UNIVERSIDADE FEDERAL DO PAMPA

DAVIDSON PEREIRA MACHADO

IMPLEMENTAÇÃO E VALIDAÇÃO DE UMA ARQUITETURA EM HARDWARE DO INTERPOLADOR PARA O BLOCO DE COMPENSAÇÃO DE MOVIMENTO BASEADO NO PADRÃO HEVC

Bagé
2016

DAVIDSON PEREIRA MACHADO

IMPLEMENTAÇÃO E VALIDAÇÃO DE UMA ARQUITETURA EM HARDWARE DO INTERPOLADOR PARA O BLOCO DE COMPENSAÇÃO DE MOVIMENTO BASEADO NO PADRÃO HEVC

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Fábio Luís Livi Ramos

Bagé
2016

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais).

M252i MACHADO, DAVIDSON
IMPLEMENTAÇÃO E VALIDAÇÃO DE UMA ARQUITETURA EM HARDWARE DO INTERPOLADOR PARA O BLOCO DE COMPENSAÇÃO DE MOVIMENTO BASEADO NO PADRÃO HEVC / DAVIDSON MACHADO.
59 p.

Trabalho de Conclusão de Curso(Graduação)--
Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2016.
"Orientação: FÁBIO RAMOS".

1. HEVC. 2. VÍDEO CODING. 3. HARDWARE. 4. MOTION COMPENSATION. 5. CONSUMO DE POTÊNCIA.

DAVIDSON PEREIRA MACHADO

IMPLEMENTAÇÃO E VALIDAÇÃO DE UMA ARQUITETURA EM HARDWARE DO INTERPOLADOR PARA O BLOCO DE COMPENSAÇÃO DE MOVIMENTO BASEADO NO PADRÃO HEVC

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 08 de julho e 2016.

Banca examinadora:

_____

Prof. MSc. Fábio Luís Livi Ramos
UNIPAMPA

_____

Prof. Dr. Bruno Silveira Neves
UNIPAMPA

_____

Prof. MSc. Julio Saraçol Domingues Júnior
UNIPAMPA

AGRADECIMENTO

Agradeço primeiramente a Deus que me permitiu vivenciar esse momento de imensa alegria e satisfação de mais uma etapa de vida concluída.

Agradeço aos meus pais e irmã, pelo amor, incentivo e apoio incondicional que sempre me deram ao longo de minha vida, e não somente nestes anos como universitário.

Agradecimentos aos meus amigos, aos que chamarei de irmãos, agradeço minha namorada. Agradeço por fazerem parte da minha formação e por fazerem parte da minha vida.

Agradeço a todos os professores por me proporcionarem conhecimentos e lições de vida que hoje fomenta o caráter que levarei adiante, serei eternamente grato por me tornarem a pessoa que sou, uma pessoa mais capaz de dar continuidade a essa busca pela realização das minhas idealizações.

A todos que direta ou indiretamente fizeram parte da minha formação, os meus mais sinceros agradecimentos.

RESUMO

O grande aumento de dispositivos com altas resoluções faz crescer também a demanda pela qualidade de vídeo que, por sua vez, aumenta o tráfego de vídeo pela Internet e demais formas de comunicação. Com isso, faz-se necessária a codificação de vídeo, afim de eliminar informações redundantes presente em sequências de vídeo. A tarefa da codificação apresenta alto grau de complexidade, o que inviabiliza que sua execução puramente em software consiga atender aos usuários, com vídeos que apresentem taxas de quadros suficiente para que não ocorra *latency-at-game* (LAG), isto é, que o intervalo de tempo de exibição entre os quadros seja grande suficientemente para que o usuário sinta a sensação de travamentos. A bateria é o maior gargalo da tecnologia móvel moderna, o que implica que a mesma deva ser utilizada de forma eficiente. Assim sendo, soluções em hardware surgem como possibilidades de contorno, desses contextos. O presente trabalho propõe a implementação em hardware do bloco interpolador dentro do contexto da compensação de movimento baseado no padrão HEVC de codificação de vídeo, visando ganhos em desempenho e redução do consumo de potência.

Palavras-Chave: HEVC, Codificação de Vídeo, *Hardware*, Compensação de Movimento, Consumo de potência.

ABSTRACT

The large increase in devices with high resolutions also grows the demand for video quality that, in turn, increases the video traffic over the Internet and other forms of communication. Thus, it is necessary to encode video in order to eliminate redundant information in video sequences. The task of coding has a high degree of complexity, which prevents its performance purely in software to serve users with videos that have frame rates enough, so that there is latency-at-game (LAGs), namely the display time interval between frames, which is large enough for the user to feel the sense of crashes. Battery is the main bottleneck of the current mobile embedded technology, which implies it should be used efficiently. Therefore, hardware solutions come as contour possibilities at this mentioned context. This work proposes a hardware implementation of interpolating block within the motion compensation context based on the standard video encoding HEVC to achieve gains in performance and reduced power consumption.

Keywords: HEVC, Video Coding, Hardware, Motion Compensation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AVC - Advanced Video Coding

CABAC - Context Adaptive Binary Arithmetic Coding

Cb - Crominância azul

Cr - Crominância vermelha

CTB - Coding Tree Block

CTU - Coding Tree Unit

CU - Coding Unit

FPGA - Field Programmable Gate Array

FME - Fractional Motion Estimation

FREB - Filtro Redutor de Efeitos de Bloco

GNU - General Public License

HEVC - High Efficiency Video Coding

HM - Modelo HEVC Test

HSI - Hue, Saturation e Intensity

IP – Intra-predição

ITU-T - ITU Telecommunication Standardization Sector

JCT-VC - Joint Collaborative Team on Video Coding

MC – Motion Compensation

ME - Motion Estimation

MPEG - Moving Picture Experts Group

PIXEL - picture element

PU - Prediction Unit

Q - Quantização direta

QI - Quantização indireta

RGB - Red, Green, Blue

SAO - Sample Adaptive Offset

SVN - Subversion

T - Transformadas diretas

TV - Televisão

TU - Transform Unit

TI - Transformadas inversas

VHDL - VHSIC Hardware Description Language

VCEG - Video Coding Experts Group

YCbCr – Luminância, Crominância azul, Crominância vermelha

Y – Luminância

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, observa-se uma grande tendência para que tudo esteja cada vez mais conectado. Isso é possível graças ao uso da Internet e demais formas de comunicação entre dispositivos, tais como o Bluetooth, por exemplo. Além disso, são vários os dispositivos que fazem uso da comunicação com outros equipamentos eletrônicos (e.g. Smartphones, relógios inteligentes, geladeiras, TVs, notebooks, entre outros).

Porém, com o aumento de dispositivos conectados, o tráfego de dados pela rede aumenta cada vez mais o que, por sua vez, pode se tornar um gargalo de comunicação, o que motiva o estudo de medidas que possam evitar esse empecilho.

Tendo como base a popularidade e demanda das aplicações multimídia, bem como a intensidade e complexidade de seu volume de tráfego, de 80% a 90% de todo trafego global de Internet em 2019 estará associado a transmissões de vídeo sobre IP (CISCO, 2015). Nesse contexto, observa-se a necessidade de codificação para que esses possam ser transmitidos com eficiência.

A codificação de vídeo tem como objetivo a redução do número de bits necessários para representar um vídeo digital, sem que se tenha perda de informações significativas para sua visualização (i.e. mantendo a mesma qualidade visual esperado pelo usuário).

A compactação de vídeo é efetuada através da exploração de uma propriedade existente em sequências de vídeo: a redundância de informações, o que implica que grande parte dos dados necessários para representar um vídeo digital seja supérflua (DA SILVA, 2007).

O padrão H.264/AVC é o padrão mais utilizado atualmente em dispositivos de compressão de vídeo, tendo sido desenvolvido com o objetivo de dobrar a taxa de compressão em relação aos padrões anteriores, tal como o MPEG-2 (SUNNA, 2014). A primeira versão do H.264/AVC foi aprovada em 2003 e, desde então, este padrão tem sido foco de incessante pesquisa e desenvolvimento por grupos espalhados ao redor do mundo (ROSA, 2007).

O novo padrão de compressão de vídeo, o *High Efficiency Video Coding* (HEVC), promete compactar vídeos com o dobro da capacidade do H.264/AVC, mantendo a mesma qualidade visual do anterior (SULLIVAN, 2012). O novo padrão

apresenta uma série de ferramentas, melhorias e mudanças em relação ao padrão de codificação anterior (SULLIVAN, 2012).

Uma das principais caraterísticas de dispositivos portáteis é a de depender de alimentação por bateria para a sua operação. Adicionalmente, em várias situações, o tipo de carga a ser processada apresenta requisitos de tempo real (URRIZA, 2004).

Dado o grau de complexidade do codificador HEVC, composto por várias etapas de compactação, atualmente, é inviável atingir o desempenho de tempo real utilizando uma solução do codificador puramente em software. Assim sendo, uma solução em hardware necessária para atingir tal desempenho (TEIXEIRA, 2014).

Considerando as diversas funcionalidades do codificador de vídeo HEVC, as etapas de codificação de vídeo podem ser divididas em diversos blocos operacionais, cada um contendo uma etapa dentro do fluxo de codificação de vídeo, as quais seriam, de uma maneira macro: intra-predição, inter-predição, transformadas, quantização, codificação de entropia e filtros (ITU-T, 2013). Cada etapa citada, no todo, farão o codificador HEVC alcançar a taxa de compactação de dados pretendida.

Presente no codificador como no decodificador, a compensação de movimento é responsável por reconstruir o *frame* atual utilizando *frames* vizinhos, esses nomeados de *frames* de referência (ZATT et al., 2006). A compensação de movimento faz uso de filtros de interpolação com precisão de até ¼ de pixel para a reconstrução do quadro a ser armazenado/enviado adiante (ITU-T, 2013).

Sendo interpolação na matemática definida como o método que permite construir um novo conjunto de dados a partir de um conjunto discreto de dados pontuais previamente conhecidos, sendo neste trabalho tal técnica aplicada a vídeo.

Nesse contexto, o presente trabalho apresenta a implementação e validação de uma arquitetura em hardware para o interpolador do bloco de compensação de movimento baseado no padrão HEVC, visando ganho em desempenho e consumo de potência, visto que a arquitetura pode ser empregada em dispositivos onde o consumo de energia pode ser ou não um ponto de grande relevância.

Objetivando o desenvolvimento do trabalho, as seguintes etapas foram seguidas:

1. Estudo do padrão de codificação de vídeo HEVC;
2. Estudo da literatura sobre arquiteturas e soluções do estado da arte para o bloco interpolador da compensação de movimento;

3. Descrição da arquitetura do interpolador em VHDL;
4. Simulação e validação da arquitetura do interpolador, utilizando o software de referência como parâmetro;
5. Sintetize da arquitetura do interpolador para FPGA e analise de seu desempenho e consumo de potência.

As partes que compõem esse trabalho estão organizadas da seguinte maneira:

O capítulo 1 compreende a introdução do trabalho;

O capítulo 2 trata da apresentação geral sobre os conceitos de codificação de vídeo e sobre o padrão HEVC;

O capítulo 3 trata da apresentação do funcionamento do processo de interpolação dentro do escopo da compensação de movimento do padrão HEVC, além de serem apresentados os trabalhos do estado da arte encontrados na literatura;

O capítulo 4, trata da apresentação da arquitetura desenvolvida para o interpolador de compensação de movimento;

O capítulo 5 trata da apresentação do ambiente de validação, além dos resultados de síntese;

No capítulo 6 conclui-se esse trabalho.

2 CODIFICAÇÃO DE VÍDEO

Nesse capítulo serão apresentados conceitos importantes para o entendimento da codificação de dados, mais especificamente tratando sobre compressão de vídeo, no qual o escopo desse trabalho está inserido.

2.1 Compressão de vídeo

Comprimir um dado é o nome dado ao ato de reduzir o espaço necessário para armazená-lo. A compressão baseia-se na eliminação de redundância, uma vez que muitos dados apresentam informações que podem ser eliminadas, de modo a diminuir o tamanho para sua representação, seja ele para armazenamento ou transmissão pela rede.

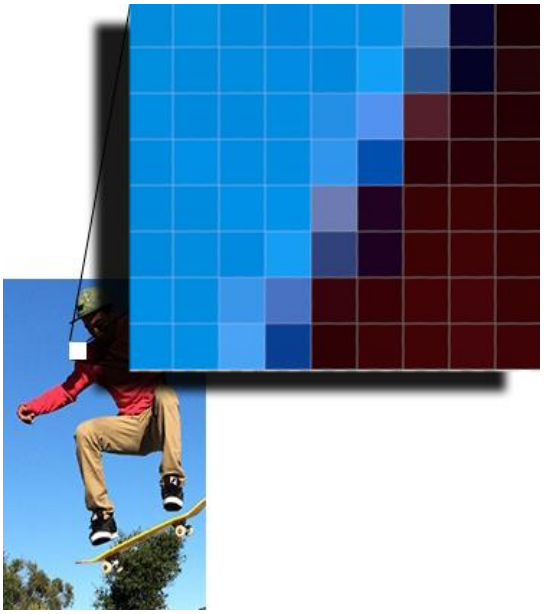
A compressão de vídeo não difere do conceito apresentando antes. Um vídeo corresponde a uma sequência de imagens que, ao serem exibidas uma após a outra, gera no espectador a sensação de movimento, ilustrado pela figura 1.

Figura 1 - Sequência de imagem

Fonte: Próprio autor

Cada imagem é representada por uma matriz formada pelo arranjo de linhas e colunas. Cada posição dessa matriz corresponde a unidade básica na formação de uma imagem digital, o *pixel (picture element*) (SANTOS et al*., 2015*). Cada imagem digital é constituída por milhares de *pixels*, ilustrados na figura 2, e esses, por sua vez, em uma ou mais componentes, por exemplo, como no caso mais comum, as componentes RGB – *Red*, *Green* e *Blue*.

Figura 2 - Zoom nos pixels da imagem

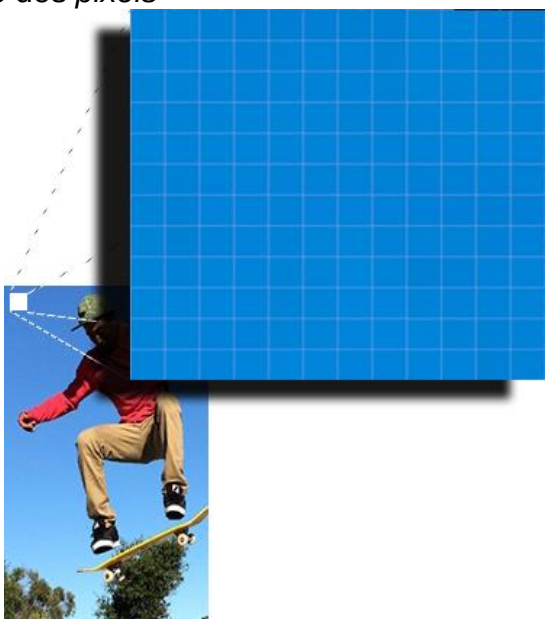


Fonte: Próprio autor

A compressão de vídeo pode ser entendida como a eliminação de redundância de informações presentes nas amostras de *pixels*. Comumente enquadradas em umas das duas a seguir:

- **Redundância espacial**: consiste na semelhança dos *pixels* adjacentes de uma imagem. A figura 3 ilustra um conjunto de *pixels* que são praticamente uniformes.

Figura 3 - Uniformidade dos *pixels*



Fonte: Próprio autor

- **Redundância temporal**: consiste no aproveitamento da similaridade existente entres as imagens, também conhecido como quadros sucessivos, que formam uma imagem dinâmica. A figura 4 ilustra visualmente esse conceito, onde um conjunto de *pixels* sofre apenas um deslocamento para a esquerda.

Figura 4 - Ilustração do deslocamento de pixels

Fonte: Próprio autor

Além das redundâncias citadas anteriormente, existe também a redundância psicovisual, que está relacionada à informação visual real ou quantificada em uma cena, onde algumas informações têm menos importância relativa do que outros no processamento visual normal, pois o olho humano não responde com uma mesma sensibilidade a todas as informações visíveis (DE OLIVEIRA, 2005).

Além do formato RGB já citado, existem outros espaços de cores utilizados na representação de imagens digitais, tais como *Hue, Saturation and Intensity* (HIS) e YCbCr. No espaço de cores YCbCr as três componentes utilizadas são luminância (Y), que define a intensidade luminosa ou o brilho, crominância azul (Cb) e crominância vermelha (Cr) (SHI; SUN, 1999).

Mesmo sendo o RGB o espaço de cor mais comum na representação de uma imagem digital, a compressão de vídeo é aplicada sobre espaços de cores do tipo YCbCr, isso graças a informação de cor estar completamente separada da informação de brilho, o que difere no RGB, onde as componentes apresentam alto grau de correlação (RICHARDSON, 2002). O motivo do uso do espaço YCbCr é que o sistema visual humano possui cerca de 240 milhões de bastonetes (células óticas sensíveis à

luz) e 13 milhões de cones (células óticas sensíveis à cor), que o faz ser mais sensível a informações de luminância do que de crominância (GONZALES; WOODS, 2003).

Assim sendo, ao fazer o uso do espaço YCbCr, se consegue aproveitar o comportamento do olho humano para reduzir as amostras de *pixel* que são menos sensíveis ao olho humano (e, portanto, que não vão acarretar praticamente nenhuma perda visual subjetiva). Para isto, as componentes de cor podem ser sub-amostradas em relação a componente de luz.

2.2 Padrão HEVC

Com a grande adesão de dispositivos de alta resolução por parte dos usuários, as operadoras de rede serão submetidas a novos desafios, quando se trata da oferta de largura de banda satisfatória para a demanda imposta pelos consumidores.

Como solução parcial para a questão citada antes, os holofotes são direcionados para o novo padrão de codificação de vídeo *High Efficiency Video Coding* (HEVC) que surgiu como sucessor do H.264 *Advanced Video Coding* (H.264/AVC), ofertando um ganho de até 50% na capacidade de compressão ante seu antecessor, mantendo a mesma qualidade visual (SULLIVAN,2012).

O HEVC é resultado dos esforços conjuntos entre as organizações de padronização ITUT VCEG (*Video Coding Experts Group*) e a ISO/IEC MPEG (*Moving Picture Experts Group*) em parceria nomeada como JCT-VC (*Joint Collaborative Team on Video Coding*) (ITU-T, 2013).

O primeiro modelo de teste do HEVC, intitulado de HM1.0, surgiu após 27 propostas submetidas para avaliação, no qual se observou melhorias significativas em relação ao H.264/AVC, sendo este modelo submetido a constante evolução a cada reunião feita do JCT-VC (BARONCINI et al., 2010).

O padrão HEVC teve como um dos princípios norteadores de projeto prover suporte para as aplicações no qual o padrão H.264/AVC já estava difundido. A implementação do HEVC foi feita sempre buscando mesclar dois pontos importantes:

- Codificação de vídeo de altas resoluções;
- Exploração de paralelismos em grande escala, para melhor exploração da execução em hardware.

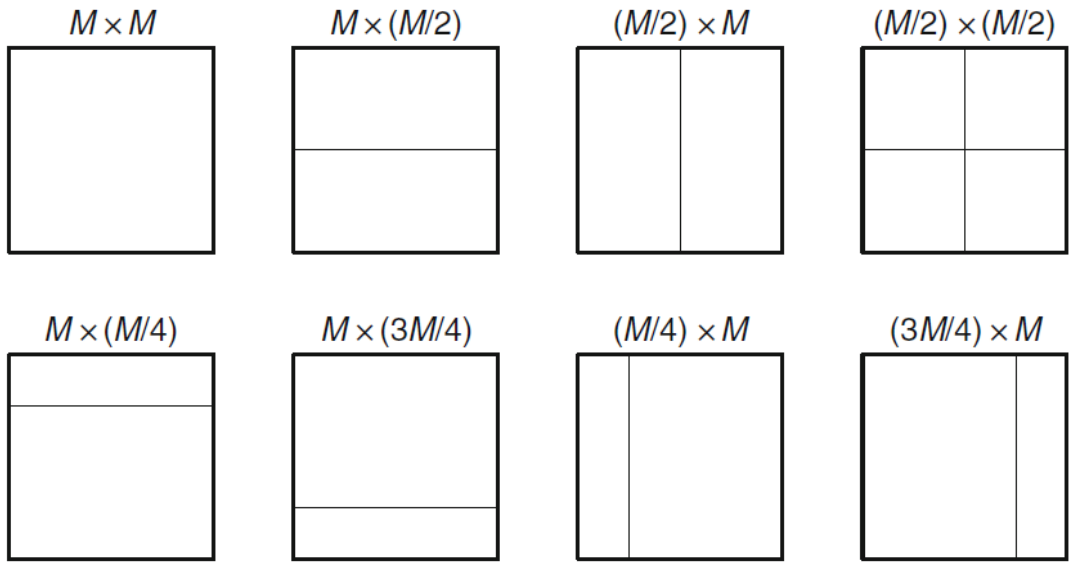
2.2.1 Principais Funcionalidades e Arquitetura do HEVC

No HEVC, a imagem a ser codificada é particionada para possibilitar a codificação de cada bloco com uma configuração especifica escolhida dentro de um conjunto de parâmetros pré-definidos, considerando-se que, em geral, um modelo único não consegue mapear eficientemente as propriedades de uma imagem completa (OLIVEIRA, ALENCAR, 2014).

No padrão HEVC, a imagem a ser codificada é dividida em *Coding Tree Units* (CTU), essas compostas por um *Coding Tree Block* (CTB) de luminância, e dois CTBs de crominância, onde o tamanho de cada CTU pode ser variável, sendo que as dimensões, LxL, podem ser de 32, ou 64 amostras de *pixels*.

Cada CTU pode se subdivida em outras pequenas unidades de codificação, chamadas de CU's (*Coding Units*), que podem ser tão grandes como a própria CTU, ou podem ser divididas de forma recursiva em quatro partes de tamanhos iguais, que por sua vez, ainda podem ser divididas em pequenas Unidades de Predição, as PU´s (*Prediction Units*) e em Unidade de Transformadas, as TU's (*Transform Units*), onde cada unidade de codificação pode conter uma ou mais PU´s, e cada um dos PU´s podem ser tão grandes quanto a sua CU (MOREIRA, 2014). A figura 5 ilustra as divisões citadas anteriormente.

Figura 5 - Divisão hierárquica dos CTU`s



Fonte: Sze et al. (2014)

Analisando a figura 5, observa-se que o tamanho dos blocos de predição no HEVC pode variar de 4x4 amostras até 64x64. Os grandes blocos de predição se tornam a forma mais eficiente para codificar grandes áreas lisas de imagem simples, do que blocos de predição menores. Isso se explica devido as maiores resoluções de vídeo atuais, onde existe a tendência de existirem áreas maiores com valores de *pixels* semelhantes. Transformadas podem ser usadas para alcançar a precisão em áreas menores compostas por detalhes mais complexos da imagem.

A tabela 1 apresenta a relação entre o padrão HEVC e seus antecessores quanto ao tamanho de blocos suportados.

Tabela 1 - Comparação de tamanhos de bloco de compensação de movimento, suportada em diferentes normas

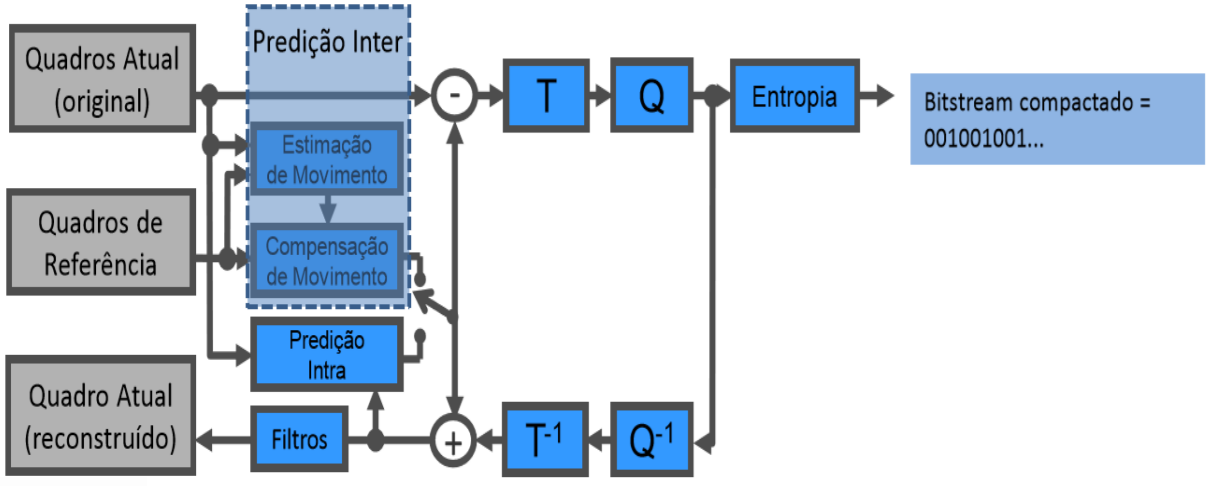
Padrões de codificação de vídeo	**Tamanho de blocos suportados**
H.262 j MPEG-2 Vídeo	16x16
H.263	16x16, 8x8
MPEG-4 Visual	16x16, 8x8
H.264 j MPEG-4 AVC	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4
HEVC	64x64, 64x48, 64x32, 64x16, 48x64, 32x64, 16x64,32x32, 32x24, 32x16, 32x8, 24x32, 16x32, 8x32,16x16, 16x12, 16x8, 16x4, 12x16, 8x16, 4x16,8x8, 8x4, 4x8

Fonte: Sze et al. (2014)

O chamado codificador HEVC compreende os principais blocos de um codificador e de um decodificador, bem como suas funcionalidades. O codificador HEVC é composto pelos blocos de predição intra-quadro, predição inter-quadros, transformadas diretas e inversas (T e TI), quantização direta e inversa (Q e QI), filtro redutor do efeito de bloco, filtro de deslocamento adaptativo de amostras, e codificação de entropia, conforme a figura 6.

Pode-se notar que o codificador possui quase um decodificador completo dentro de si, pois é necessário remontar as imagens já codificadas a serem usadas futuramente. Isso é necessário pois um dos processos envolvidos na codificação insere perdas e, portanto, para garantir que a mesma imagem de referência seja usada tanto do lado do codificador quanto do decodificador que esse processo de decodificação é necessário.

Figura 6 - Diagrama de bloco do codificador HEVC



Fonte: Próprio autor

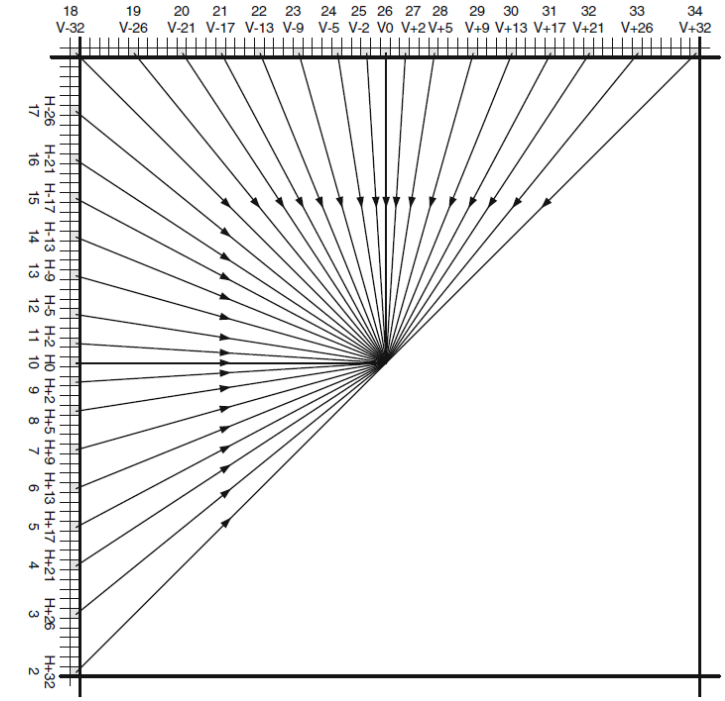
2.2.1.1 Intra-predição (IP)

A técnica, que explora a redundância espacial contida em uma mesma imagem do vídeo é a intra-predição (RAMOS, 2010). Presente no codificador e no decodificador, onde esta predição é gerada para cada bloco na estrutura do HEVC com base nas amostras de uma mesma imagem do vídeo a ser codificado.

Esta predição fornece ganhos de compressão quando a predição inter quadros não encontra um bom resultado, ou seja, não pode ser aplicada (DE OLIVEIRA; DE ALENCAR, 2014). Por exemplo, quando temos o primeiro quadro em uma sequência de vídeo, obrigatoriamente teremos que aplicar a intra-predição, visto que não temos ainda nenhum outro quadro de referência.

Outra especificação abrangida pelo HEVC é o maior número de modos de intra-predição quando comparado ao H.264/AVC, sendo incluindo modos para aplanar e aproximar uma superfície de *pixels* vizinhos, e um outro plano, com trinta e cinco modos de predição angulares (MOREIRA, 2014), conforme ilustrada pela figura 7.

Figura 7 - Ângulos da predição intra-predição em HEVC



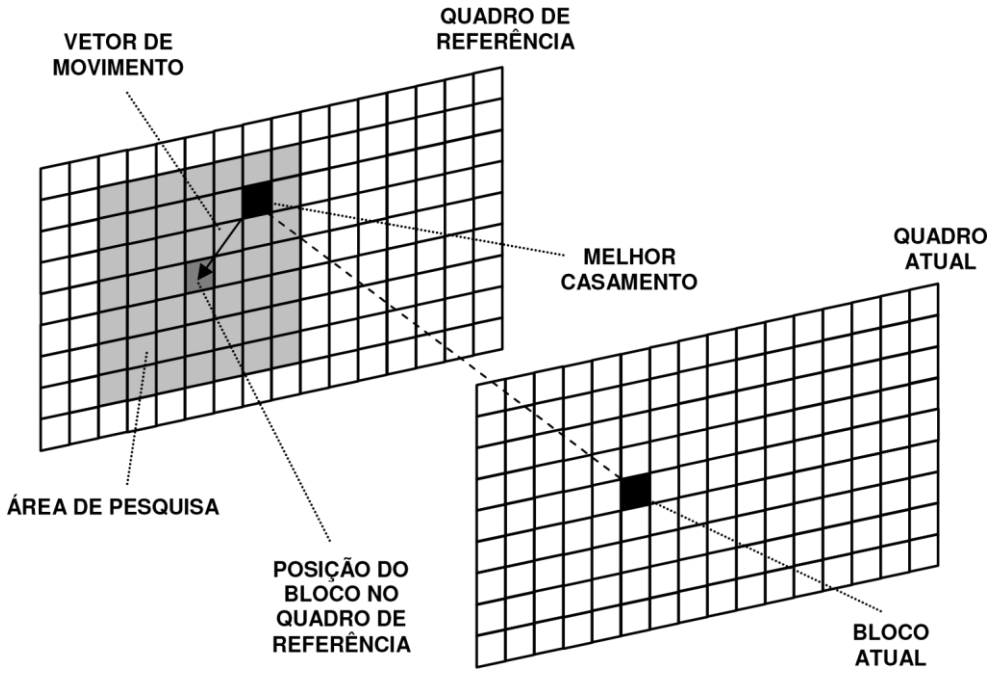
Fonte: Sze et al. (2014)

2.2.1.2 Estimação de Movimento (ME)

A predição inter-quadros é composta pelo bloco de Compensação de Movimento (MC) e pelo bloco de Estimação de Movimento (ME), sendo esse último presente apenas no codificador. Segundo (PURI, 2004) a predição inter-quadros é responsável pela maior complexidade computacional de um codificador. Contudo, a predição inter-quadro é aquela que fornece o maior potencial de ganhos de codificação (compressão).

A figura 8 a seguir ilustra o processo de estimação de movimento. O processo de estimação tem como objetivo localizar nos quadros de referência qual o bloco mais semelhante ao bloco do quadro atual, que está sendo processado. Segundo Diniz (2009) a busca geralmente é feita em uma área ao redor do bloco em questão, denominada área de pesquisa. Assim que o melhor casamento é encontrado, a ME gera um vetor de movimento que indica o deslocamento do bloco analisado, ou seja, a partir do quadro de referência, gera-se um vetor que indicará o deslocamento desse mesmo bloco para o quadro atual.

Figura 8 - Processo realizado pela ME

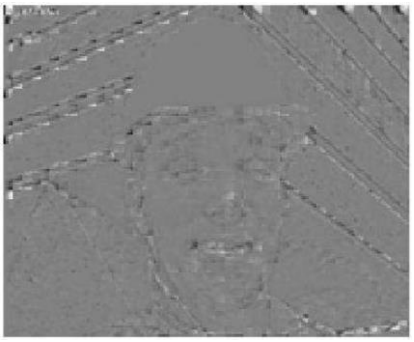


Fonte: Diniz (2009)

2.2.1.3 Compensação de movimento (MC)

Considerando dois quadros sucessivos de uma sequência de vídeo o quadro passado é usado como um preditor para o quadro atual. O quadro residual, ou resíduo, representado pela figura 9, é formado a partir da diferença entre o preditor e o quadro atual.

Figura 9 - Quadro de resíduos



Fonte: Ramos (2010)

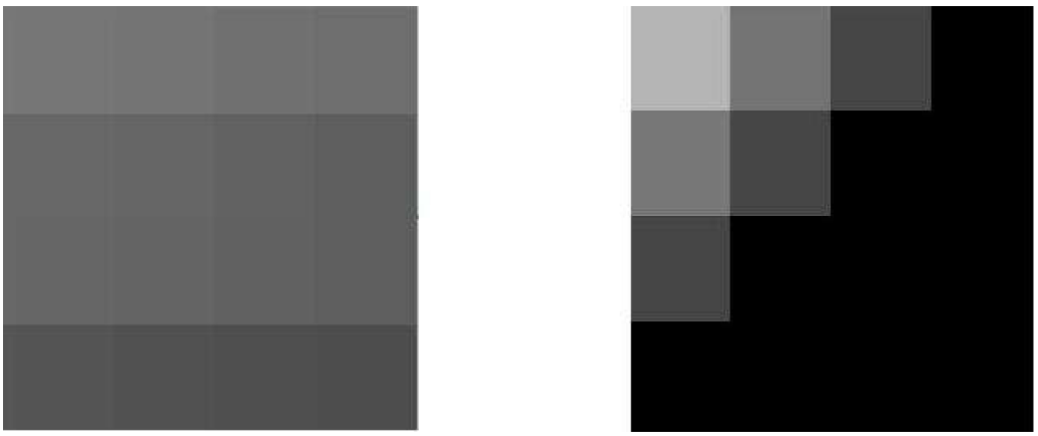
Em outras palavras, após a predição temporal (e mesmo na predição espacial), há ainda quantidade significativa de informações que podem ser exploradas para maior compressão, como já mencionado antes. É nesse contexto que entra em cena a MC, responsável pela operação de utilizar os resíduos, juntos com os dados fornecidos pela ME para remontar o quadro atual, a ser usado como futuro quadro de referência.

2.2.1.4 Transformadas Diretas (T)

A transformada direta é responsável por remanejar os blocos de resíduo de predição para o domínio da frequência e está, assim como a ME, presente apenas no codificador.

O processo de transformadas consiste em reorganizar os resíduos (diferenças entre os *pixels* preditos e os *pixels* codificados, como mostrados na figura 9) de uma forma que se torne potencialmente mais fácil de se compactar os dados processados, representando na figura 10, onde a figura 10a ilustra o bloco original e à figura 10b ilustra o mesmo bloco após o processo.

Figura 10 - Bloco ao sofrer processo de transformada



(a) (b)

Fonte: Ramos (2010)

2.2.1.5 Quantização Direta (Q)

A quantização direta está presente apenas no codificador, realizando uma quantização escalar ao resultado dos coeficientes processados pela transformada direta (i.e. uma divisão inteira sobre os resíduos das transformadas).

Essa etapa é a única na codificação que insere perdas (*lossy*) no processo, representada pela figura 11, onde, figura 11a ilustra o bloco após sofrer a transformada e figura 11b a quantização.

Figura 11 - Bloco ao sofrer processo de quantização

(a) (b)

Fonte: Ramos (2010)

2.2.1.6 Quantização Inversa (QI)

A quantização inversa é aplicada no codificador e no decodificador. No decodificador, ela é aplicada logo após a de decodificação de entropia. No codificador, ela é aplicada após a etapa de quantização direta.

Ela serve para corrigir a escala para o cálculo das transformadas, sendo aplicada após as transformadas.

2.2.1.7 Transformadas Inversas (TI)

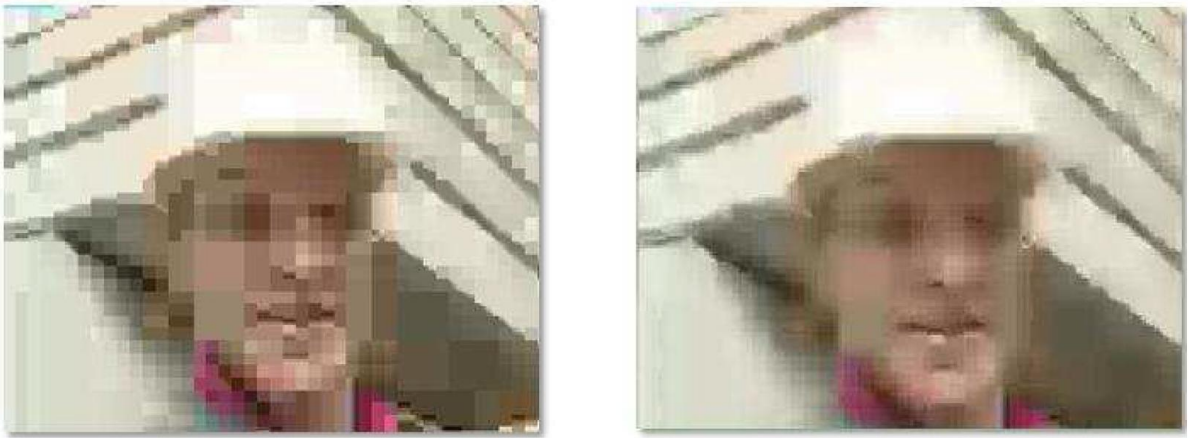
Já a transformada inversa é responsável pelas transformadas inversas dos coeficientes resultantes das etapas de transformadas T quantização Q diretas.

As operações das transformadas inversas são muito semelhantes às operações das transformadas diretas, apenas passando novamente os resíduos para o domínio de origem.

2.2.1.8 Filtro Redutor de Efeitos de Bloco (FREB)

Como o próprio nome sugere, FREB é responsável pela redução de efeito de bloco na imagem, ou seja, ele tem por objetivo deixar a imagem mais suave, reduzindo o efeito de bloco, representado pela figura 12, onde, a figura 12b apresenta a imagem ilustrada pela figura 12a ao passar pelo filtro.

Figura 12 - À esquerda imagem sofrendo o efeito de bloco



(a) (b)

Fonte: Ramos (2010)

2.2.1.9 Filtro de deslocamento adaptativo de amostras (SAO)

No padrão HEVC, diferentemente do H.264/AVC, tem se a aplicação de mais um filtro no processo de recuperação da imagem de referência, o *Sample Adaptive Offset* (SAO).

Essa nova técnica é aplicada dentro do processo de predição inter quadro, depois do filtro FREB, tendo como principal meta a reconstrução das amplitudes do sinal original, aplicando mapeamento não-linear de amplitudes, usando uma tabela de

consulta descrita por parâmetros adicionais que podem ser determinados numa análise do histograma no codificador (DE OLIVEIRA, 2014).

2.2.1.10 Codificação de Entropia

No HEVC, existe apenas um método de codificação de entropia empregado, o CABAC (*Context Adaptive Binary Arithmetic Coding*).

O bloco de codificação de entropia trabalha sobre os elementos sintáticos gerados nos processos anteriores da codificação (tamanho de CUs, tamanho de PUs, resíduos, etc.).

Ele serve para extrair estatísticas que correlacionam estes dados para representá-los com menor número de bits, isto é, elementos sintáticos mais frequentes tendem a ter uma representação usando menos bits, enquanto que os mais raros tendem a ter mais bits para representá-los.

3 INTERPOLAÇÃO PARA A COMPENSAÇÃO DE MOVIMENTO

Devido às justificativas já apresentadas anteriormente neste texto, a codificação de vídeo é um processo fundamental para tornar vídeos digitais viáveis, o que faz com que essa área de pesquisa seja adotada por vários pesquisadores, onde cada qual tenta contribuir com estudo e elaboração de trabalhos a fim de se obter arquiteturas cada vez mais eficientes quanto ao desempenho e ao consumo de potência.

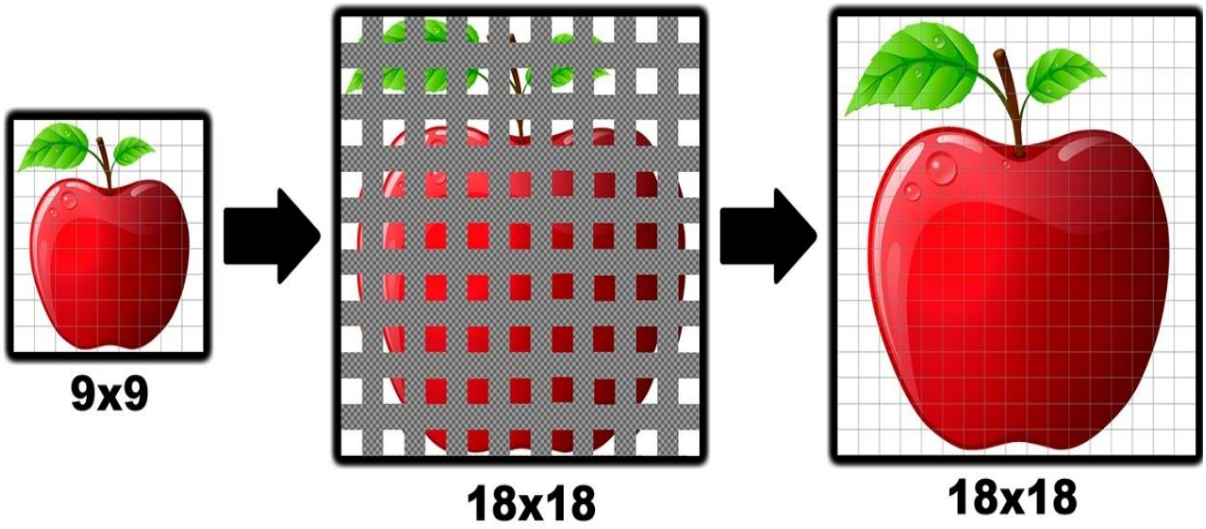
A seguir será apresentado o conceito de interpolação e como ela é empregada no padrão HEVC dentro do bloco de Compensação de Movimento (MC) para entendimento do trabalho aqui desenvolvido, sendo esse, mais uma contribuição para a área de pesquisa.

3.1 Filtro interpolador de *pixel*

Vamos ao conceito de interpolação, interpolar é predizer (ou estimar) o valor da variável sob estudo num ponto amostrado (LANDIM, 2002). Em outras palavras, a interpolação é uma técnica que permite reconstruir um conjunto de dados a partir de um conjunto de dados discretos.

A figura 13 ilustra o processo de ampliação de uma imagem, onde a figura 13a apresenta uma imagem de uma maçã que foi dividida em nove linhas e nove colunas, sendo esses uma ilustração dos *pixels* da mesma. A figura 13b ilustra a imagem após uma ampliação, onde as suas componentes horizontais e verticais foram dobradas, (contudo tal ampliação faz com que surja lacunas entre as representações dos *pixels*). Já a figura 13c mostra a imagem também após uma ampliação, porém sobre a mesma a técnica de interpolação de *pixels* foi aplicada, havendo assim a determinação dos *pixels* que ocupam as lacunas originadas da ampliação.

Figura 13 - Processo de ampliação de imagem



(a) (b) (c)

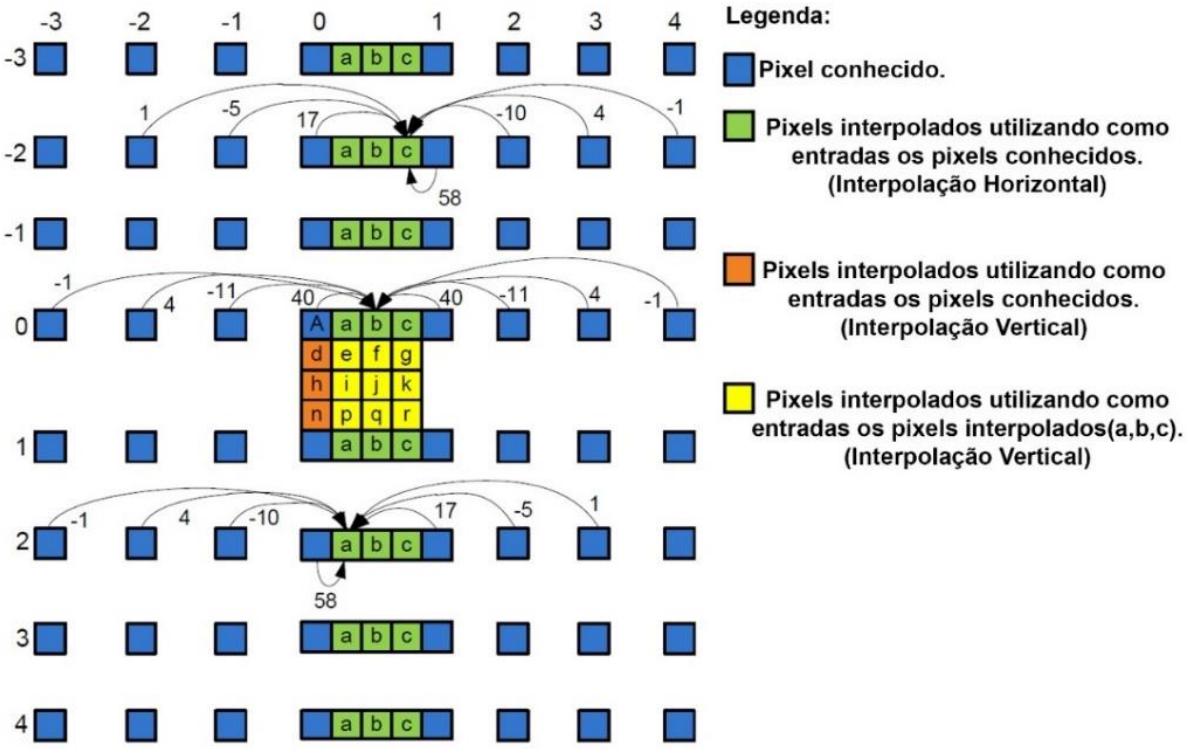
Fonte: Próprio autor

Visto a ilustração acima a interpolação de *pixel* é usada a fim de se reconstruir a imagem digital a partir de um conjunto discreto de *pixels*. Isso se dá devido à compressão que minimiza a representação de uma grande parcela de *pixels* em uma minoria que podem ser utilizadas para obtenção dos demais *pixels*.

Os quadrados verdes representam *pixels* obtidos a partir da interpolação horizontal com ½ de precisão de *pixel*, e os quadros de cores laranja e amarela representavam *pixels* obtidos a partir de uma interpolação vertical, onde os laranjas tem precisão de ½ de *pixel* e os amarelos tem precisão de ¼ de *pixel*.

Como apresentado na tabela 1 os blocos no padrão HEVC podem ter dimensões de 64x64. A figura 14 ilustra o funcionamento da interpolação de *pixels* no padrão HEVC num bloco 8x8 para luminância, onde os quadrados azuis representam *pixels* reais previamente conhecidos.

Figura 14 - Processo de interpolação de pixels



Fonte: Diniz (2015)

A precisão do *pixel* interpolado está diretamente relacionada ao conjunto de *pixels* utilizados como entrada para sua interpolação, onde ½ de precisão de *pixel* é denominado aos *pixels* originados de uma interpolação que use os *pixels* inteiros e ¼ de precisão de *pixel* é denominado aos *pixels* resultados de uma interpolação que use *pixels* interpolados anteriormente, ou seja, *pixels* com ½ de precisão.

A figura 14 acima apresentam três interpolações que utilizam coeficientes diferentes sendo a representada mais acima faz utilização

As equações (1-15) apresentam os cálculos efetuados no processo no qual os *pixels* previamente conhecidos “A” são utilizados na interpolação dos *pixels* resultantes da interpolação a, b, c, d, e, f, g, h, i, j, k, n, p, q, r.

$$a_{2x0} = -A_{2x-3} + 4A_{2x-2} - 10A_{2x-1} + 58A_{2x0} + 17A_{2x1} - 5A_{2x2} + A_{2x3} \quad (1)$$

$$b_{0x0} = -A_{0x-3} + 4A_{0x-2} - 11A_{0x-1} + 40A_{0x0} + 40A_{0x1} - 11A_{0x2} + 4A_{0x3} - A_{0x4} \quad (2)$$

$$c_{-2x0} = A_{-2x-2} - 5A_{-2x-1} + 17A_{-2x0} + 58A_{-2x1} - 10A_{-2x2} + 4A_{-2x3} - A_{-2x4} \quad (3)$$

$$d_{0x0} = -A_{-3x0} + 4A_{-2x0} - 10A_{-1x0} + 58A_{0x0} + 17A_{1x0} - 5A_{2x0} + A_{3x0} \quad (4)$$

$$e_{0x0} = -a_{-3x0} + 4a_{-2x0} - 10a_{-1x0} + 58a_{0x0} + 17a_{1x0} - 5a_{2x0} + a_{3x0} \quad (5)$$

$$f_{0x0} = -b_{-3x0} + 4b_{-2x0} - 10b_{-1x0} + 58b_{0x0} + 17b_{1x0} - 5b_{2x0} + b_{3x0} \quad (6)$$

$$g_{0x0} = -c_{-3x0} + 4c_{-2x0} - 10c_{-1x0} + 58c_{0x0} + 17c_{1x0} - 5c_{2x0} + c_{3x0} \quad (7)$$

$$h_{0x0} = -A_{-3x0} + 4A_{-2x0} - 11A_{-1x0} + 40A_{0x0} + 40A_{1x0} - 11A_{2x0} + 4A_{3x0} - A_{4x0} \quad (8)$$

$$i_{0x0} = -a_{-3x0} + 4a_{-2x0} - 11a_{-1x0} + 40a_{0x0} + 40a_{1x0} - 11a_{2x0} + 4a_{3x0} - a_{4x0} \quad (9)$$

$$j_{0x0} = -b_{-3x0} + 4b_{-2x0} - 11b_{-1x0} + 40b_{0x0} + 40b_{1x0} - 11b_{2x0} + 4b_{3x0} - b_{4x0} \quad (10)$$

$$k_{0x0} = -c_{-3x0} + 4c_{-2x0} - 11c_{-1x0} + 40c_{0x0} + 40c_{1x0} - 11c_{2x0} + 4c_{3x0} - c_{4x0} \quad (11)$$

$$n_{0x0} = A_{-2x0} - 5A_{-1x0} + 17A_{0x0} + 58A_{1x0} - 10A_{2x0} + 4A_{3x0} - A_{4x0} \quad (12)$$

$$p_{0x0} = a_{-2x0} - 5a_{-1x0} + 17a_{0x0} + 58a_{1x0} - 10a_{2x0} + 4a_{3x0} - a_{4x0} \quad (13)$$

$$q_{0x0} = b_{-2x0} - 5b_{-1x0} + 17b_{0x0} + 58b_{1x0} - 10b_{2x0} + 4b_{3x0} - b_{4x0} \quad (14)$$

$$r_{0x0} = c_{-2x0} - 5c_{-1x0} + 17c_{0x0} + 58c_{1x0} - 10c_{2x0} + 4c_{3x0} - c_{4x0} \quad (15)$$

Como mencionado antes, coeficientes são utilizados para interpolação, sendo esses determinados pelas normas do HEVC, a tabela 2 apresenta os coeficientes presentes no padrão HEVC, tanto para interpolação de Luma como pra Chroma.

As equações (1-15) podem ser divididas em três tipos, como a seguir, de acordo com os coeficientes utilizados na interpolação (i.e. os mesmos coeficientes utilizados):

- Equações (1) e (4-7);
- Equações (2), (8-11);
- Equações (3) e (12-15).

O presente trabalho está embasado na interpolação de Luma, sendo as equações (1-15) normalizadas no HEVC para esse tipo de interpolação, onde os coeficientes utilizados nas equações estão descritos na tabela 2 na parte correspondente a Luma.

Tabela 2 - Coeficientes da norma HEVC

	Tipo do filtro	**Coeficientes dos filtros**							
Luma	1/4	-1	4	-10	58	17	-5	1	0
	2/4	-1	4	-11	40	40	-11	4	-1
	3/4	0	1	-5	17	58	-10	4	-1
Chroma	1/8			-2	58	10	-2		
	2/8			-4	54	16	-2		
	3/8			-6	46	28	-4		
	4/8			-4	36	36	-4		
	5/8			-4	28	46	-6		
	6/8			-2	16	54	-4		
	7/8			-2	10	58	-2		
	Índice	-3	-2	-1	0	1	2	2	4

Fonte: Próprio autor

A seguir serão apresentados os trabalhos mais relevantes encontrados na literatura sobre o interpolador da MC para o padrão HEVC, sendo que alguns foram feitos baseados no padrão H.264/AVC, mas, devido à similaridade destes, algumas das propostas também se aplicam ao HEVC.

O trabalho de Diniz (2015) a seguir foi o primeiro encontrado na literatura, com uma técnica de substituir as multiplicações por shifts e somas (será melhor apresentado adiante nesse trabalho), tal técnica também foi utilizada no trabalho de Wang et al. (2014), tem-se esses dois trabalhos como as principais peças para elaboração, os demais trabalhos, Teixeira (2014) e Azevedo et al. (2007) foram trabalhos relevantes encontrados na literatura utilizados para estudo sobre a codificação de vídeo.

3.2 Trabalho de TEIXEIRA

O trabalho de Teixeira (2014) apresenta a implementação de um bloco de *Fractional Motion Estimation* (FME) em hardware. No trabalho citado se obteve melhorias relevantes se comparado ao estado da arte, sendo uma das melhorias abordadas a possibilidade de considerar o custo de cada vetor de movimento, o que permitiu uma melhoria na compressão.

Outra melhoria abordada foi a possibilidade de usar os filtros mais preciso para *pixels* que estejam nas proximidades do centro da PU, o que lhe permitiu a obtenção de vídeos de melhor qualidade. Contudo, se enfatiza o maior custo de implementação em termos de área de circuito e frequência de clock menor.

As ferramentas utilizadas por Teixeira (2014) para elaboração do trabalho foram as seguintes.

- Software Quartus II da Altera → Software utilizado para sintetizar a implementação.
- FPGA EP4CE15F17A7 da família Cyclone IV → Segundo Oliveira, a escolha desse dispositivo se deu por ser um chip de baixo custo, além de existir várias plataformas de desenvolvimento voltados para estudantes.
- Aria II GX EP2AGX45CU17C4 → No caso desse dispositivo, a escolha se deu por se tratar de um dispositivo adequado para ser usado em aplicações de alto desempenho.

Na síntese realizada para o FPGA Aria II citado, Teixeira (2014) alega que foram usados 7343 ALUTs das 36100 disponíveis, o que corresponde a 20% da área total, e a frequência de operação obtida foi de 79.88 MHz, além de apresentar 900 miliWatts de consumo de potência a 85 graus Celsius.

3.3 Trabalho de DINIZ et al

No trabalho de Diniz et al. (2015) é apresentado uma arquitetura de hardware reconfigurável para a filtragem de interpolação em alta codificação de vídeo eficiente.

Segundo Diniz a arquitetura desenvolvida se adapta ao tempo de execução de alterações do número de interpolação. Ainda segundo Diniz, com isso se obteve um

grande ganho quanto à eficiência energética, isso devido a reconfiguração empregada para estimação do número de interpolação em tempo de execução. O esquema de seleção feito em tempo de execução permite determinar a melhor versão a ser empregada a cada imagem.

Quantitativamente é apresentado um ganho de 57% quanto ao uso de recursos se comparados a outras arquiteturas de trabalhos que estejam no mesmo escopo do trabalho Diniz.

3.4 Trabalho de AZEVEDO et al

No trabalho de AZEVEDO et al. (2007), é feito apresentação do desenvolvimento de uma arquitetura para a compensação de movimento bi-preditiva para o decodificador do H.264/AVC. O projeto e desenvolvimento da arquitetura é caracterizado em uma hierarquia de memória no qual se busca a redução da largura de banda, além do número de ciclos necessários para acessar a memória. Isso é possível graças ao processamento paralelo das amostras de luminância e crominância, e também ao uso de *datapaths* simplificados no processamento das áreas com referências bi-preditivas.

3.5 Trabalho de WANG et al

No trabalho Wang *et al*. (2014) é apresentado uma arquitetura de compensação de movimento (MC) para decodificador de vídeo resolução de 7680x4320 (8K ou *Ultra High Definition* (UHD)) HEVC. O projeto é caracterizado pelo fato do perfil UHD aumentar significativamente a taxa de transferência e tráfego de memória. No trabalho é proposto três optimizações: banco de cache 2D paralelo no qual buscam reduzir o trafego de memória em 61,86%, propõem também mecanismo *Write-Through* (WTM) pipeline afim de se alcançar um desempenho livre de conflitos, diminuído 50% da área de memória, e por fim um interpolador altamente paralelo. Como resultado apresentaram uma arquitetura de 90nm, com custo de 103.6K em portas lógicas, 12kB de memória cache. A arquitetura desenvolvida suporta decodificar um vídeo com resolução de 7680x4320 a 30 fps em 280MHz.

A tabela abaixo apresenta de forma resumida os trabalhos apresentados acima.

Tabela 3 - Resumos dos trabalhos

Nome	Padrão	Bloco	Recursos	Frequência/Throughput
TEIXEIRA	HEVC	ME	7343 ALUTs	79.88 MHz
DINIZ et al	HEVC	FILTRO	1772 ALUTs	-
AZEVEDO et al	H.264/AVC	MC	-	-
WANG et al	HEVC	MC	-	280 MPixel/s

Fonte: Próprio autor

4 DESENVOLVIMENTO DA ARQUITETURA

Nesse capitulo será apresentado a implementação do interpolador para o bloco de compensação de movimento baseado no padrão HEVC, o desenvolvimento da arquitetura foi baseado no trabalho de (WANG et al., 2014) encontrado na literatura.

4.1 Arquitetura do interpolador

Analisando-se a figura 14 e as equações de (1-15) podemos notar algumas características relevantes, onde é possível observar que o interpolador faz uso de oito *pixels* como entrada para interpolação de um *pixel*.

Na figura 14, nas equações, algumas multiplicações não são ilustradas nem representadas devido ao valor do coeficiente corresponder a zero (filtro do tipo luma ¼ e ¾ apresentado na tabela 2) o que tira a necessidade de sua representação.

Outra característica ilustrada pela figura 14 é uso da interpolação utilizando-se alguns coeficientes na determinação dos *pixels* a, b e c. Nesse contexto podemos agrupar os coeficientes em três grupos A, B e C, utilizados para determinação dos *pixels* interpolados a, b e c respectivamente. Sendo que o A e C são os mesmos, apenas invertidos na ordem como apresentados na tabela 3.

Tabela 4 - Coeficientes dos tipos A, B e C

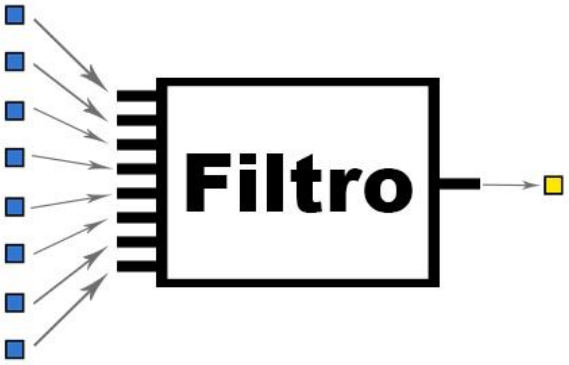
Filtro tipo A	-1	4	-10	58	17	-5	1	0
Filtro tipo B	-1	4	-11	40	40	-11	4	-1
Filtro tipo C	0	1	-5	17	58	-10	4	-1

Fonte: Próprio autor

Olhando com mais atenção para figura 14, podemos ver que os *pixels* interpolados d, h e n também fazem uso dos mesmos coeficientes utilizados pelos *pixels* a, b e c como pode ser visto nas equações (1-4), (8) e (12). A única diferença é que ao invés de usar *pixels* inteiros horizontais, eles vão utilizar *pixels* inteiros verticais.

Tendo isso em mente, se analisado de uma perspectiva onde se tem o filtro como uma caixa preta, ou seja, sabe-se que funciona, porém não como, o mesmo pode ser representado pela figura 15.

Figura 15 - Caixa preta do filtro



Fonte: Próprio autor

Onde os quadrados de azul representam os *pixels* de entrada para efetuação da interpolação, já o quadrado amarelo representa o *pixel* resultado da interpolação.

Agora suponhamos um contexto onde os *pixels* de entrada da figura 14 sejam nomeados de (A-H), e que esteja acontecendo uma interpolação do tipo B, teremos o seguinte cenário apresentado pela equação 16 onde temos os devidos coeficientes multiplicados pelas correspondes entradas.

$$PixelInterpolado = -A + 4B - 11C + 40D + 40E - 11F + 4G - H \quad (16)$$

Como dito antes e apresentado parcialmente na equação 16, o filtro faz multiplicações dos *pixels* de entrada por coeficientes, tal operação pode ser efetuada utilizando-se somas e shifts, sendo essas equivalentes a multiplicações. Contudo, tal operação apresenta melhor desempenho, uma vez que operações de soma e shifts são mais simples se comparadas com operações de multiplicação, sendo essa técnica herdada de Diniz (2015).

A seguir as equações (17 - 25) apresentam a manipulação efetuada sobre os coeficientes para o ganho de tal equivalência, onde cada coeficiente foi decomposto em fatores múltiplos de dois, afim de se fazer uso de somas e shifts. Considere uma entrada “X” (que é um *pixel*), onde a mesma é multiplicada pelos coeficientes tais como apresentado nas equações abaixo (17 - 25).

$$4 * X = X \ll 2 \quad (17)$$

$$-10 * X = -((X \ll 3) + (X \ll 1)) \quad (18)$$

$$58 * X = (X \ll 5) + (X \ll 4) + (X \ll 3) + (X \ll 1) \quad (19)$$

$$17 * X = (X \ll 4) + X \quad (20)$$

$$-5 * A = -((X \ll 2) + X) \quad (21)$$

$$1 * X = X \quad (22)$$

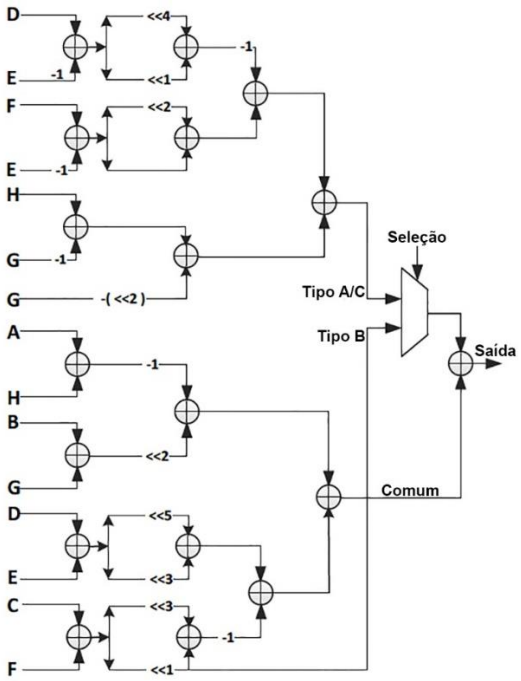
$$-11 * X = -((X \ll 3) + (X \ll 1) + X) \quad (23)$$

$$40 * X = (X \ll 5) + (X \ll 3) \quad (24)$$

$$-1 * X = -X \quad (25)$$

As equações acima apresentam as magnitudes dos coeficientes Luma apresentados na tabela 2 e suas equivalências em somas e shifts, e a figura 16 a seguir ilustra o circuito em baixo nível desenvolvido com utilização das equivalências em somas e shifts.

Figura 16 - Forma estrutura do filtro



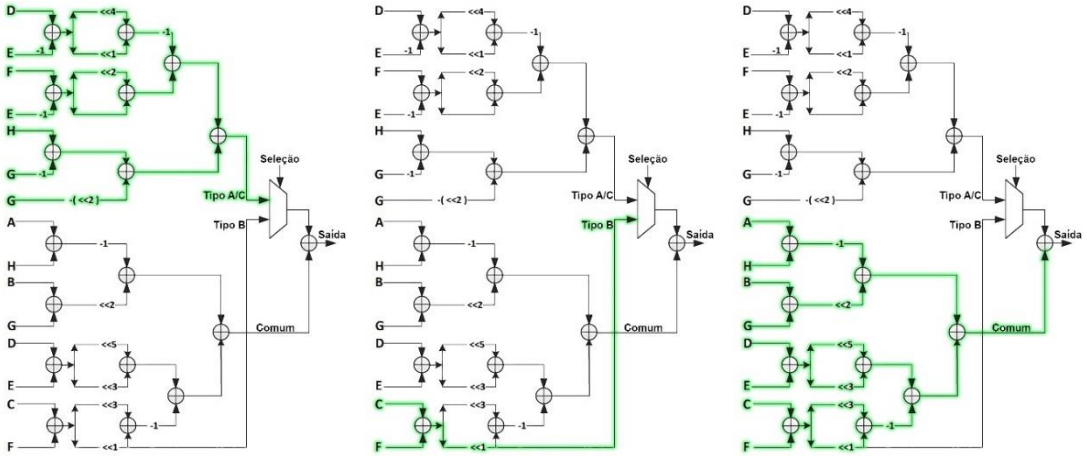
Fonte: Wang et al (2014)

O circuito acima é apto a fazer todos os cálculos de interpolação de luminância horizontal ou vertical, dentro do contexto do padrão HEVC. As letras A, B, C, D, E, F, G e H representam os *pixels* de entrada.

Na figura 17a, de forma destacada, observa-se uma parte como Tipo A/C, isso se deve ao fato mencionado anteriormente, onde abordamos que os filtros do tipo A e C são espelhados, evitando assim um *overhead*, ou seja, a requisição de hardware em excesso, visto que tal tarefa de executar uma interpolação do tipo C pode ser facilmente executada por um circuito que faz interpolação do tipo A, havendo a necessidade apenas de uma inversão nas entradas.

Na figura 17b, de forma destacada, é apresentado a parte do circuito responsável pela interpolação do tipo B; e na figura 17c é apresentado de forma destacada parte do circuito intitulado de comum por corresponder a parte do circuito comum tanto ao tipo A/C como ao tipo B.

Figura 17 - Partes do filtro



(a) (b) (c)

Fonte: Próprio autor

Na figura 17, a Saída corresponde ao *pixel* resultado da interpolação, onde a parte Comum é somada ao valor selecionado pelo Mux, que pode alternar entre A/C e B de acordo com o valor do sinal de Seleção.

4.2 Proposta arquitetural para redução do consumo

Tal proposta surgiu mediante a uma característica das interpolações ao longo de uma sequência de vídeo. Em uma análise em cima de uma sequência de vídeo real, observou-se que a alternância entre os filtros era feita após um grande processamento de cada tipo.

Quando o filtro Tipo A/C está sendo utilizado, o mesmo é executado várias vezes até que se tem a mudança para o filtro do Tipo B que, por sua vez, também é executado várias vezes.

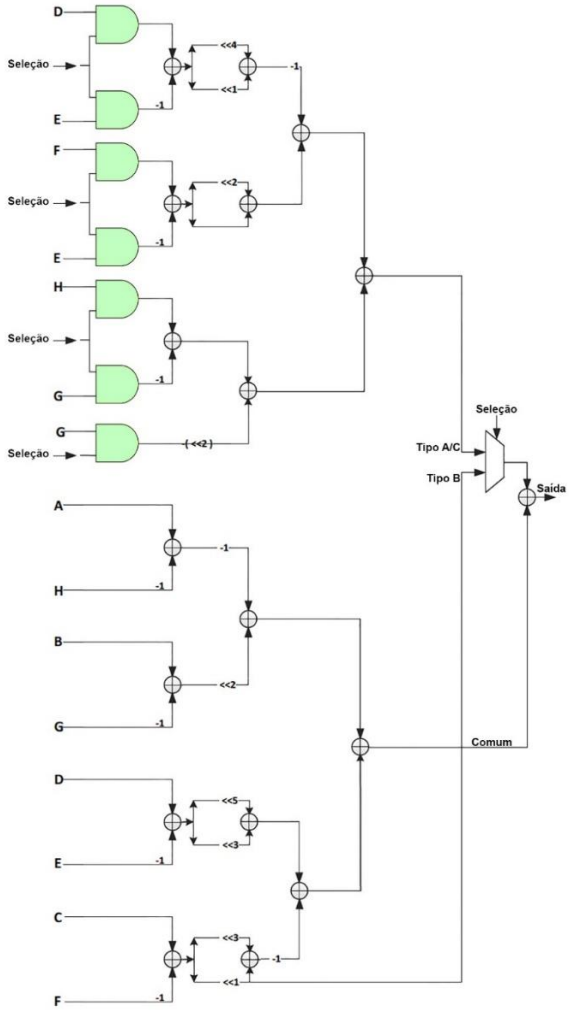
Sendo essa execução contigua de um tipo do filtro até que haja a mudança para o outro tipo, tem-se que quando o filtro do tipo B está executando, todos os cálculos feitos pelos somadores envolvido na lógica do tipo A/C são desconsiderados, como ilustrado antes na figura 16.

Sendo tais resultados desconsiderados, a execução desses somadores quando o filtro do tipo B está em execução é desnecessária. Nas quinze equações da norma de interpolação do HEVC, cinco são do tipo B, correspondendo a 33,33% do total de equações. Ambos fatores fazem com que esse ponto seja um ótimo alvo para se trabalhar uma proposta de diminuição no consumo de energia.

A técnica utilizada foi setar algumas das entradas para zero utilizando uma porta logica AND e um sinal de *enable,* sendo o sinal de *enable* o próprio sinal de Seleção utilizado no Mux, como ilustrado na figura 18, onde tal técnica é conhecida como *Operand Isolation* (CORREALE, 1995).

Esta estratégia irá permitir que não haja chaveamento devido as entradas estarem bloqueadas nos momentos em que aquela lógica não é necessária. Isso evitará consumo dinâmico de carga do capacitor nas saídas dos somadores que, sendo circuitos combinacionais grandes (somadores de *n* bits), vão tender a diminuir o consumo de energia da arquitetura como um todo.

Figura 18 - Forma estrutural do filtro com ANDs nas entradas



Fonte: Próprio autor

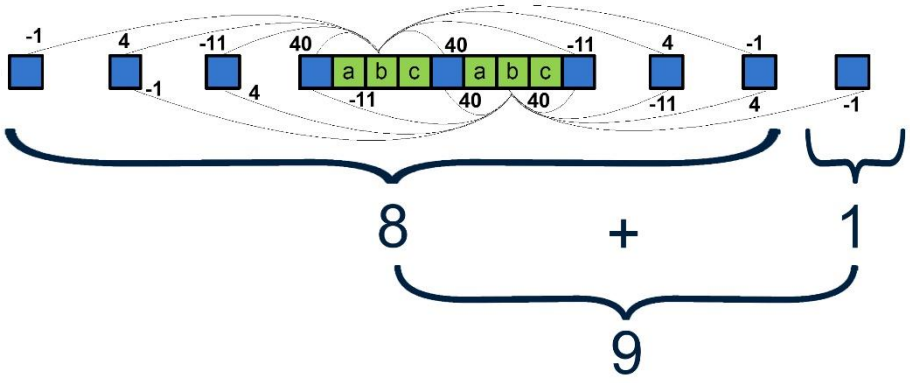
O sinal de controle para ligar/desligar as portas ANDs é o sinal de seleção do MUX na saída dos filtros, visto que é ele quem escolhe quando a filtragem A/C ou B vai ser a necessária e, portanto, ele quem vai decidir também quando determinado cálculo não precisará ser feito, de acordo com o tipo de filtragem necessária naquele momento.

4.3 Paralelismo na interpolação

Tendo o devido conhecimento mais aprofundado sobre o funcionamento do filtro de interpolação, podemos passar adiante, agora abordando a ideia de como seria possível a introdução de paralelismo na interpolação. Analisando-se a figura 19 abaixo, podemos analisar visualmente a ideia de paralelismo na interpolação

horizontal, onde temos duas interpolações usando coeficientes do filtro tipo B para a geração de *pixels* interpolados horizontais adjacentes.

Figura 19 - Interpolação paralela horizontal



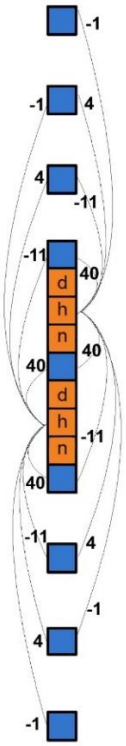
Fonte: Próprio autor

Observa-se que, para que uma segunda interpolação aconteça de forma simultânea, é agregado ao conjunto de *pixels* de entrada mais um *pixel*, sendo agora nossa entrada composta por nove *pixels* ao invés de oito. Relembrando os dados apresentado na tabela 1, onde é mostrado que os blocos no padrão HEVC podem ter dimensões de 64x64, sendo assim a figura 14 é apenas uma ilustração de um bloco 8x8.

O paralelismo não se limita apenas a duas interpolações simultâneas, pois o presente trabalho apresenta uma arquitetura onde oito filtros funcionaram de forma paralela. A arquitetura desenvolvida tem como entrada quinze *pixels,* oito *pixels* que seriam a entrada de um interpolador, mais a agregação de mais um *pixel* para cada interpolador associado adjacente.

A imagem 20 ilustra a ideia de paralelismo na interpolação vertical, onde também temos duas interpolações utilizando coeficientes do filtro tipo B para a geração de *pixels* interpolados verticais adjacentes.

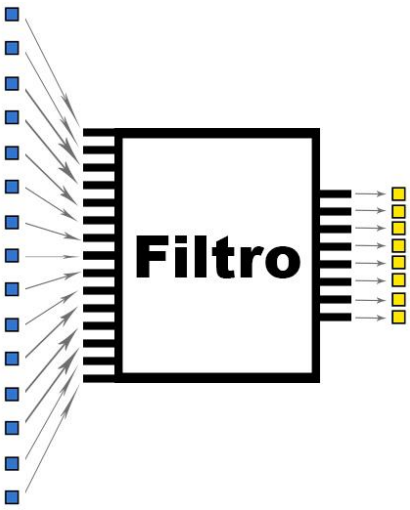
Figura 20 - Interpolação paralela vertical



Fonte: Próprio autor

Retomando o conceito de caixa preta, o componente desenvolvido é equivalente ao ilustrado pela figura 21, o mesmo apresenta um conjunto de 15 entradas e oito saídas, ou seja, a arquitetura é capaz de fazer oito interpolações simultâneas, sejam horizontais ou verticais.

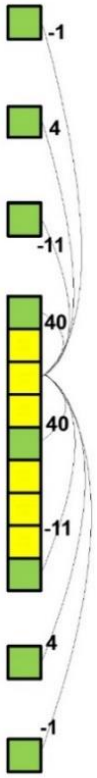
Figura 21 - Filtro interpolador paralelo



Fonte: Próprio autor

Visto então o paralelismo antes empregado na arquitetura, vamos a mais um paralelismo também agregado na arquitetura, analisando a figura 22 a seguir:

Figura 22 - Interpolação vertical de ¼ de precisão

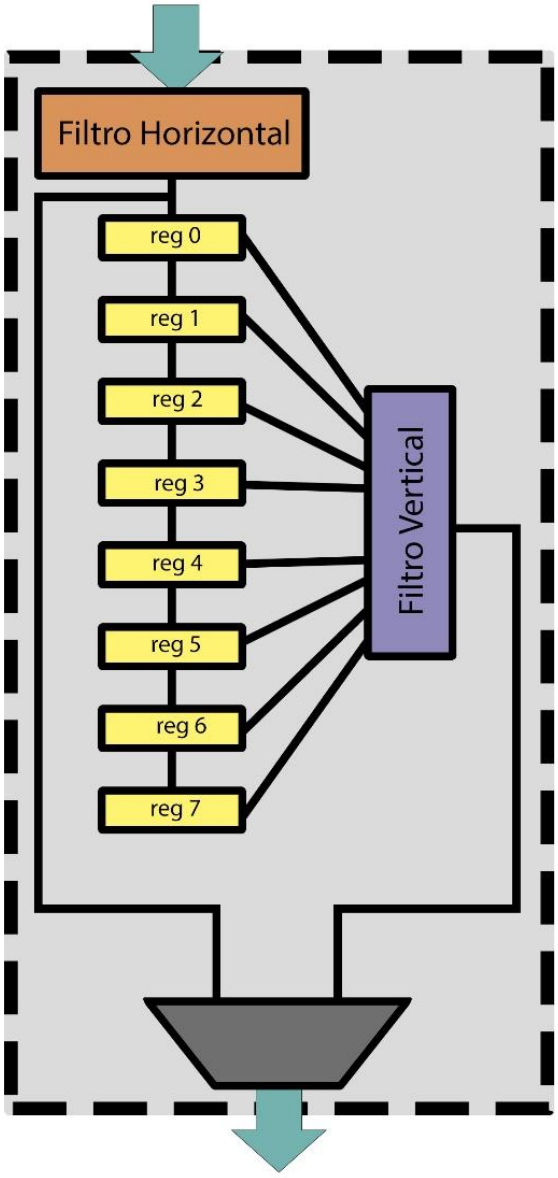


Fonte: Próprio autor

A figura 22 acima ilustra um conjunto de *pixels* que se diferem em alguns aspectos, ilustrados assim na figura por cores diferentes. Os *pixels* de verde são originados da interpolação horizontal (i.e. precisão de ½ de *pixel*), onde foram utilizados os *pixels* originais. Os *pixels* em amarelo são determinados com interpolação vertical utilizando os *pixels* originados das interpolações horizontais, como mostrado na figura 14.

Tendo isso em mente, consideremos o seguinte cenário, onde os *pixels* de verde que são determinados por interpolações horizontais já têm seus valores calculados. Nesse momento, os mesmos podem ser usados para interpolação vertical para calcular os *pixels* em amarelo. Isso torna possível que as interpolações horizontais e verticais ocorram de forma simultâneas. A figura 23 ilustra uma unidade de interpolação com dois filtros, um horizontal e outro vertical.

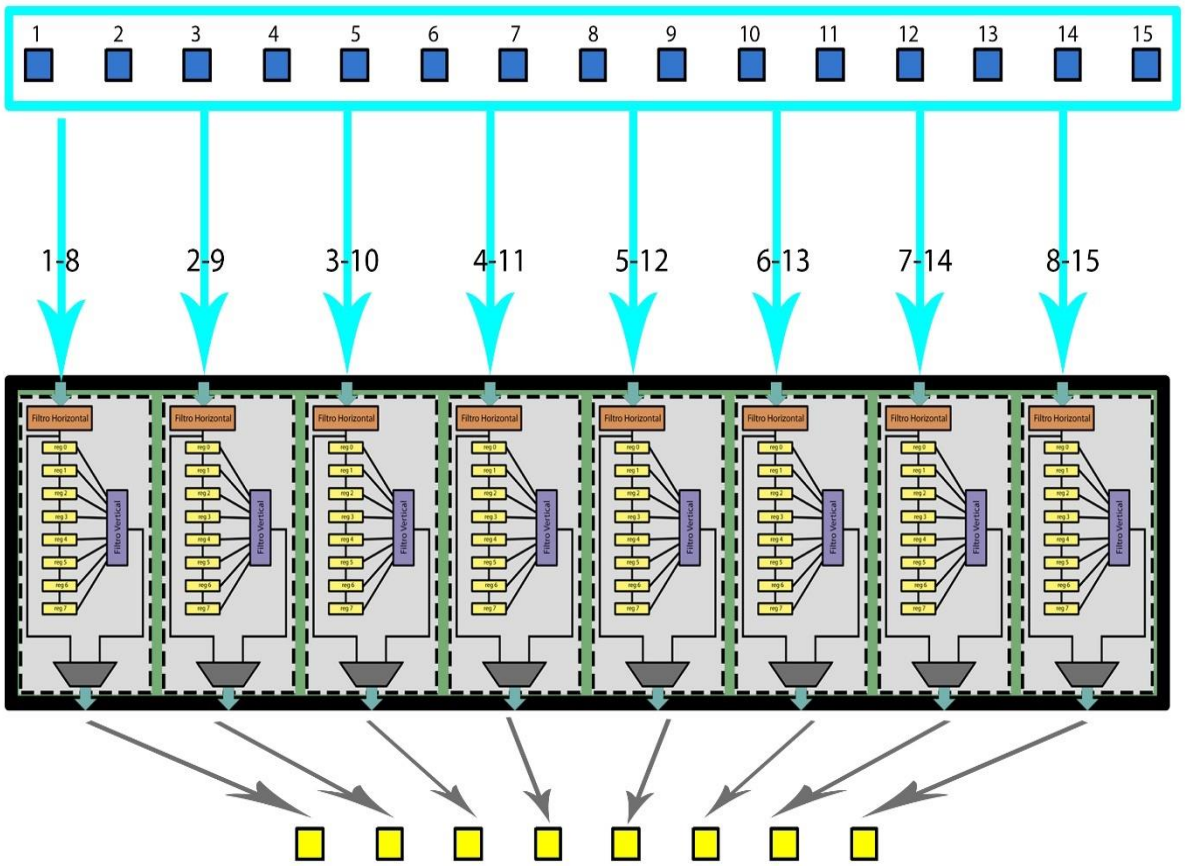
Figura 23 - Associação filtro horizontal e filtro vertical



Fonte: Próprio autor

A figura 23 acima apresenta um filtro vertical associado a um filtro horizontal, onde os *pixels* gerados pelo filtro horizontal são propagados por registradores a cada sinal de clock, o que posteriormente é usado como entrada para o filtro vertical, assim que tivermos o pipeline de registradores cheio (i.e. a partir do oitavo ciclo de clock). É importante destacar que o dito filtro horizontal acima pode fazer filtragens horizontais e também verticais, ambos com precisão de ½ de *pixel*.

Figura 24 - Arquitetura completa



Fonte: Próprio autor

A figura 24 acima apresenta mais detalhadamente a arquitetura desenvolvida, ou seja, a mesma apresenta os quinze *pixels* de entrada, as oito unidades de interpolação cada uma composta por um filtro horizontal e outro vertical, a imagem também ilustra os oito *pixels* gerados pela arquitetura.

A arquitetura implementada é capaz de interpolar oito *pixels* horizontais e verticais com ½ de precisão de *pixel* por ciclo de clock, assim como, no momento que tivermos pelo menos oito *pixels* horizontais de ½ de precisão guardados nos registradores de cada filtro (i.e. a partir do pipeline estar cheio), teremos também a possibilidade de interpolador oito *pixels* verticais com precisão de ¼ de *pixel* por ciclo de clock.

5. VALIDAÇÃO E RESULTADOS

Esse capítulo apresentará o processo empregado na validação da arquitetura desenvolvida. Também será apresentado resultados da síntese da arquitetura para FPGA.

5.1 Processo de validação

Com base na arquitetura apresentada anteriormente, agora será apresentado toda metodologia utilizada na validação da arquitetura.

O software de referência é a principal ferramenta no processo e validação, pois dele foram extraído os dados necessários para validação.

O software de referência está programado em Linguagem de programação C++. Encontra-se dividido em vários arquivos de programação, que juntos exercem a função do sucessor do H.264.

Levando se em consideração o diagrama de blocos do HEVC ilustrado pela figura 6, cabe salientar que o software de referência não possui mesma organização, ou seja, o código não está organizado de forma em módulos, onde cada módulo represente um bloco do diagrama.

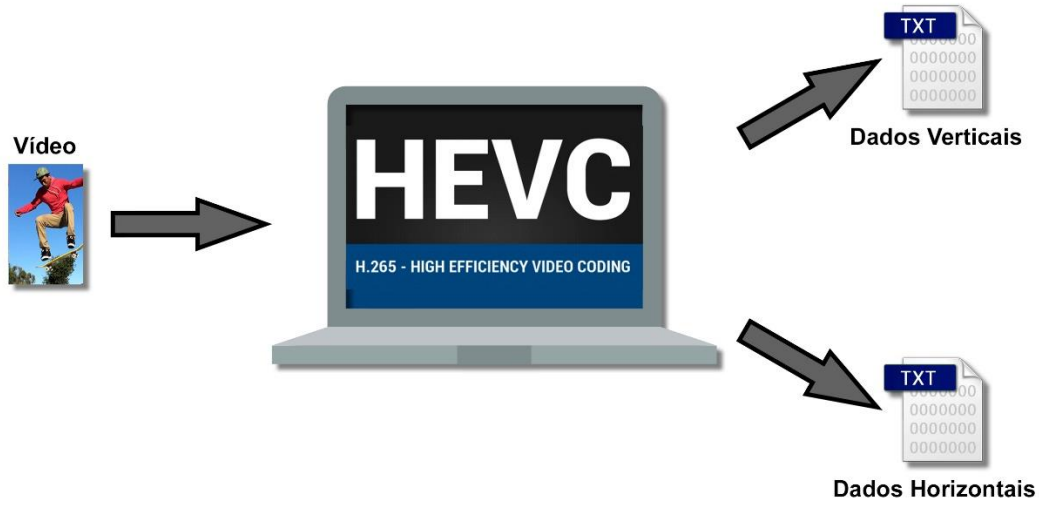
Nota-se no software de referência uma certa organização, onde podemos notar uma divisão dos códigos, de acordo com a funcionalidade estar presente somente no codificador, somente no decodificador, ou em ambos, conforme citado abaixo:

Codificador;

Decodificador;

Comum tanto ao codificador como ao decodificador.

Figura 25 - Ilustração da obtenção de dados



Fonte: Próprio autor

Num primeiro momento, o desafio foi identificar se o interpolador pertencia ao codificador, decodificador ou comum aos dois. Como este bloco está presente em ambos, a próxima etapa foi identificar entre todos os arquivos que eram comuns ao codificador e ao decodificador qual era responsável pela função de interpolação.

Realizada a etapa anterior de identificação do trecho de código de interesse, foram efetuadas a inclusão de algumas rotinas no código, onde essas rotinas consistiam em manipulação de arquivos (utilizadas com a finalidade de se salvar as entradas e saídas do trecho de código).

Feita tal mudança, ao colocar o software de referência para executar utilizando o vídeo-exemplo, foram gerados dois arquivos de texto, um intitulado de Dados Verticais e outro de Dados Horizontais, arquivos gerados com as interpolações Verticais e Horizontais respectivamente.

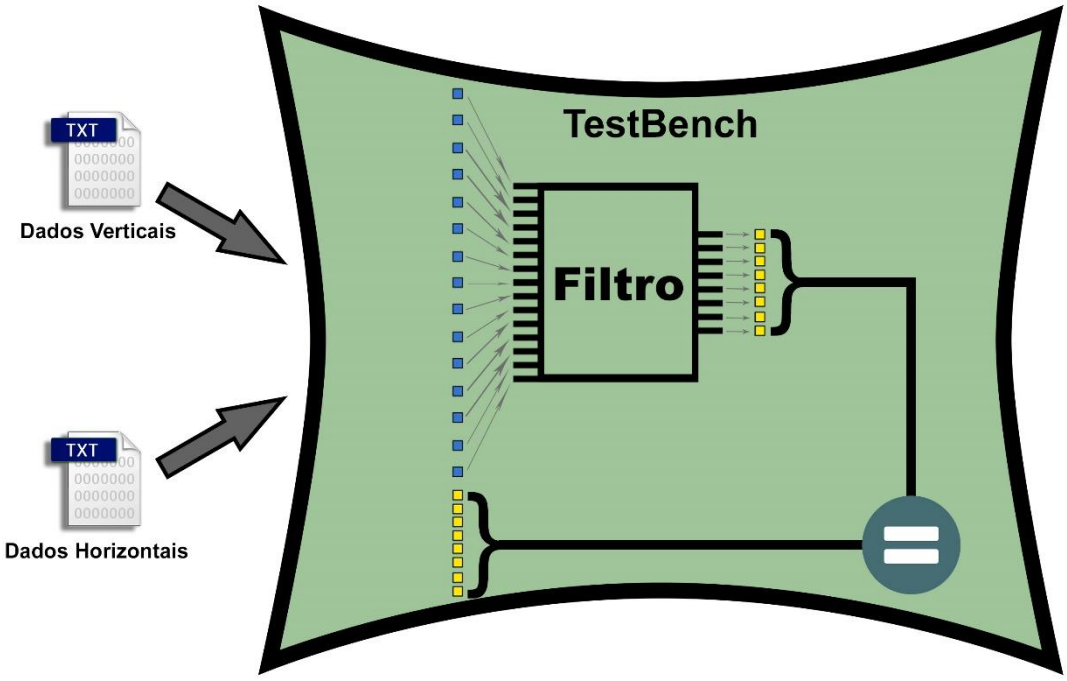
A figura 25 ilustra o processo de obtenção dos dados para validação, que consistiu em colocar o software de referência para executar, atribuindo a ele como entrada uma sequência de vídeo real com resolução de 1280x720 com taxa de 30 fps e, a partir disso, retirar os dados relevantes para validação.

Como dito anteriormente é impossível se conseguir um bom desempenho utilizando o HEVC em puro software, nota-se isso com mais claridade na tentativa de obtenção dos dados para validação, que foram limitados para serem extraídos apenas

sobre três *frames*, onde um arquivo texto de tamanho 6,24 GB contendo 1.110.916.652 entradas foi gerado.

Terminado o processo descrito anteriormente, a próxima etapa consistiu em alimentar a arquitetura com as entradas a fim de se obter as saídas referentes a cada entrada. O arquivo desenvolvido foi um TestBench em VHDL, esse que englobava a arquitetura desenvolvida.

Figura 26 - Esquemático da validação



Fonte: Próprio autor

A figura 26 anterior ilustra o TestBench desenvolvido, onde o mesmo engloba toda a arquitetura desenvolvida. O TestBench foi utilizado para fazer a leituras das entradas e saídas gerados pelo software de referência.

O TestBench usou as entradas lidas para alimentar a arquitetura. Após esse passo, o mesmo comparava as saídas lidas com as saídas geradas pela arquitetura. Realizada a execução e análise dado a dado, obteve-se sucesso em todas as comparações feitas, e com isso a arquitetura foi devidamente validada.

O processo de validação se contextualizou em manipulação de arquivo, no quais como mencionado as entradas e saídas do trecho de código alvo foram gravados.

Os arquivos tinham caráter textual, já os dados tinham representação decimal, o que proporcionou a comprovação da motivação da proposta de baixo consumo, uma vez que os arquivos texto podiam ser analisando visualmente.

5.2 Resultados para síntese FPGA

Ambas as arquiteturas desenvolvidas (sem e com a proposta para redução de consumo) foram sintetizadas para o FPGA da família Cyclone IV GX EP4CGX150DF31I7AD, que possui tecnologia de 60 nm.

Tabela 5 - Recursos utilizados no FPGA

Recursos utilizados	Filtro normal	Filtro modificado
Elementos lógicos	5230	5810
Funções combinacionais	4863	5604
Registradores	1083	1083
Pinos	242	242
Frequência máxima	104.49 MHz	102.64 MHz

Fonte: Próprio autor

A tabela 4 apresenta os recursos utilizados em cada arquitetura produzida, onde podemos observar um aumento nos Elementos lógicos e nas Funções combinacionais do filtro normal para o filtro modificado, isso se deve a introdução das ANDs nas entradas, o que também implicou na redução da frequência máxima alcançada, uma vez que as ANDs representam um aumento do caminho crítico do circuito.

Como apresentado anteriormente a arquitetura desenvolvida consegue entregar a cada ciclo oito *pixels* interpolados, isso devido ao paralelismo de oito interpoladores.

O filtro normal (sem a técnicas para redução do consumo) com sua frequência de 104,49 MHz obtido com sua sintetização para FPGA, se mostrou capaz de interpolar 835.920.000 *pixels* por segundo.

Já o filtro modificado (com a técnica de *Operand Isolation*) com sua frequência de 102,64 MHz obtido com sua síntese para FPGA, se mostrou capaz de interpolar

821.120.000 *pixels* por segundo. Tais valores foram obtidos com utilização das equações (26) e (27).

$$\text{Ps\_N} = 104{,}49 * 10^6 * 8 \rightarrow 835.920.000\ Pixels\ por\ segundo \quad (26)$$

$$\text{Ps\_M} = 102{,}64 * 10^6 * 8 \rightarrow 821.120.000\ Pixels\ por\ segundo \quad (27)$$

Com essa capacidade de processamento, foi feito o cálculo da quantidade de *frames* por segundo em resoluções de 1920x1080 (Full HD), resolução de 3280x2160 (4K) e resolução de 7680x4320 (8K), como apresentado na tabela 4.

Tabela 6 - Taxa alcançada utilizando arquitetura

Resoluções	**Pixels Horizontais x Vertical**	**Frames por segundo processado pela arquitetura**	
		Normal	**Modificada**
Full HD(HD ou 1080p)	1920x1080	403	395
4K(UHD ou 2160p)	3840x2160	100	98
8K(FUHD ou 4320p)	7680x4320	25	24

Fonte: Próprio autor

Os valores de frame por segundo em cada resolução são originados do uso das seguintes fórmulas (28) e (29) descritas a seguir.

$$pixels\ horizontais * \ pixels\ verticais = pixels\ por\ frame \quad (28)$$

$$\frac{pixels\ por\ frame}{pixels\ por\ segundo} = frames\ por\ segundo \quad (29)$$

Assim sendo, ambas as arquiteturas propostas alcançam desempenho de tempo real para vídeos Full HD e 4K HD (mais de 30 quadros por segundo). Considerando o caso de vídeos 8K HD, as arquiteturas propostas estão no liminar de alcançar o requisito, onde poderíamos agregar novos *cores* de filtragem para

aumentar o *throughput*; ou uma síntese para um FPGA com tecnologia menor; ou ainda a síntese para ASIC, onde uma frequência maior de clock pode ser alcançada.

5.3 Resultados consumo de potência

A tabela 6 apresenta a potência dinâmica de cada arquitetura, onde para analise justa, as mesmas foram simuladas com a mesma frequência de 71,42 MHz, durante o mesmo intervalo de tempo de 20 us, onde se obteve uma diminuição de 11,24% no consumo dinâmico de energia.

Tabela 7 - Potência dinâmica dos filtros

Frequência de teste	**Tempo de analise**	**Filtro**	**Potência dinâmica**
71,42 MHz	20 us	Normal	116,74
		Modificado	103,63
Porcentagem de diferença			11,24%

Fonte: Próprio autor

Embora mesmo o circuito do tipo A/C fique desligado 33,33% das execuções, obter tal ganho na diminuição do consumo de potência dinâmico mostra-se inviável apenas com tal técnica, isso devido ao fato do tipo A/C não necessariamente corresponder a 33,33% do circuito normal, além de que as ANDs agregadas também terão consumo dinâmico.

6 CONCLUSÃO

Este trabalho apresentou uma arquitetura de interpolador no contexto da MC para o padrão HEVC. Durante o processo de desenvolvimento, foram utilizados aprimoramentos presentes na literatura desde o surgimento do HEVC, como o uso de shifts e somas ao invés de multiplicações, que são operações ligeiramente mais complexas do que as anteriores.

Outro aprimoramento empregado foi o de paralelismo que busca a obtenção de um maior *throughput*, onde foi utilizado na arquitetura desenvolvida uma associação x8, onde oito interpoladores foram empregados em paralelo, capazes de realizar todas as interpolações de luminância, sejam horizontais e verticais de ½ de precisão de *pixel*, sejam as verticais de precisão de ¼ de *pixel*.

A arquitetura desenvolvida foi validada com a utilização de dados retirados do software de referência, onde os dados foram obtidos após identificação do trecho de código equivalente a arquitetura, a validação foi efetuada utilizando a versão mais recente do software de referência, HM-16.9.

Na validação foi utilizado um vídeo com resolução de 1280x720 com taxa de 30 fps, a obtenção dos dados para validação se deu sobre três *frames*, gerando um arquivo texto de tamanho 6,24 GB contendo 1.110.916.652 entradas.

Foram alcançadas taxas de processamento de 835.920.000 e 821.120.000 *pixels* por segundo, utilizando a arquitetura com o filtro normal e modificado, respectivamente.

As arquiteturas propostas alcançaram desempenho de tempo real para vídeos Full HD e 4K HD (mais de 30 quadros por segundo), onde novos *cores* de filtragem, ou uma síntese para um FPGA de nodo de tecnologia menor, ou ainda a síntese para ASIC, poderia contribuir para que o tempo real fosse alcançada para maiores resoluções.

Uma contribuição interessante para a literatura é a redução estimada no consumo de potência com a técnica de *Operand Isolation* que, por ser simples de ser empregada, pode se tornar uma grande melhoria nesse contexto, uma vez que tal circuitos pretende ser utilizado como parte de sistemas embarcados, onde o consumo de energia é um gargalo importante a ser considerado. A utilização dessa técnica na arquitetura foi comprovada ser eficiente no contexto do interpolador, onde apesar de

uma redução da frequência máxima de operação se obteve uma diminuição de 11,24% no consumo dinâmico de energia.

REFERÊNCIAS

AZEVEDO, A.; ZATT, B.; AGOSTINI, L.; BAMPI, S. **MoCHA: a Bi-Predictive Motion Compensation Hardware for H.264/AVC Decoder Targeting HDTV**. [S.l.]:IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2007. 4p

BARONCINI, Vittorio; OHM, Jens-Rainer; SULLIVAN, Gary. **Report of subjective test results of responses to the joint call for proposals (cfp) on video coding technology for high efficiency video coding (HEVC)**. ISO/IEC JTC1/SC29/WG11 MPEG2010 N, v. 11275, 2010.

BELTRÃO, Gabriel; ARTHUR, Rangel; IANO, Yuzo. **Introdução ao HEVC–High Efficiency Video Coding**. Campinas: Universidade Estadual de Campinas. 2011.

CISCO SYSTEMS. 2015. **Cisco Visual Networking Index: Forecast and Methodology, 2015-2020**. White Paper. May 27, 2015. Disponível em: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generationnetwork/white_paper_c11-481360.pdf. Acesso em: 13 jun 2016, 11:30:30.

CORREALE JR, Anthony. **Overview of the power minimization techniques employed in the IBM PowerPC 4xx embedded controllers**. Proceedings of the 1995 international symposium on Low power design. ACM, 1995. p. 75-80.

DA SILVA, André Marcelo Coelho. **Um Estudo Sobre o Padrão H. 264/AVC de Compressão de Vídeo**. 2007. Tese de Doutorado. UNIVERSIDADE CATÓLICA DE PELOTAS.

DE OLIVEIRA, Jean Felipe Fonseca; DE ALENCAR, Marcelo Sampaio. **Padrão HEVC–Novas Tecnologias para Aplicações de Elevadas Taxas de Compressão de Vídeo**. Revista de tecnologia da informação e comunicação, v. 4, n 2, outubro 2014. p 54-62

DE OLIVEIRA BETETTO, Luciana Aparecida**. MÉTODO PARA COMPRESSÃO DE IMAGENS DIGITAIS FUNDAMENTADO EM PROCEDIMENTOS DE HUFFMAN E WAVELETS**.Santa Catarina: Universidade Federal de Santa Catarina,2005.

DINIZ, Claudio Machado. **Arquitetura de hardware dedicada para a predição intra-quadro em codificadores do padrão H. 264/AVC de compressão de vídeo**. Porto Alegre: Universidade Federal do Rio Grande do Sul. 2009.

DINIZ, Claudio M. et al. **A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding. Computer-Aided Design of Integrated Circuits and Systems**, IEEE Transactions on, v. 34, n. 2, p. 238-251, 2015.

DINIZ, Claudio Machado. **Dedicated and reconfigurable hardware accelerators for high efficiency video coding standard**. Porto Alegre: Universidade Federal do Rio Grande do Sul. 2015.

GONZALES, R. C.; WOODS, R. E. **Processamento de imagens digitais**. Edgard Blücher Ltda. São Paulo, 527 p., 2000.

ITU-T and ISO/IEC, **High Efficiency Video Coding**, ITU-T Recommendation H.265 and ISO/IEC 23008-2, 2013.

LANDIM, PAULO M. BARBOSA; MONTEIRO, RUBENS CALDEIRA; CORSI, ALESSANDRA CRISTINA. **Introdução à confecção de mapas pelo software SURFER**. Rio Claro: UNESP, 2002.

MOREIRA, Tom Jones. **Conceitos e Fundamentos do HEVC/H. 265**, São Paulo. Revista Set. p 86. 2015.

POURAZAD, Mahsa T. et al. **HEVC: The new gold standard for video compression: How does HEVC compare with H. 264/AVC**. Consumer Electronics Magazine, IEEE, v. 1, n. 3, p. 36-46, 2012.

PURI, Atul; CHEN, Xuemin; LUTHRA, Ajay. **Video coding using the H. 264/MPEG-4 AVC compression standard. Signal processing: Image communication**, v. 19, n. 9, p. 793-849, 2004.

RAMOS, Fabio Luis Livi. **Arquitetura para o algoritmo CAVLC de codificação de entropia segundo o padrão H. 264/AVC**. Porto Alegre: Universidade Federal do Rio Grande do Sul. 2010.

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

ROSA, L. **Investigação sobre Algoritmos para a Estimação de Movimento na Compressão de Vídeos Digitais de Alta Definição: Uma Análise Quantitativa**.

Monografia (Bacharelado em Ciência da Computação) –Universidade federal de Pelotas, Pelotas, 2007.

DOS SANTOS, Tayline Tyene. **Construção de um Fantoma Físico para Controle de Qualidade de Imagens de Radiografia Digital**. Brazilian Journal of Radiation Sciences, v. 3, n. 1A, 2015.

SEIDEL, Ismael et al. **Análise do impacto de pel decimation na codificação de vídeos de alta resolução**. Florianópolis: Universidade Federal de Santa Catarina 2014.

SILVA, Nelson Tiago Lopes. **Redução da potência de uma interface de alta velocidade em tecnologia CMOS através de" clock gating**. Porto: Faculdade de Engenharia da Universidade do Porto. 2008.

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

SULLIVAN, Gary J. et al. **Overview of the high efficiency video coding (HEVC) standard. Circuits and Systems for Video Technology**, IEEE Transactions on, v. 22, n. 12, p. 1649-1668, 2012.

SUNNA, p. **Avc / h.264/avc – an advanced video coding system for sd and hd broadcasting**. European broadcasting union technical review, [s.l.], n. 302, apr 2005. Disponível em: <http://www.ebu.ch/en/technical/trev/trev_302-sunna.pdf>. Acesso em: 01 jun 2015, 18:30:30.

SZE, Vivienne; BUDAGAVI, Madhukar; SULLIVAN, Gary J. **High efficiency video coding (HEVC).** In: Integrated Circuit and Systems, Algorithms and Architectures. Springer, 2014. p. 1-375.

TEIXEIRA, Gabriel Diego. **Desenvolvimento de uma arquitetura de hardware de um estimador de vetores de movimento de precisão sub-pixel seguindo o padrão HEVC**. Tese de Doutorado. Porto Alegre: UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. 2014.

URRIZA, JOSÉ M. ET AL. **ECONOMIA DE ENERGIA EM DISPOSITIVOS MÓVEIS.VI WORKSHOP DE COMUNICAÇÃO SEM FIO E COMPUTAÇÃO MÓVEL**. 2004. P. 48-56.

ZATT, Bruno et al. **Validação De Uma Arquitetura para Compensação de Movimento Segundo o Padrão H. 264/AVC**. XII IBERCHIP WORKSHOP, Costa Rica. 2006.

ZATT, Bruno et al. **Motion compensation hardware accelerator architecture for h. 264/avc**. In: Advances in Image and Video Technology. Springer Berlin Heidelberg, 2007. p. 24-35.

WANG, Shihao; ZHOU, Dajiang; GOTO, Satoshi. **Motion compensation architecture for 8K UHDTV HEVC decoder**. 2014 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2014. p. 1-6.