam necessários processadores de propósito geral de alto desempenho, o que acarretaria em um aumento de consumo energético e inviabilizaria a execução de vídeos digitais em alguns dispositivos portáteis. Além disso, como o processamento de vídeo digital envolve uma quantidade muito grande de dados, se torna inviável a utilização deste tipo de vídeo em aplicações de tempo real, sem o uso de técnicas de compressão. O bloco codificador, responsável pela compressão nos vídeos digitais, apresenta diversas etapas, cada uma delas com o objetivo de explorar algum tipo de redundância dos dados existentes nos vídeos. 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 existentes, o padrão H.264/AVC (*Advanced Video Coding*) é o mais eficiente em termos de desempenho na compressão de dados, ao custo de uma maior complexidade computacional (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, todos os esforços para aumentar esta eficiência têm sido insuficientes, uma vez que existe a necessidade de atender a determinadas expectativas de desempenho. Estas expectativas estão relacionadas ao aumento sistemático das resoluções de

vídeo, principalmente no que se refere a aplicações destinadas aos dispositivos com recursos computacionais ou energéticos limitados, como é o caso dos dispositivos portáteis.

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. Algumas destas características estão relacionadas à predição inter-quadros, mais especificamente ao bloco de estimação de movimento (ME – *Motion Estimation*) e serão abordadas neste trabalho.

Considerando um codificador de vídeo genérico, a etapa de ME visa à identificação e redução ou até a eliminação das redundâncias do tipo temporal. Uma técnica importante que pode ser empregada na ME, é a técnica conhecida como estimação de movimento fracionária (FME – *Fractional Motion Estimation*) ou ME com precisão de sub-pixel, a qual possibilita uma maior eficiência na codificação. Este trabalho apresenta arquiteturas de hardware desenvolvidas para a implementação dos filtros de interpolação da FME do padrão HEVC.

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 é extremamente motivador para os pesquisadores da área de codificação de vídeo, uma vez que existem inúmeras oportunidades de contribuição científica neste momento. Além disso, os dispositivos eletrônicos são cada vez mais exigidos, pois sistematicamente estes equipamentos têm seus recursos ampliados, como por exemplo, a execução de vídeos digitais 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. De outra forma, aplicações como a execução de vídeos digitais em dispositivos móveis

não seria possível.

1.2 Objetivos

Este trabalho tem como objetivo principal o desenvolvimento de arquiteturas de hardware de alto desempenho para a FME do padrão de codificação de vídeo HEVC. Também é objetivo deste trabalho a realização de um estudo sobre o estado da arte em termos de arquiteturas para FME dos padrões H.264/AVC e HEVC, o qual possibilite um embasamento teórico para o desenvolvimento de arquiteturas de hardware eficientes. A partir dos resultados obtidos com este trabalho, será desenvolvido o hardware completo da interpolação de ¼ de pixel para amostras de luminância do padrão HEVC. A arquitetura para interpolação que será desenvolvida será integrada, futuramente, a uma etapa de busca de ¼ de pixel para, após a integração destas etapas, ser obtida 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 vídeo digital, 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. Os conceitos apresentados neste capítulo estão estreitamente relacionados com a arquitetura de hardware desenvolvida neste trabalho, a qual será apresentada nos próximos capítulos.

2.1 Conceitos Básicos de Vídeo Digital

Um vídeo digital consiste em uma sequência de imagens independentes, captadas com um determinado intervalo de tempo entre as mesmas. Estas imagens independentes, que formam uma sequência de vídeo, são chamadas de quadros (*frames)* e são compostas por pixels. Os pixels são os pontos que formam a imagem e são representados através de três amostras, que correspondem às componentes de brilho ou cor, conforme o sistema utilizado para representação das cores. Os quadros de uma sequência de vídeo, compostos pelos pixels, também podem ser divididos em blocos pelos codificadores. Estes blocos podem apresentar diferentes tamanhos de acordo com o padrão de codificação e, inclusive, em um mesmo padrão o tamanho do bloco pode ser variável, permitindo a aplicação mais eficiente das técnicas de compressão. 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 em uma matriz de pontos chamada de resolução do vídeo, que tem o objetivo de formar os quadros da sequência de vídeo. A qualidade da imagem será melhor, quanto maior for a resolução do vídeo, uma vez que mais pixels serão usados na representação dos quadros. Já a amostragem temporal está relacionada aos intervalos de tempo entre

a captura das imagens em uma sequência de vídeo. A percepção da movimentação no vídeo será melhor, quanto maior for a taxa de amostragem temporal. Para que seja obtida uma sensação de tempo real, a taxa de captura das imagens deve ser de, no mínimo, 24 a 30 imagens a cada segundo (GONZALEZ, 2003), sendo que algumas aplicações usam até 60 imagens por segundo.

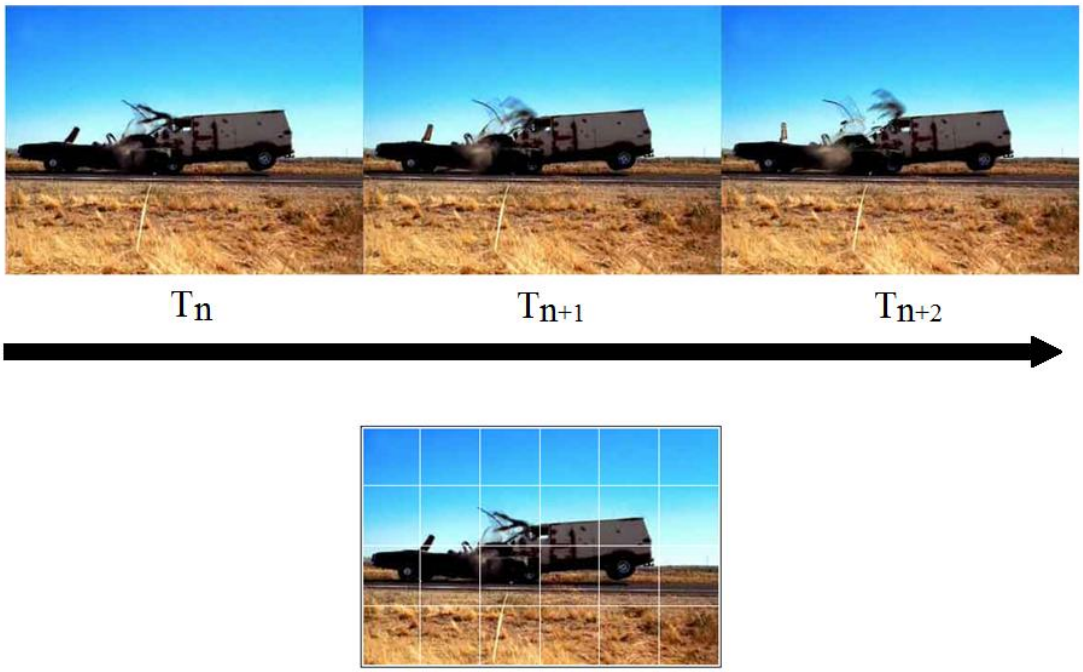

Figura 2.1 – Sequência de quadros e blocos de um vídeo digital.

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 sem o uso de técnicas de compressão em aplicações de tempo real. 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 perdas mínimas 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. Apesar de compressores sem perdas reduzirem o tamanho dos arquivos, este método de compressão não funciona bem com imagens e vídeos digitais, obtendo taxas de compressão reduzidas. 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. Para tal, o método de compressão com perdas é o mais adequado (RICHARDSON, 2003). Utilizando-se técnicas de compressão de vídeo com perdas, o padrão H.264/AVC atinge taxas de compressão de cinquenta vezes, mantendo a qualidade visual da imagem muito próxima a do vídeo original (AGOSTINI, 2007).

2.3 Espaço de Cores e Subamostragem de Cores

Um espaço de cores consiste em um sistema para representação de cores. (SHI, 1999). O espaço de cores YCbCr é composto por três componentes: luminância (Y), que define o brilho; crominância azul (Cb); e crominância vermelha (Cr) (MIANO, 1999). Este sistema é bastante utilizado na compressão de vídeos digitais (RICHARDSON, 2002), pois as informações de brilho podem ser tratadas separadamente das informações de cor nos codificadores, o que não acontece em outros sistemas como o RGB (*Red*, *Green* e *Blue*). Esta característica do sistema YCbCr é importante na medida que o sistema visual humano é mais sensível às informações de brilho do que de cor (GONZALEZ, 2003). Dessa forma, os codificadores de vídeo diminuem a taxa de amostragem espacial das informações de crominância em relação às informações de luminância, para aumentar a eficiência da codificação (RICHARDSON, 2002). Esta técnica é chamada de subamostragem de cores.

A subamostragem de cores pode relacionar componentes de crominância e luminância de várias formas, sendo que os formatos 4:4:4, 4:2:2 e 4:2:0 são os mais comuns. O formato 4:4:4 considera que para cada quatro amostras de Y, existem quatro amostras de Cb e quatro amostras de Cr. Já no formato 4:2:2, para cada quatro amostras de Y, existem apenas duas amostras de Cb e duas amostras de Cr. O formato 4:2:0, por sua vez, considera que para cada quatro amostras de Y, existe apenas uma amostra de Cb e uma amostra de Cr. Considerando o formato 4:4:4, não existe perda na qualidade da imagem, pois as taxas de amostragem espacial de luminância e crominância são iguais. Por outro lado, apesar de ocasionar perda na

qualidade da imagem, a utilização do formato 4:2:0 reduz o tamanho total do vídeo à metade só pela subamostragem de cores. O padrão HEVC, por padrão, utiliza o formato de subamostragem 4:2:0. Contudo, os formatos 4:4:4 e 4:2:2 também podem ser utilizados. Inclusive, existe a opção de não utilizar amostras de crominância (monocromático) (BROSS *et al.,* 2012). A Figura 2.2 mostra exemplos de vários formatos de subamostragem de crominância.

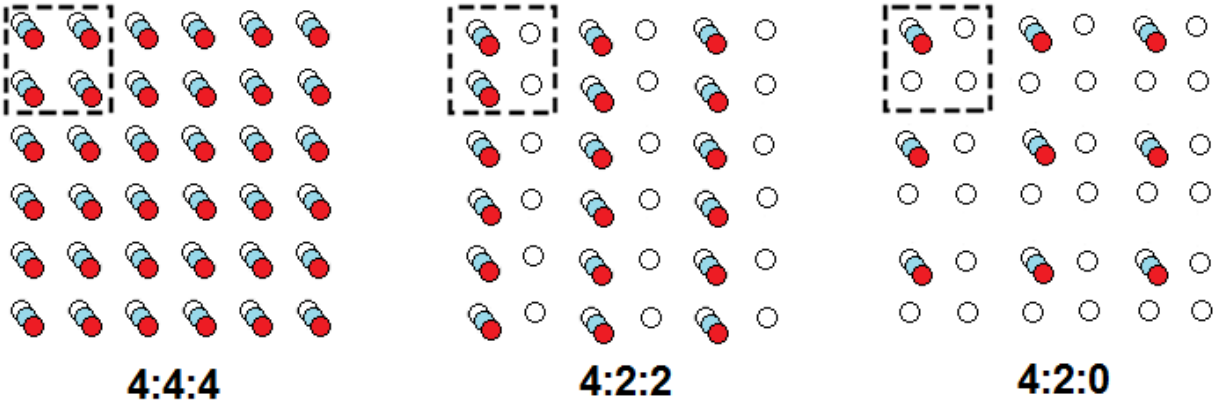

Figura 2.2 – Formatos de subamostragem de crominância.

2.4 Redundância de Dados na Representação de Vídeos Digitais

Em geral, vídeos digitais apresentam redundância nas suas informações e estas redundâncias podem ser exploradas amplamente pelos codificadores. Neste contexto, destacam-se três tipos de redundância de dados, as redundâncias espacial, temporal e entrópica.

A redundância espacial, também conhecida como redundância intra-quadro (*intraframe*), está relacionada com a tendência que pixels vizinhos de um mesmo quadro têm de apresentar valores semelhantes. Já a redundância temporal, também conhecida como redundância inter-quadros (*interframe*), corresponde à grande semelhança existente entre quadros temporalmente próximos de uma sequência de vídeo (GHANBARI, 2003). A redundância entrópica está relacionada com o número de bits usados para representar cada símbolo codificado. Os codificadores que exploram a redundância entrópica visam à transmissão das informações utilizando o menor número de bits possível por símbolo codificado, através de técnicas de compressão sem perdas (SHI, 1999).

2.5 Codificadores de Vídeo Digital

Na Figura 2.3 pode ser observado o diagrama em blocos de um modelo de codificador de vídeo de acordo com a maioria dos padrões de compressão de vídeo atuais (AGOSTINI, 2007), incluindo os padrões H.264/AVC e HEVC.

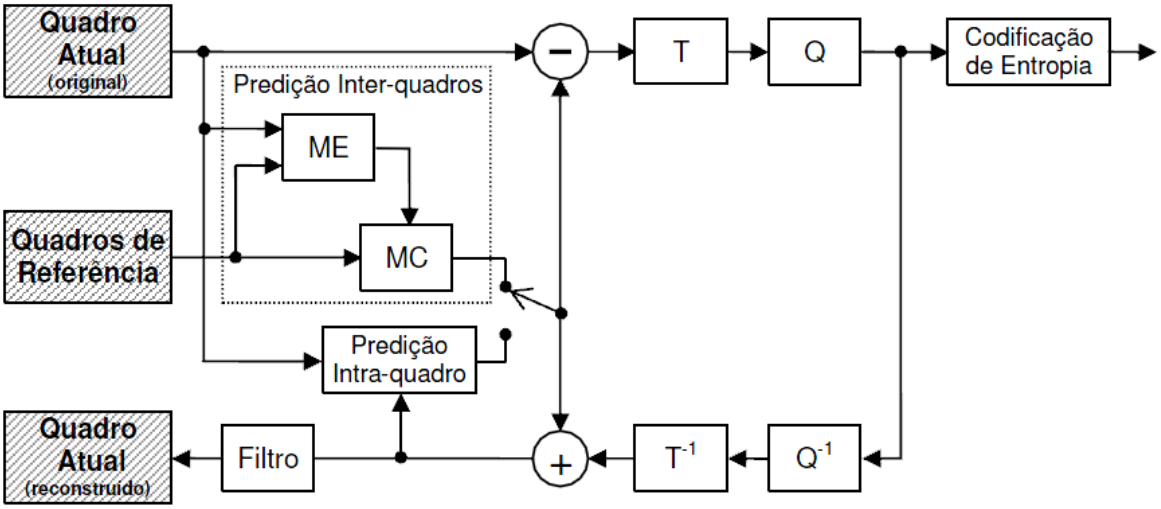

Figura 2.3 – Modelo genérico de codificador de vídeo.

Fonte: (AGOSTINI, 2007, p.34).

Como pode ser observado na Figura 2.3, 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 possa ser eliminada, ou pelo menos reduzida. Isto é feito através de uma comparação dos blocos de vídeo do quadro atual, que estão sendo codificados, com blocos de quadros processados anteriormente, chamados quadros de referência. Um vetor de movimento é gerado indicando a localização do bloco processado mais semelhante ao bloco do quadro atual que está sendo codificado, permitindo que seja enviado apenas o vetor e a informação referente à diferença entre os blocos. 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. Para aplicação da técnica, são necessárias apenas as informações do quadro atual. Este quadro é dividido em blocos, permitindo a comparação do bloco que está sendo codificado com outros blocos do quadro.

A chave seletora representada na Figura 2.3, corresponde a uma etapa de controle no codificador, a qual é responsável por escolher entre as predições inter-quadros ou intra-quadro, a que será utilizada em cada bloco, de acordo com as características do vídeo a ser codificado. Em seguida, uma operação de subtração entre os valores do quadro atual e do quadro reconstruído é realizada, permitindo a obtenção de um resíduo.

Após a obtenção do resíduo, existe a etapa chamada de transformada (T), cujo objetivo é transformar as informações do domínio espacial para o domínio das frequências. Esta transformação é necessária para que a operação chamada de quantização (Q) possa ser utilizada. O princípio de funcionamento da quantização está baseado na eliminação das frequências menos relevantes ao sistema visual humano. A quantização gera perdas no processo de codificação. Porém, em geral, estas perdas podem ser controladas e interferem de forma pouco significativa na qualidade da imagem.

Em seguida, 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 o uso de codificação de comprimento de palavra variável, sendo que diversos tipos de algoritmos podem ser utilizados.

Ainda se fazem necessárias, no codificador, as etapas de transformada e quantização inversas para geração do quadro atual reconstruído. Este quadro é obtido através da soma do resíduo com a saída da transformada inversa e é muito importante, pois é utilizado como quadro de referência para a predição inter-quadros ou intra-quadro, na codificação dos blocos ou quadro subsequentes. Estas operações existem para permitir que codificadores e decodificadores utilizem as mesmas referências, já que a quantização gera perdas irreversíveis no processo de codificação.

3 PADRÃO HEVC DE CODIFICAÇÃO DE VÍDEO

Neste capítulo é apresentada uma visão geral do padrão de codificação de vídeo HEVC. Alguns assuntos importantes são introduzidos, 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 e Visão Geral

O padrão HEVC está em desenvolvimento desde Janeiro de 2010 através da colaboração de especialistas, 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 de imagem. 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.

Uma das principais inovações do padrão HEVC está no novo esquema de compressão de vídeo baseado em uma hierarquia altamente 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 o codificador todo pode explorar o tamanho de bloco flexível.

A estrutura geral de codificação do Padrão HEVC permite o particionamento da imagem em *treeblocks*. Uma *treeblock* é composta por um bloco NxN de amostras de luminância juntamente com outros dois blocos correspondentes às amostras de crominância. Os tamanhos dos blocos de crominância dependem de qual é a amostragem de cores adotada. O conceito de *treeblock* 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 *treeblock* é 64x64 amostras de luminância, o que corresponde também ao maior tamanho de CU permitido. A Figura 3.1 apresenta um quadro dividido em 99 *treeblocks*.

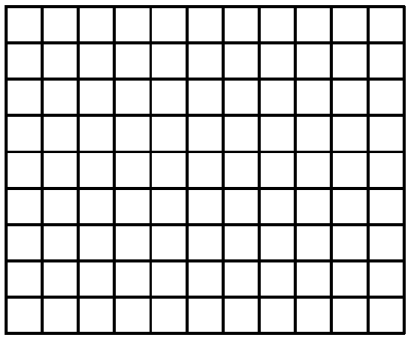

Figura 3.1 – Exemplo de quadro dividido em *treeblocks*.
Fonte: (McCANN *et al.,* 2011, p. 12).

Uma *treeblock* é composta por uma ou mais CUs. As CUs são utilizadas para predição inter-quadros/intra-quadro, são sempre quadradas e podem assumir tamanhos a partir de 8x8 amostras de luminância até o tamanho da *treeblock*. O conceito de CU permite a divisão recursiva em quatro blocos de tamanho igual, partindo do *treeblock*. 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 3.2 e 3.3 mostram uma *treeblock* dividida em CUs e a árvore quadrática que corresponde a esta divisão, respectivamente.

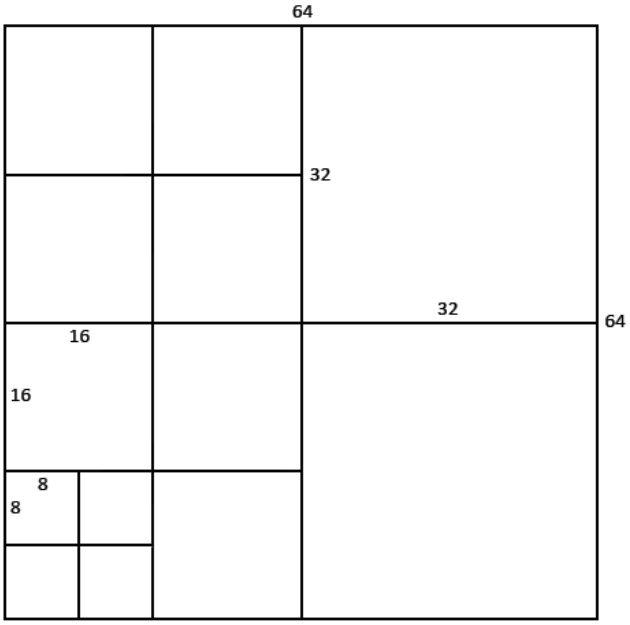

Figura 3.2 - Exemplo de uma *treeblock* dividida em CUs.

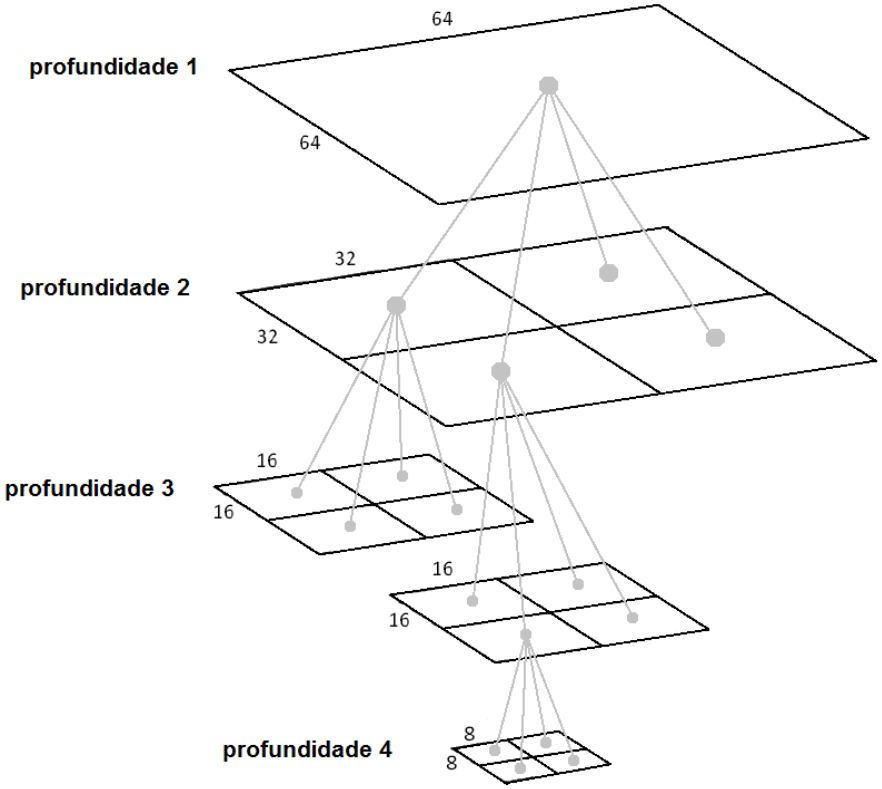

Figura 3.3 - Exemplo de árvore quadrática de uma *treeblock*.

Na Figura 3.3 se pode observar uma árvore quadrática de uma *treeblock*. Primeiramente a *treeblock* 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.

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. Cada CU pode conter uma ou mais PUs, sendo que o tipo de divisão NxN só é permitido quando o tamanho correspondente da CU é maior do que o menor tamanho de CU permitido. A Figura 3.4 mostra os diferentes tipos de divisão de uma CU em PUs.

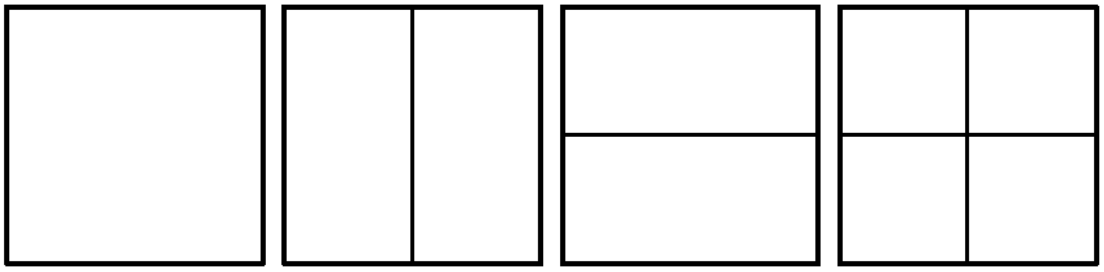

Figura 3.4 – Quatro tipos de divisão de uma CU em PUs.
Fonte: (McCANN *et al.,* 2011, p. 13).

A TU é a unidade básica para os processos de transformada e quantização e é sempre formada por quadrados. 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 3.5.

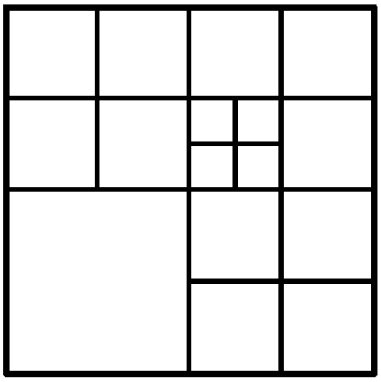

Figura 3.5 – Exemplo de CU 32x32 dividida em TUs.
Fonte: (McCANN *et al.,* 2011, p. 14).

3.2 Estimação de Movimento

Considerando o codificador de vídeo genérico apresentado no capítulo anterior, a ME é a etapa que visa à identificação e redução ou até a eliminação das redundâncias do tipo temporal. A redundância temporal consiste na 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 a partir de quadros anteriormente codificados que são chamados de quadros de referência. O quadro atual é dividido em macroblocos e estes são comparados aos macroblocos do quadro de referência, considerando uma determinada área de busca (área de pesquisa). No caso de semelhança entre os macroblocos destes dois tipos de quadros, é gerado um vetor de movimento que irá permitir a localização do macrobloco que gerou o melhor casamento. Esta visão geral do processo de ME pode ser observada na Figura 3.6. Na etapa seguinte, a MC, é construído o quadro estimado que é baseado nas informações dos vetores de movimento gerados pela ME. Juntas, a ME e a MC compõem a operação que é chamada de inter-quadros nos codificadores. Dentro do codificador, a operação inter-quadros é a responsável pelos melhores resultados em termos de taxas de

compressão, apesar de também ser a operação que apresenta maior complexidade computacional.

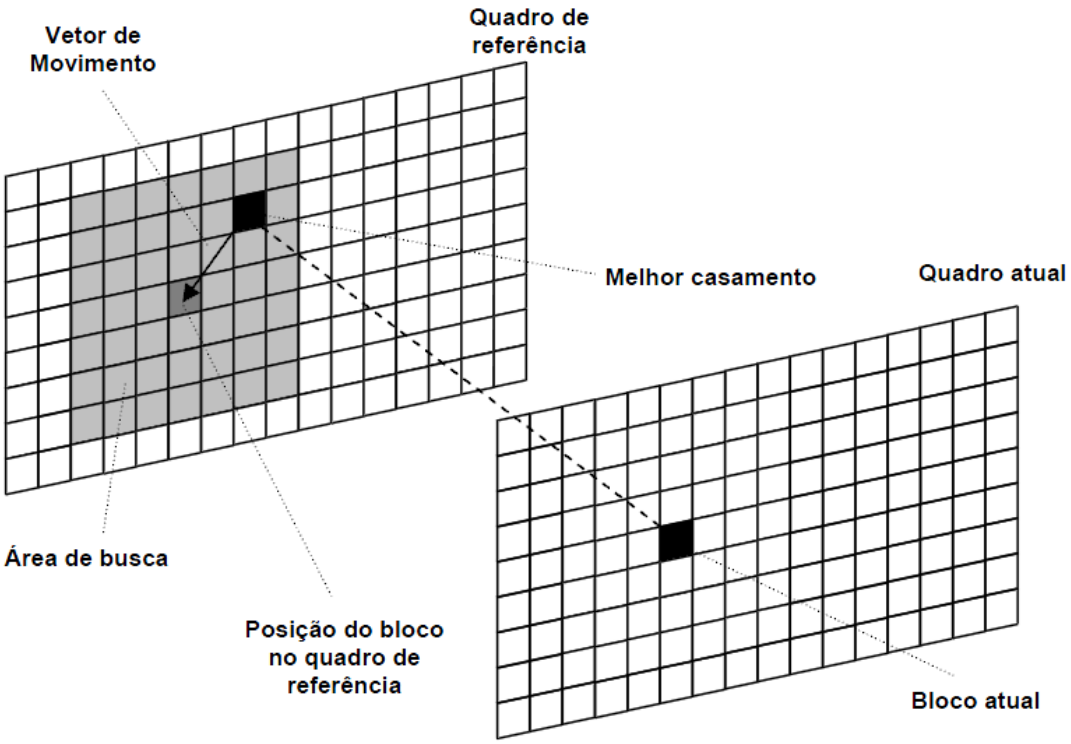

Figura 3.6 – Elementos da Estimação de Movimento
Fonte: (PORTO, 2011, p.31).

3.3 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 3.7. O padrão H.264/AVC utiliza FME com o objetivo de permitir casamentos bons entre os macroblocos 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, principalmente com relação aos filtros de interpolação utilizados para posições fracionárias de ¼ de pixel.

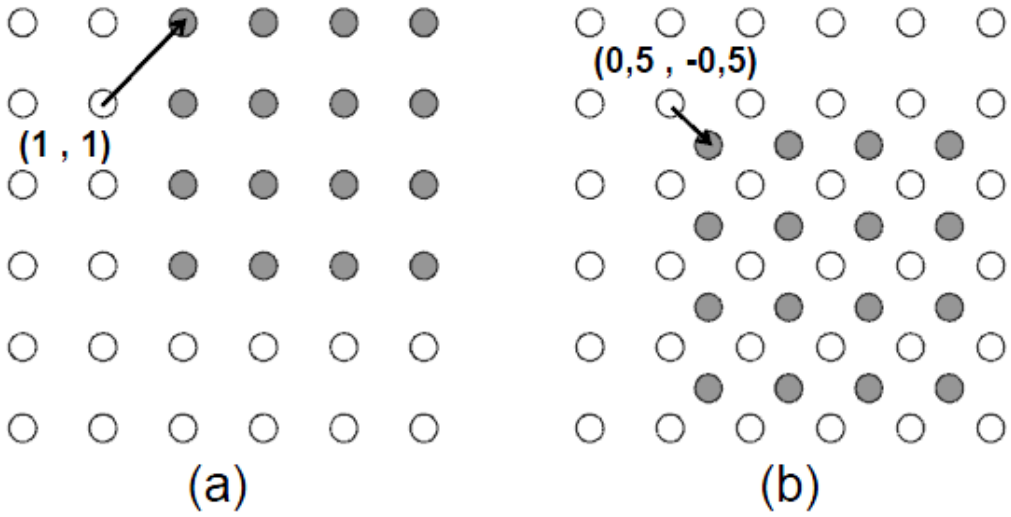

Figura 3.7 – Vetores de Movimento: (a) ME em posições inteiras e (b) FME.
Fonte: (CORRÊA; SCHOENKNECHT, 2011a).

Como mencionado anteriormente, a técnica de FME permite uma maior eficiência na codificação, uma vez que possibilita bons casamentos entre os macroblocos 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.

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, cada vez que a precisão é aumentada, 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. Este processo pode ser observado na Figura 3.8.

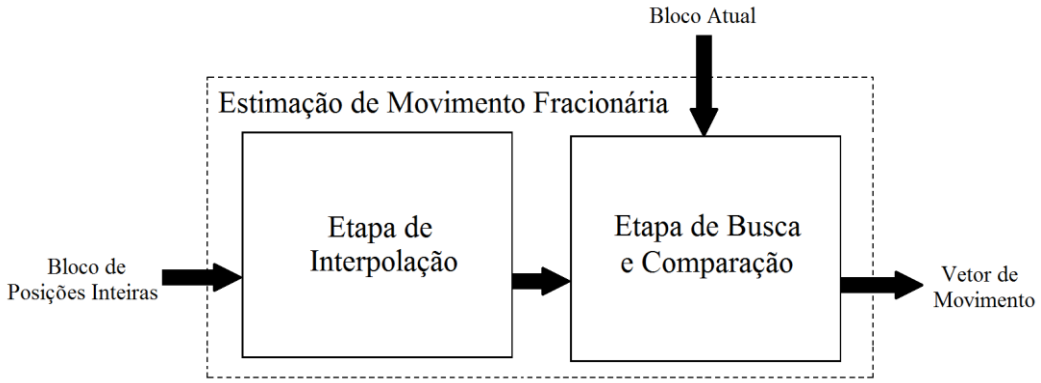

Figura 3.8 – Processo de FME.

3.4 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.,* 2011b). 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 posições de precisão de ½ pixel são geradas a partir das posições inteiras das amostras de luminância através de um filtro FIR (*Finite Impulse Response)* de 6 *taps*. Ainda nesta etapa, após a interpolação, é realizada a busca por posições fracionárias de ½ pixel que permitam um casamento melhor do que o obtido através da busca por posições inteiras. Caso seja encontrado um melhor casamento em precisão de ½ pixel, é realizada uma segunda etapa. A segunda etapa consiste em encontrar as 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).

3.5 Estado da Arte das Arquiteturas para FME

Para permitir o estudo das arquiteturas de FME dos padrões H.264/AVC e HEVC foram pesquisados na literatura científica alguns trabalhos relacionados.

Porém, não foram encontrados trabalhos que envolvam o desenvolvimento de arquiteturas para a FME com precisão de sub-pixel do padrão HEVC. Desta forma, foram considerados para comparação alguns trabalhos que apresentam projetos de arquiteturas para FME do padrão H.264/AVC.

Corrêa (2011b) apresenta uma arquitetura para FME com precisão de ½ pixel para o padrão H.264/AVC, sendo capaz de processar vídeos *Full HD* (1920×1080), mas com abordagem focada em blocos de tamanho 8x8. A arquitetura proposta é dividida em duas partes: uma para a interpolação e outra para a busca. A unidade de interpolação é constituída basicamente de unidades de processamento e *buffers*, sendo que as unidades de processamento nada mais são do que a implementação dos filtros FIR de 6 *taps* citados anteriormente. Após a interpolação das amostras de luminância com precisão de ½ pixel a partir de amostras inteiras, é realizada a busca, cujo hardware consiste basicamente de uma etapa para calcular a soma das diferenças absolutas (SAD – *Sum of Absolute Differences*) das posições com precisão de sub-pixel, um comparador de SAD e *buffers*.

A abordagem de Corrêa (2011a) para implementar a arquitetura de FME com precisão de ¼ de pixel é similar. Basicamente, as diferenças estão no processo de interpolação, no qual as unidades de processamento para interpolação de ¼ de pixel utilizam outra estratégia conforme descrito anteriormente. Para adotar a decisão de utilizar tamanho de bloco fixo 8x8, Corrêa (2011a) 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. Com base nos resultados, observou-se pequenas quedas na qualidade de imagem em detrimento de redução de hardware. A arquitetura permite o processamento em tempo real de vídeos QHDTV (3840x2048).

Kao (2006) propõe uma arquitetura para FME com suporte a tamanho de bloco variável e ao modo de decisão para o padrão H.264/AVC. Esta proposta emprega um modelo matemático para estimar SADs em posições de ¼ de pixel da FME 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. A arquitetura de hardware dá suporte a tamanho de bloco variável e é capaz de processar vídeos QUXGA (3200x2400) em tempo real.

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, o que se torna importante no foco desse trabalho, o qual é voltado para aplicações portáteis. A arquitetura dá suporte a tamanho de bloco variável, mas devido à taxa de processamento obtida, este trabalho consegue processar em tempo real vídeos VGA (640x480).

Yalcin (2006) propõe uma arquitetura de hardware para a FME com precisão de ½ pixel do padrão H.264/AVC com suporte a tamanho de bloco variável na qual é realizada a interpolação e busca de ½ pixel para cada tamanho de bloco. A arquitetura proposta foi desenvolvida com o objetivo de atender a aplicações portáteis. Este trabalho apresenta uma taxa de processamento adequada para processar vídeos HDTV (1280x720).

Como pode ser observado, existem diversos trabalhos na literatura científica relacionados à FME do padrão H.264/AVC. Alguns trabalhos abordam o desenvolvimento de arquiteturas para precisão de ¼ de pixel e outros para ½ pixel. Os resultados obtidos 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. Estas diferenças serão abordadas na seção seguinte.

3.5.1 Comparação entre os Trabalhos Relacionados

A Tabela 3.1 apresenta alguns resultados obtidos com as arquiteturas para FME com precisão de ½ pixel do padrão H.264/AVC desenvolvidas por Yalcin (2006) e Corrêa (2011b). A síntese de ambos os projetos utilizou o dispositivo FPGA Xilinx Virtex II. Considerando os dados apresentados para arquiteturas de ½ pixel, é possível verificar que a abordagem de Corrêa (2011b), a qual considera tamanho de bloco fixo 8x8, contribui 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. Já, no caso da abordagem de Yalcin (2006), que suporta tamanho de bloco variável, não foi apresentado o modo de decisão, o que não permite uma análise mais precisa com relação ao trabalho de Corrêa (2011b).

As frequências máximas de operação dos trabalhos de Yalcin (2006) e Corrêa (2011b), considerando os mesmos dispositivos FPGA, são semelhantes, atingindo 85 MHz e 91MHz, respectivamente. Contudo, o trabalho de Yalcin (2006) requer um número de ciclos elevado para processar um macrobloco 16x16 quando comparado com Corrêa (2011b), já que a abordagem de Corrêa (2011b) necessita de 148 ciclos e a proposta de Yalcin (2006) requer 720 ciclos. Além disso, considerando uma síntese em dispositivo FPGA Xilinx Virtex4, a frequência obtida por Corrêa (2011b) chegaria a 140MHz. Com esta frequência, o trabalho de Corrêa (2011b) seria adequado para o processamento de vídeos QHDTV (3840×2048) em tempo real. Considerando que a síntese de ambos os projetos utilize o dispositivo FPGA Xilinx Virtex II, o trabalho de Yalcin (2006) é capaz de processar vídeos HDTV (1280x720) em tempo real enquanto que a abordagem de Corrêa (2011b) pode processar em tempo real vídeos *Full HD* (1920×1080).

Tabela 3.1 – Resultados obtidos em diferentes arquiteturas de FME com precisão de ½ pixel para o padrão H.264/AVC.

Artigo Científico	**Yalcin (2006)**	**Corrêa (2011b)**
Suporte para Tamanho de Bloco Variável	Sim	Não
Frequência	85 MHz	91 MHz
Ciclos para processar um macrobloco 16x16	720	148
Resolução obtida	HDTV (1280x720)	*Full HD* (1920×1080)

Dispositivo FPGA: Xilinx Virtex II

A Tabela 3.2 mostra alguns resultados obtidos por três trabalhos que propõem arquiteturas para a FME com precisão de ¼ de pixel do padrão H.264/AVC. Os trabalhos apresentados foram sintetizados para diferentes tecnologias. Quanto a Frequência Máxima de operação, Corrêa (2011a) obteve 245MHz enquanto Oktem (2007) e Kao (2006) obtiveram 60MHz e 100MHz, respectivamente. Assim como

nas arquiteturas de ½ pixel, nas comparações para arquiteturas de ¼ de pixel é possível observar que a proposta de Corrêa (2011a) considera blocos de tamanho fixo 8x8, ao contrário das outras duas abordagens de Oktem (2007) e Kao (2006) que dão suporte a tamanho de bloco variável. Contudo, somente Kao (2006) apresenta o modo de decisão em sua abordagem. Como mencionado anteriormente, a estratégia de Corrêa (2011a) de utilizar blocos de tamanho fixo reduz a complexidade computacional e têm impactos na diminuição da qualidade da imagem e do custo de hardware. Porém, conforme as avaliações de qualidade de imagem, é possível observar que a queda de qualidade não é muito expressiva, com aproximadamente 0,32dB. Cabe salientar que a abordagem de Kao (2006), com suporte a tamanho de bloco variável e baseada em modelo matemático também apresenta uma queda de qualidade de aproximadamente 0,27dB.

Através dos resultados de desempenho obtidos com as arquiteturas é possível constatar que a proposta de Corrêa (2011a) apresenta as melhores taxas de processamento, sendo capaz de processar vídeos QHDTV (3840x2048) em tempo real, enquanto os outros trabalhos, de Oktem (2007) e de Kao (2006), processam vídeos VGA (640x480) e QUXGA (3200x2400), respectivamente. Analisando os resultados de frequência contextualizados com os demais resultados, se pode observar que a solução de Kao (2006) é relevante se for considerado que o foco do trabalho está na redução do consumo energético, pois esta solução alcançou uma elevada taxa de processamento, com pequena queda na qualidade de imagem e uma frequência muito menor em relação ao trabalho de Corrêa (2011a).

Tabela 3.2 – Resultados obtidos em diferentes arquiteturas de FME com precisão de ¼ de pixel para o padrão H.264/AVC.

Artigo Científico	**Oktem (2007)**	**Kao (2006)**	**Corrêa (2011a)**
Tecnologia	FPGA Virtex II	TSMC 0,13µm	FPGA Stratix III
Suporte para Tamanho de Bloco Variável	Sim	Sim	Não
Queda na qualidade (PSNR)	-	~ 0,27dB	~ 0,32dB
Frequência Máxima	60 MHz	100 MHz	245 MHz
Resolução obtida	VGA (640x480)	QUXGA (3200x2400)	QHDTV (3840x2048)

Com as comparações realizadas é possível constatar que através de escolhas adequadas é possível conseguir um bom equilíbrio entre qualidade de imagem, custo de hardware e desempenho, utilizando blocos de tamanho fixo. 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 para, com base nos resultados, observar quedas de qualidade de imagem. 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 dois softwares, um 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, da Altera. Por sua vez, o processo de validação foi feito com auxílio do software Microsoft Office Excel 2007, no qual foram geradas planilhas com valores de saída para comparação com os resultados obtidos na simulação do Quartus II.

4.2 Arquiteturas de Hardware Desenvolvidas e Metodologia Utilizada

Este trabalho apresenta o desenvolvimento de arquiteturas de hardware para FME com precisão de ¼ de pixel para as amostras de luminância 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 deste trabalho está no desenvolvimento dos filtros de interpolação que geram as posições de sub-pixel a partir das amostras de luminância em posições inteiras. Para o projeto

foi considerado que as amostras de luminância são de 8 bits e do tipo *unsigned*. A localização dos filtros de interpolação na FME pode ser observada na Figura 4.1.

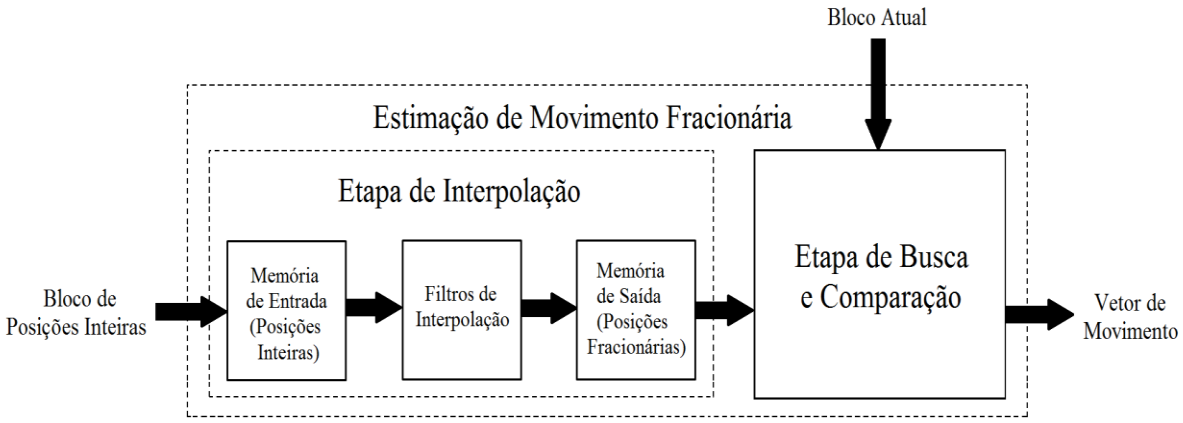

Figura 4.1 – Diagrama da FME com detalhamento da etapa de interpolação.

As arquiteturas de hardware desenvolvidas foram baseadas em dois documentos elaborados pelo JCT-VC: HM3 – *High Efficiency Video Coding* (HEVC) *Test Model 3 Encoder Description* (McCANN *et al.,* 2011) e WD3 – *Working Draft 3 of High-Efficiency Video Coding* (WIEGAND *et al.,* 2011). 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 (quadrados sombreados e com letras maiúsculas), bem como as 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. Amostras em posições inteiras e amostras em posições fracionárias para interpolação das amostras de luminância com precisão de ¼ de pixel do padrão HEVC.

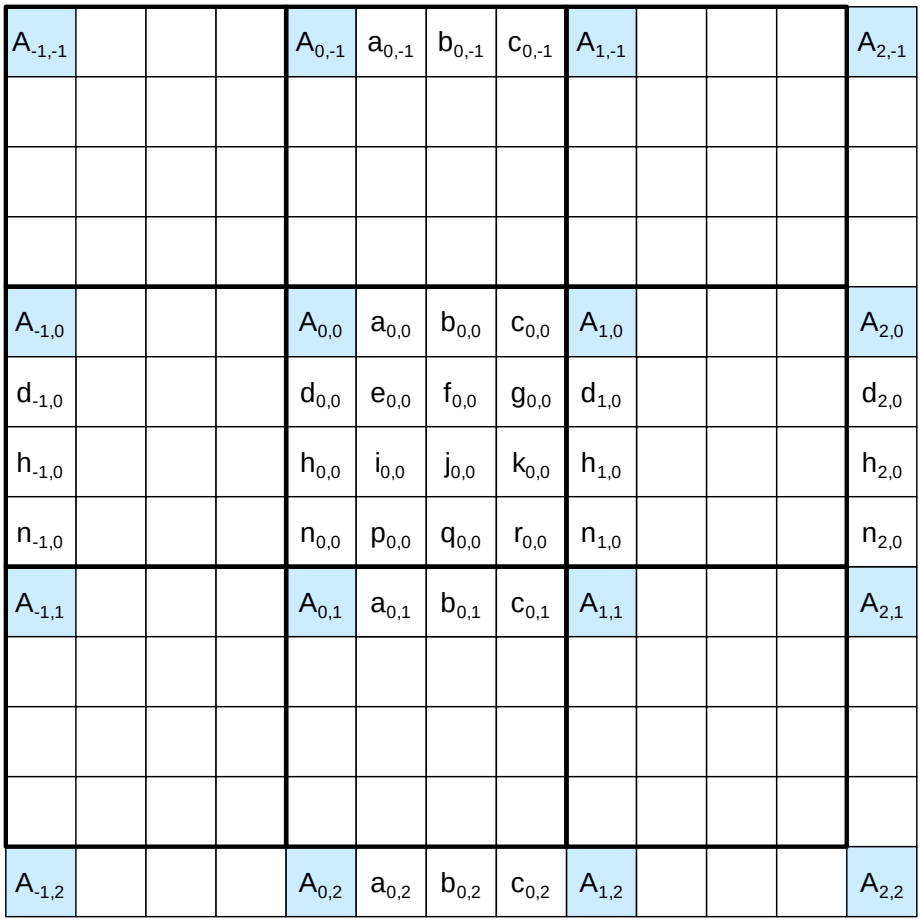

Figura 4.2 – Amostras de luminância em posições inteiras e fracionárias do HEVC.
Fonte: (WIEGAND *et al.,* 2011, p. 121).

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, que nada mais são do que os algoritmos retirados do documento WD3 (WIEGAND *et al.,* 2011)..

$$a_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 10*A_{-1,0} + 57*A_{0,0} + 19*A_{1,0} - 7*A_{2,0} + 3*A_{3,0} - A_{4,0} + offset1) >> shift1 \quad (1)$$

$$b_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 11*A_{-1,0} + 40*A_{0,0} + 40*A_{1,0} - 11*A_{2,0} + 4*A_{3,0} - A_{4,0} + offset1) >> shift1 \quad (2)$$

$$c_{0,0} = (-A_{-3,0} + 3*A_{-2,0} - 7*A_{-1,0} + 19*A_{0,0} + 57*A_{1,0} - 10*A_{2,0} + 4*A_{3,0} - A_{4,0} + offset1) >> shift1 \quad (3)$$

$$d_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 10*A_{0,-1} + 57*A_{0,0} + 19*A_{0,1} - 7*A_{0,2} + 3*A_{0,3} - A_{0,4} + offset1) >> shift1 \quad (4)$$

$$h_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 11*A_{0,-1} + 40*A_{0,0} + 40*A_{0,1} - 11*A_{0,2} + 4*A_{0,3} - A_{0,4} + offset1) >> shift1 \quad (5)$$

$$n_{0,0} = (-A_{0,-3} + 3*A_{0,-2} - 7*A_{0,-1} + 19*A_{0,0} + 57*A_{0,1} - 10*A_{0,2} + 4*A_{0,3} - A_{0,4} + offset1) >> shift1 \quad (6)$$

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 de valores das posições fracionárias intermediárias $d1_{i,0}$, $h1_{i,0}$ e $n1_{i,0}$ onde i varia de -3 a 4. Os valores são obtidos a partir

das amostras de luminância em posições inteiras e de filtros FIR de 8 *taps*, como pode ser observado nas equações 7-9.

$$d1_{i,0} = -A_{i,-3} + 4*A_{i,-2} - 10*A_{i,-1} + 57*A_{i,0} + 19*A_{i,1} - 7*A_{i,2} + 3*A_{i,3} - A_{i,4} \quad (7)$$

$$h1_{i,0} = -A_{i,-3} + 4*A_{i,-2} - 11*A_{i,-1} + 40*A_{i,0} + 40*A_{i,1} - 11*A_{i,2} + 4*A_{i,3} - A_{i,4} \quad (8)$$

$$n1_{i,0} = -A_{i,-3} + 3*A_{i,-2} - 7*A_{i,-1} + 19*A_{i,0} + 57*A_{i,1} - 10*A_{i,2} + 4*A_{i,3} - A_{i,4} \quad (9)$$

Após serem encontrados os valores intermediários $d1_{i,0}$, $h1_{i,0}$ e $n1_{i,0}$, finalmente pode ser realizado 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 $r_{0,0}$ que, além de utilizar os valores intermediários, utiliza os filtros FIR de 8 *taps*, conforme as equações 10-18.

$$e_{0,0} = (-d1_{-3,0} + 4*d1_{-2,0} - 10*d1_{-1,0} + 57*d1_{0,0} + 19*d1_{1,0} - 7*d1_{2,0} + 3*d1_{3,0} - d1_{4,0} + offset2) >> shift2 \quad (10)$$

$$f_{0,0} = (-d1_{-3,0} + 4*d1_{-2,0} - 11*d1_{-1,0} + 40*d1_{0,0} + 40*d1_{1,0} - 11*d1_{2,0} + 4*d1_{3,0} - d1_{4,0} + offset2) >> shift2 \quad (11)$$

$$g_{0,0} = (-d1_{-3,0} + 3*d1_{-2,0} - 7*d1_{-1,0} + 19*d1_{0,0} + 57*d1_{1,0} - 10*d1_{2,0} + 4*d1_{3,0} - d1_{4,0} + offset2) >> shift2 \quad (12)$$

$$i_{0,0} = (-h1_{-3,0} + 4*h1_{-2,0} - 10*h1_{-1,0} + 57*h1_{0,0} + 19*h1_{1,0} - 7*h1_{2,0} + 3*h1_{3,0} - h1_{4,0} + offset2) >> shift2 \quad (13)$$

$$j_{0,0} = (-h1_{-3,0} + 4*h1_{-2,0} - 11*h1_{-1,0} + 40*h1_{0,0} + 40*h1_{1,0} - 11*h1_{2,0} + 4*h1_{3,0} - h1_{4,0} + offset2) >> shift2 \quad (14)$$

$$k_{0,0} = (-h1_{-3,0} + 3*h1_{-2,0} - 7*h1_{-1,0} + 19*h1_{0,0} + 57*h1_{1,0} - 10*h1_{2,0} + 4*h1_{3,0} - h1_{4,0} + offset2) >> shift2 \quad (15)$$

$$p_{0,0} = (-n1_{-3,0} + 4*n1_{-2,0} - 10*n1_{-1,0} + 57*n1_{0,0} + 19*n1_{1,0} - 7*n1_{2,0} + 3*n1_{3,0} - n1_{4,0} + offset2) >> shift2 \quad (16)$$

$$q_{0,0} = (-n1_{-3,0} + 4*n1_{-2,0} - 11*n1_{-1,0} + 40*n1_{0,0} + 40*n1_{1,0} - 11*n1_{2,0} + 4*n1_{3,0} - n1_{4,0} + offset2) >> shift2 \quad (17)$$

$$r_{0,0} = (-n1_{-3,0} + 3*n1_{-2,0} - 7*n1_{-1,0} + 19*n1_{0,0} + 57*n1_{1,0} - 10*n1_{2,0} + 4*n1_{3,0} - n1_{4,0} + offset2) >> shift2 \quad (18)$$

Algumas variáveis utilizadas para o cálculo dos valores de luminância com precisão de sub-pixel são definidas segundo o documento WD3 (WIEGAND *et al.,* 2011) pelas seguintes expressões.

- *bit_depth_luma_minus8* pode ter valores de 0 a 6
- $BitDepth_Y = 8 + bit\_depth\_luma\_minus8$
- $shift1 = BitDepth_Y - 8$
- $shift2 = BitDepth_Y - 2$
- *offset1* = 0 se *shift1* = 0; nos outros casos $offset1 = 1 << (shift1 - 1)$
- $offset2 = 1 << (shift2 - 1)$

Contudo, no software de referência do HEVC (HM, 2012), 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 WD3 (WIEGAND *et al.,* 2011), é possível calcular os valores das demais variáveis, como segue abaixo:

- *bit_depth_luma_minus8* = 6
- $BitDepth_Y$ = 14
- *shift1* = 6
- *shift2* = 12
- *offset1* = 32
- *offset2* = 2048

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 6 grupos diferentes de hardware para construção da arquitetura da FME. As posições $a_{0,0}$, $d_{0,0}$, $c_{0,0}$ e $n_{0,0}$, por exemplo, podem utilizar o mesmo hardware, uma vez que apesar das posições inteiras utilizadas serem diferentes, as constantes utilizadas nas multiplicações são as mesmas para as 4 posições.

Além disso, os custos de hardware e desempenho para a implementação das operações que envolvem *shift1*, *shift2*, *offset1* e *offset2* são pequenos. 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, desconsiderando inicialmente as operações que envolviam as variáveis *shift* e *offset*. 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 1, o filtro utilizado para o cálculo das posições fracionárias $a_{0,0}$, $d_{0,0}$, $c_{0,0}$, $n_{0,0}$, $d1_{i,0}$, $n1_{i,0}$, $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $g_{0,0}$, $k_{0,0}$ e $r_{0,0}$. Por sua vez, o filtro tipo 2 é utilizado para o cálculo das posições fracionárias $b_{0,0}$, $h_{0,0}$, $h1_{i,0}$, $f_{0,0}$, $j_{0,0}$ e $q_{0,0}$.

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

Posições fracionárias	Similaridade
$a_{0,0}$, $d_{0,0}$, $c_{0,0}$ e $n_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –7, –10, 3, 4, 19 e 57), *shift1* e *offset1*.
$b_{0,0}$ e $h_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40), *shift1* e *offset1.*
$d1_{i,0}$ e $n1_{i,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –7, –10, 3, 4, 19 e 57).
$h1_{i,0}$	Utiliza as seguintes constantes nas multiplicações (–1, –11, 4 e 40).
$e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $g_{0,0}$, $k_{0,0}$ e $r_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –7, –10, 3, 4, 19 e 57), *shift2* e *offset2.*
$f_{0,0}$, $j_{0,0}$ e $q_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40), *shift2* e *offset2.*

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

Filtro	Posições fracionárias	Similaridade
Tipo 1	$a_{0,0}$, $d_{0,0}$, $c_{0,0}$, $n_{0,0}$, $d1_{i,0}$, $n1_{i,0}$, $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $g_{0,0}$, $k_{0,0}$ e $r_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –7, –10, 3, 4, 19 e 57).
Tipo 2	$b_{0,0}$, $h_{0,0}$, $h1_{i,0}$, $f_{0,0}$, $j_{0,0}$ e $q_{0,0}$	Utilizam as mesmas constantes nas multiplicações (–1, –11, 4 e 40).

Para o grupo referente às posições $a_{0,0}$, $d_{0,0}$, $c_{0,0}$, $n_{0,0}$, $d1_{i,0}$, $n1_{i,0}$, $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $g_{0,0}$, $k_{0,0}$ e $r_{0,0}$ foram realizadas várias otimizações. Com o objetivo de dar maior clareza às otimizações realizadas, as representações das amostras inteiras e intermediárias no documento WD3 (WIEGAND *et al.,* 2011) foram substituídas por a_0-a_7. 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 20. 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 21. Isto equivale a dizer que os deslocamentos necessários podem ser realizados apenas uma vez. Com o objetivo de reaproveitar hardware foram avaliadas também quais as operações de soma/subtração se repetiam, como pode ser observado em negrito na Equação 22. Finalmente, pode ser visto na Equação 23 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.

Posição de sub-pixel = $– a_0 + 4*a_1 – 10*a_2 + 57*a_3 + 19*a_4 – 7*a_5 + 3*a_6 – a_7$ (19)

$= – a_0 + 4*a_1 – (8*a_2 + 2*a_2) + (64*a_3 + a_3 – 8*a_3) + (16*a_4 + 2*a_4 + a_4) – (8*a_5 – a_5) + (4*a_6 – a_6) – a_7$ (20)

$= 1*(– a_0 – a_6 – a_7 + a_3 + a_4 + a_5) + 2*(a_4 – a_2) + 4*(a_1 + a_6) + 8*(– a_2 – a_3 – a_5) + 16*(a_4) + 64*(a_3)$ (21)

$= 1*(– a_0 – a_6 – a_7 + a_4 + \mathbf{(a_3 + a_5)}) + 2*(a_4 – a_2) + 4*(a_1 + a_6) + 8*(– a_2 – \mathbf{(a_3 + a_5)}) + 16*(a_4) + 64*(a_3)$ (22)

$= (– a_0 – a_6 – a_7 + a_4 + \mathbf{(a_3 + a_5)}) + (a_4 – a_2) << 1 + (a_1 + a_6) << 2 + (– a_2 – \mathbf{(a_3 + a_5)}) << 3 + (a_4) << 4 + (a_3) << 6$ (23)

No caso das posições $b_{0,0}$, $h_{0,0}$, $h1_{i,0}$, $f_{0,0}$, $j_{0,0}$ e $q_{0,0}$ 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 24-28 pode ser observado todo o processo de otimização para o segundo filtro.

Posição de sub-pixel = $– a_0 + 4*a_1 – 11*a_2 + 40*a_3 + 40*a_4 – 11*a_5 + 4*a_6 – a_7$ (24)

$= – a_0 + 4*a_1 – (8*a_2 + 2*a_2 + a_2) + (32*a_3 + 8*a_3) + (32*a_4 + 8*a_4) – (8*a_5 + 2*a_5 + a_5) + 4*a_6 – a_7$ (25)

$= 1*(– a_0 – a_2 – a_5 – a_7) + 2*(– a_2 – a_5) + 4*(a_1 + a_6) + 8*(– a_2 + a_3 + a_4 – a_5) + 32*(a_3 + a_4)$ (26)

$= 1*(– a_0 – a_7 – \mathbf{(a_2 + a_5)}) + 2*(– \mathbf{(a_2 + a_5)}) + 4*(a_1 + a_6) + 8*(\mathbf{(a_3 + a_4)} – \mathbf{(a_2 + a_5)}) + 32*\mathbf{(a_3 + a_4)}$ (27)

$= (– a_0 – a_7 – \mathbf{(a_2 + a_5)}) + (– \mathbf{(a_2 + a_5)}) << 1 + (a_1 + a_6) << 2 + (\mathbf{(a_3 + a_4)} – \mathbf{(a_2 + a_5)}) << 3 + \mathbf{(a_3 + a_4)} << 5$ (28)

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 estágios de *pipeline* destes filtros. Os tamanhos das entradas a_0-a_7 utilizadas nas equações 23 e 28 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 -4845 e 21165 ou -6120 e 22440 para algumas das posições fracionadas, dependendo do tipo de filtro. Contudo, os filtros das equações 23 e 28 podem ser utilizados considerando como entradas valores de amostras de posições fracionárias, requerendo que o tamanho das entradas seja de 16 bits. Dessa forma, como o filtro funcionará com *pipeline* e um mesmo filtro poderá receber na entrada 8 ou 16 bits, dependendo das posições a serem calculadas, o tamanho adotado para as entradas dos filtros foi de 16 bits. As duas arquiteturas com *pipeline* desenvolvidas podem ser observadas nas figuras 4.3 e 4.6, onde são apresentados os esquemas para o filtro tipo 1 e filtro tipo 2, respectivamente.

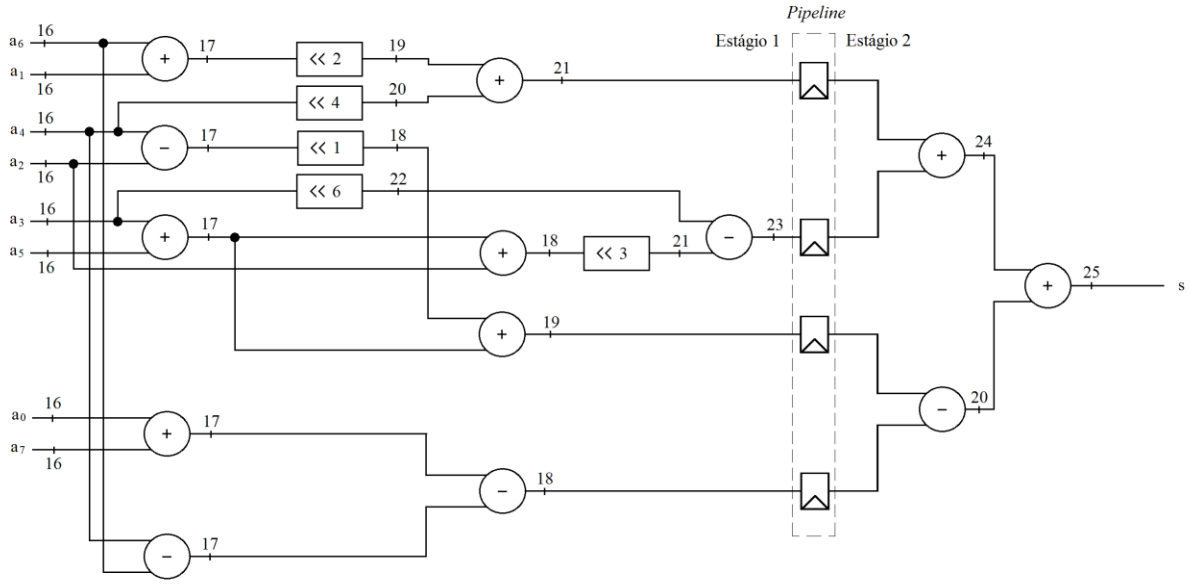

Figura 4.3 – Arquitetura de hardware do filtro tipo 1.

Nas figuras 4.4 e 4.5 se pode observar o hardware necessário para ser conectado à saída do filtro tipo 1 para que seja possível a realização das operações com *shift1* e *offset1*, e *shift2* e *offset2*, respectivamente.

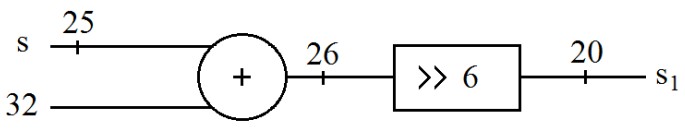

Figura 4.4 – Arquitetura com as operações de *shift1* e *offset1* para o filtro tipo 1.

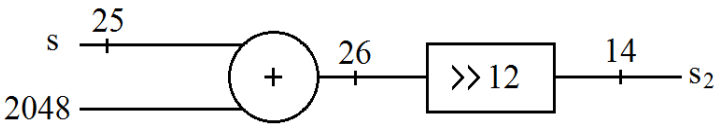

Figura 4.5 – Arquitetura com as operações de *shift2* e *offset2* para o filtro tipo 1.

Como pode ser observado nos esquemas das figuras 4.3 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, em todos os somadores/subtratores cujas entradas não apresentam o mesmo tamanho, estas operações estão presentes. 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/subtrator.

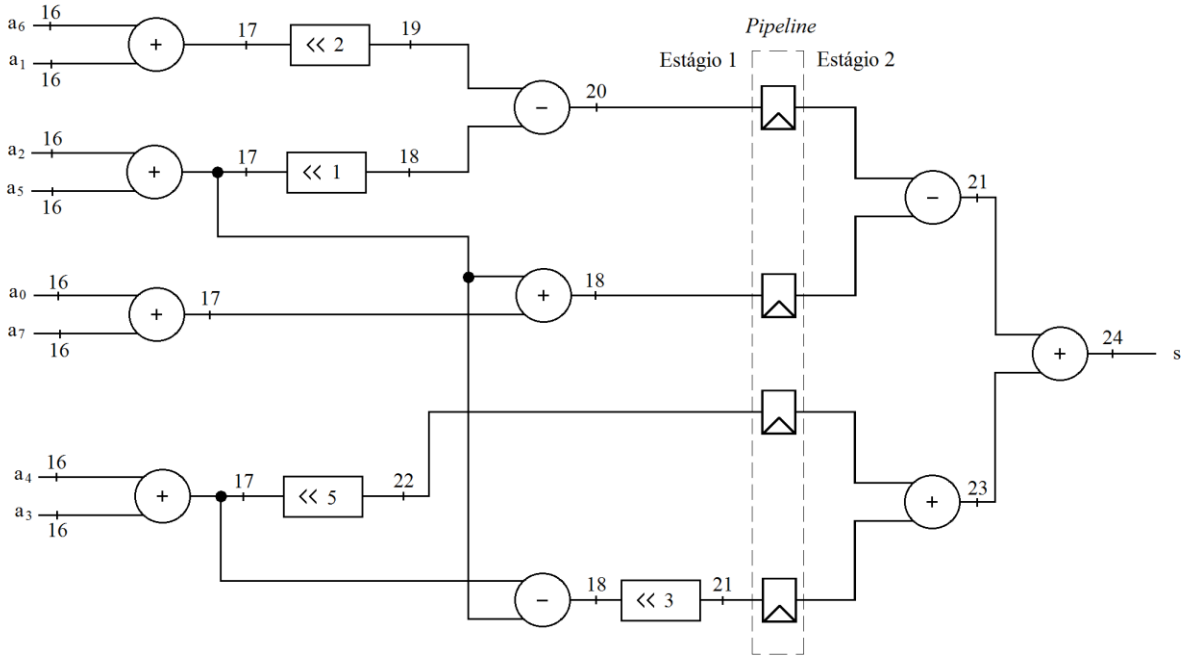

Figura 4.6 – Arquitetura de hardware do filtro tipo 2.

Nas figuras 4.7 e 4.8 se pode observar o hardware necessário para ser conectado a saída do filtro tipo 2 para que seja possível a realização das operações com *shift1* e *offset1*, e *shift2* e *offset2*, respectivamente.

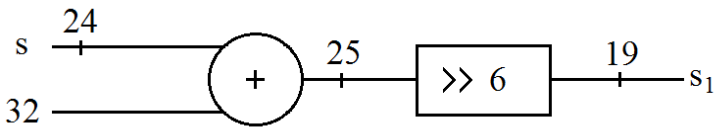

Figura 4.7 – Arquitetura com as operações de *shift1* e *offset1* para o filtro tipo 2.

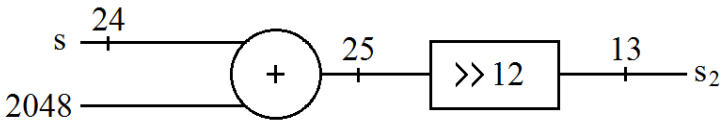

Figura 4.8 – Arquitetura com as operações de *shift2* e *offset2* para o filtro tipo 2.

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 dos filtros de interpolação foram descritas em VHDL através da ferramenta Quartus II da Altera. Os resultados obtidos 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 1, enquanto que os resultados das versões do filtro tipo 2 podem ser vistos na Tabela 5.2. Todos os resultados foram obtidos considerando o dispositivo FPGA Stratix II, modelo EP2S15F484C3. O cálculo dos valores de Frequência Máxima para as versões com *pipeline* dos filtros de interpolação foram obtidos utilizando os valores de *Worst-case* tco fornecidos pelo software, onde a Frequência Máxima pode ser obtida simplesmente dividindo 1 pelo valor de *worst-case.*

Tabela 5.1 – Resultados obtidos com as versões do filtro tipo 1.

Resultados	**Versão do filtro de interpolação tipo 1**			
	Pipeline* + *shift2/offset2	***Pipeline* + *shift1/offset1***	**Somente *pipeline***	**Combinacional**
Total de ALUTs	275	281	259	259
Total de registradores	216	222	227	0
Total de pinos I/O	145	151	156	153
***Worst-case* tco**	6,302 ns	6,356 ns	6,335 ns	–
Frequência Máxima	158,68 MHz	157,33 MHz	157,85 MHz	–
***Worst-case* tpd (versão combinacional)**	–	–	–	17,955 ns

Tabela 5.2 – Resultados obtidos com as versões do filtro tipo 2.

Resultados	**Versão do filtro de interpolação tipo 2**			
	Pipeline* + *shift2/offset2	***Pipeline* + *shift1/offset1***	**Somente *pipeline***	**Combinacional**
Total de ALUTs	238	244	223	223
Total de registradores	211	217	222	0
Total de pinos I/O	144	150	155	152
***Worst-case* tco**	6,546 ns	6,602 ns	6,306 ns	–
Frequência Máxima	152,77 MHz	151,47 MHz	158,58 MHz	–
***Worst-case* tpd (versão combinacional)**	–	–	–	16,298 ns

5.2 Validação

Para validação dos resultados obtidos com a simulação das arquiteturas desenvolvidas na ferramenta Quartus II, foi criada uma planilha no aplicativo Excel da Microsoft, na qual foram colocados diversos valores de entradas, obtendo-se os valores de saída de acordo com o tipo do filtro de interpolação utilizado. Após ambos os resultados serem obtidos, o da planilha Excel e da simulação no Quartus II, os valores foram confrontados. Esse processo foi repetido para diversos conjuntos de entradas, visando à utilização de valores críticos nas entradas sempre que possível.

Na Figura 5.1 pode ser observada a planilha do Excel utilizada juntamente com os valores de saída, dados alguns exemplos de entrada para os filtros de interpolação tipo 1 e tipo 2 na versão que utiliza somente *pipeline*. Já na Figura 5.2 pode ser observada a tela de simulação do Quartus II, considerando o filtro de interpolação tipo 1 que utiliza somente *pipeline* para os mesmos valores de entrada da Figura 5.1.

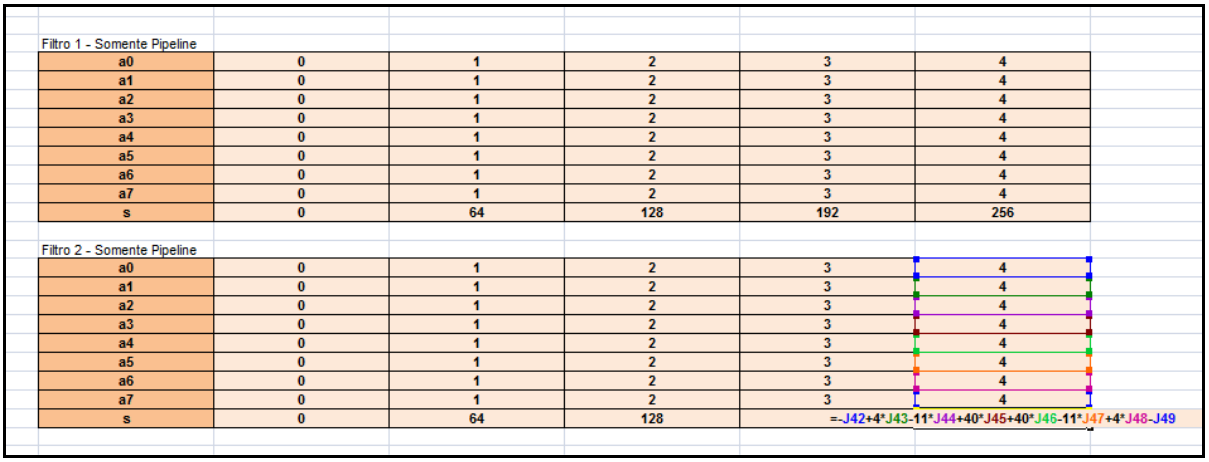

Figura 5.1 – Planilha do Excel utilizada para validação.

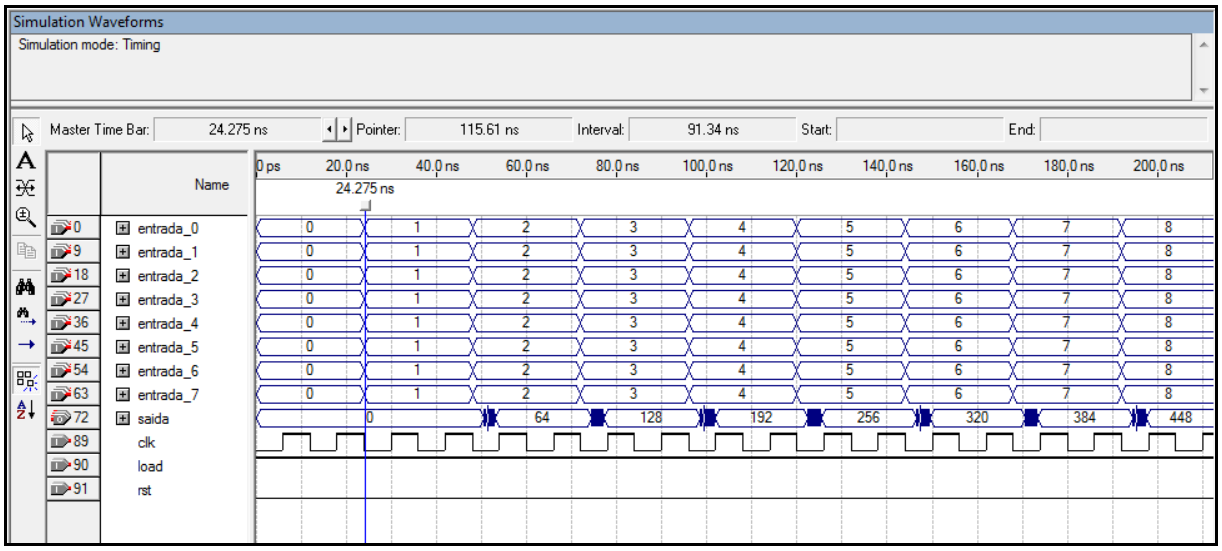

Figura 5.2 – Resultados da simulação para o filtro tipo 1.

5.3 Comentários sobre os Resultados

De posse dos resultados obtidos foi possível verificar que estes ainda podem ser melhorados em termos de desempenho para o processamento de vídeos de alta definição, foco do padrão de codificação de vídeo HEVC. Para o processamento em tempo real de vídeos *Full HD* (1920×1080) com subamostragem de cores no formato 4:2:0, por exemplo, é necessária uma taxa de processamento de 93,3 Mamostras/s, ou seja, uma amostra a cada 10,72ns. Como o objetivo é trabalhar com vídeos de resoluções maiores que *Full HD*, na continuação do desenvolvimento da etapa de interpolação e na implementação do restante da FME, deverão ser adotadas estratégias para atingir taxas de processamento bem superiores a 93,3 Mamostras/s, para que seja possível o processamento em tempo real destes vídeos. Para processar vídeos QFHD (3840x2160), por exemplo, será necessária uma taxa de processamento quatro vezes maior, quando comparado com vídeos *Full HD*. Isto resulta no processamento de uma amostra a cada 2,679 ns. Dessa forma, serão avaliadas uma série de estratégias com o objetivo de melhorar os resultados obtidos, como por exemplo, reduzir o número de bits de somadores e subtratores, através de uma análise da faixa de valores possíveis em entradas e saídas; aumentar o número de estágios de *pipeline* dos filtros; e a modificação das arquiteturas com operações de *shift* e *offset*.

Nos processos de validação de resultados dos trabalhos futuros pretende-se utilizar o software de referência do padrão HEVC (HM, 2012) juntamente com o

software ModelSim da Altera para permitir uma validação mais adequada, uma vez que a quantidade de hardware a ser validado, bem como o número de entradas e saídas irá aumentar significativamente.

6 CONCLUSÕES

Com o trabalho proposto foi possível realizar um estudo sobre o estado da arte em termos de arquiteturas para FME do padrão de codificação de vídeo H.264/AVC. Como o padrão HEVC ainda se encontra em desenvolvimento, não foram encontradas na literatura científica arquiteturas de FME desenvolvidas para tal padrão. Com base em comparações entre os trabalhos relacionados foi possível analisar o impacto da utilização de algumas estratégias durante a especificação da arquitetura.

Este trabalho também permitiu um embasamento teórico sobre a FME e a constatação de que as principais diferenças entre os processos de FME dos padrões HEVC e H.264/AVC estão relacionadas à etapa de geração das posições interpoladas, mais especificamente aos filtros utilizados.

Foram desenvolvidas arquiteturas de hardware para a implementação dos filtros de interpolação da FME do padrão de codificação de vídeo HEVC. As arquiteturas de hardware projetadas fazem parte da etapa de interpolação da FME, que será completamente desenvolvida em trabalhos futuros. O trabalho realizado corresponde a um esforço inicial para o desenvolvimento de arquiteturas para FME com precisão de ¼ de pixel para as amostras de luminância. Além da etapa de interpolação, será desenvolvida uma etapa de busca de ¼ de pixel para, após a integração destas etapas, se obter 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.

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 Test Model (HM). **Reference Software**. Disponível em: <http://hevc.info/HM-doc/> Acesso em: janeiro 2012.

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.

McCANN, K., BROSS, B., SEKIGUCHI, S. e HAN, W. **High Efficiency Video Coding Test Model 3 Encoder Description (JCTVC-E602)**, [S.l.], 2011.

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.** 2011. 117f. Proposta de 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.

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.

WIEGAND, T., HAN, W., BROSS, B., OHM, J. e SULLIVAN, G. **Working Draft 3 of High-Efficiency Video Coding (JCTVC-E603)**, [S.l.], 2011.

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.