UNIVERSIDADE FEDERAL DE PELOTAS

Centro de Desenvolvimento Tecnológico

Programa de Pós-Graduação em Computação

Dissertação

Algoritmos e Arquiteturas de Hardware para a Compressão de Quadros de Referência em Codificadores de Vídeo Digitais

Dieison Soares Silveira

Pelotas, 2015

Dieison Soares Silveira

Algoritmos e Arquiteturas de Hardware para a Compressão de Quadros de Referência em Codificadores de Vídeo Digitais

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

Orientador: Prof. Dr. Marcelo Schiavon Porto
Co-orientador: Prof. Dr. Luciano Volcan Agostini
Co-orientador: Prof. Dr. Bruno Zatt

Pelotas, 2015

Dados Internacionais de Catalogação na Publicação (CIP)

S587a Silveira, Dieison Soares

Algoritmos e arquiteturas de hardware para a compressão de quadros de referência em codificadores de vídeo digitais / Dieison Soares Silveira; Marcelo Schiavon Porto, orientador; Luciano Volcan Agostini e Bruno Zatt, coorientadores. - Pelotas, 2015.

110 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2015.

1. Codificação de vídeo. 2. Compressão de quadros de referência. 3. Redução da largura de banda de memória. 4. Projeto de hardware. I. Porto, Marcelo Schiavon, oriente. II. Agostini, Luciano Volcan, coorient. IV. Título.

CDD: 005

Catalogação na Fonte: Aline Herbstrith Batista CRB 10 / 1737

Biblioteca Campus Porto – UFPel

Banca examinadora:

.....

Prof. Dr. Marcelo Schiavon Porto (Orientador)

.....

Prof. Dr. Luciano Volcan Agostini (Co-Orientador)

.....

Prof. Dr. Bruno Zatt (Co-Orientador)

.....

Prof. Dr. José Luís Almada Güntzel

.....

Prof. Dr. Júlio Carlos Balzano de Mattos

.....

Dr. Guilherme Ribeiro Corrêa

AGRADECIMENTOS

Agradeço a minha amada esposa Cris, minha grande amiga e companheira inseparável nessa caminhada. Sou muito feliz por ter alguém como você ao meu lado me dando incentivo e motivação diariamente, sei que posso contar com você a qualquer momento. Obrigado por estar sempre presente e por fazer parte da minha vida. Como você mesma diz “sonho que se sonha só, é só um sonho, mas sonho que se sonha junto é realidade”. Te amo muito.

A minha família, meus pais e irmão, que mesmo a quilômetros de distância acompanharam minha trajetória, comemorando as minhas conquistas. Obrigado pelo incentivo constante para que eu continuasse nos momentos de adversidades, e pela compreensão nos meus momentos de ausência.

Aos meus orientadores, Marcelo Porto, Luciano Agostini e Bruno Zatt, que participaram ativamente na construção deste trabalho, estando sempre à disposição para conselhos, ensinamentos e por terem me oportunizado os primeiros passos na pesquisa. Obrigado pela confiança, amizade, ajuda e paciência durante a orientação. Espero que durante meu doutorado possamos continuar trabalhando juntos e que esta parceria continue dando frutos.

Aos colegas do grupo GACI pelos momentos de descontração e de trabalho. A todos os colegas do grupo de codificação de vídeo, em especial ao Guilherme Povala e a Lívia Amaral, que tiveram contribuição direta no desenvolvimento deste trabalho. Fica meu muito obrigado, e sei que nossa parceria vai além desse trabalho.

Aos órgãos de fomento à pesquisa, CAPES, FAPERGS e CNPq, pela concessão da bolsa de pesquisa, a qual me oportunizou trabalhar com dedicação nesse trabalho. Além de contribuírem de forma significativa na minha formação, apoiando a participação em eventos através de projetos dos meus orientadores. Os resultados desse trabalho só puderam ser publicados com o auxílio desses órgãos. Ao PPGC da UFPel, seus professores e funcionários, agradeço pela atenção e disponibilidade. Finalmente, agradeço ao povo brasileiro que financiou meus estudos desde o ensino fundamental. Que na minha vida profissional eu consiga retornar a sociedade todo o apoio investido.

Muito obrigado a todos!

“A mente que se abre a uma nova ideia

jamais voltará ao seu tamanho original”.

Albert Einstein

RESUMO

SILVEIRA, Dieison S. **Algoritmos e Arquiteturas de Hardware para a Compressão de Quadros de Referência em Codificadores de Vídeo Digitais**. 2015. 109f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas.

Os sistemas de codificação de vídeo atuais vêm exigindo uma largura de banda com a memória cada vez maior para codificar um único quadro do vídeo. Isso acontece principalmente devido ao grande aumento nas resoluções dos vídeos digitais, bem como às novas ferramentas de codificação utilizadas pelos codificadores. Entre os principais módulos dos codificadores de vídeo atuais, o módulo que mais acessa a memória é a Estimação de Movimento (ME). A ME exige uma grande largura de banda de memória, a qual é utilizada para ler e escrever os quadros de referência na memória. Esse processo acaba gerando um elevado consumo de energia, uma vez que os acessos à memória externa são as operações que exigem mais potência nos sistemas digitais atuais. Esse problema torna-se mais evidente quando dispositivos alimentados por bateria são considerados. Nesse sentido, este trabalho propõe soluções algorítmicas e arquiteturais para a compressão de quadros de referência antes de serem enviados à memória, desta forma, reduzindo os acessos à memória e a largura de banda de memória necessária durante o processo de ME. Neste trabalho foram desenvolvidas três soluções: o DRFC (*Differential Reference Frame Coder*), o DRFVLC (*Differential Reference Frame Variable-Length Coder*) e o DDRFVLC (*Double Differential Reference Frame Variable-Length Coder*). Todas essas soluções apresentam o mesmo fluxo de funcionamento, aplicando uma codificação diferencial sobre as amostras originais, seguida de codificação de entropia. A principal diferença entre elas está na quantidade de codificações diferenciais utilizadas e na abordagem utilizada para a codificação de entropia. As soluções desenvolvidas atingem altas taxas de compressão e consequentemente, de redução de largura de banda de memória. Essas soluções atingem uma taxa de compressão de 50% a 70%, sendo essa a maior taxa de compressão entre todos trabalhos estado da arte encontrados na literatura. Arquiteturas de hardware para os três algoritmos, incluindo os módulos codificador e decodificador, também foram desenvolvidas. As arquiteturas foram descritas em VHDL e sintetizadas para ASIC em standard cells. A síntese foi gerada para duas tecnologias, 180nm e 65nm, e para duas frequências de operação, 62,5MHz e 250MHz. Os resultados de síntese das arquiteturas mostraram que o DDRFVLC é a solução mais eficiente, dissipando uma potência de 1,13mW na codificação de vídeos HD 1080p e 3,25mW para vídeos UHD 4K. Este *overhead* é insignificante, uma vez que essa solução atinge uma redução de consumo de energia de 90,36mJ (65,14%) a partir da redução dos acessos à memória externa.

Palavras-chave: Codificação de vídeo; Compressão de quadros de referência; Redução da largura de banda de memória; Projeto de hardware.

ABSTRACT

SILVEIRA, Dieison S. **Algorithms and Hardware Architectures for Reference Frame Compression in Digital Video Encoders**. 109f. Dissertation (Master Degree in Computer Science) – Graduate Program in Computer Science, Center of Technological Development, Federal University of Pelotas, Pelotas.

Current video coding systems require a growing external memory bandwidth to encode a single video frame. This is happening because there is a large increase in the digital videos resolutions, as well as the new coding tools used by encoders. Among the main modules of the current video encoders, the module that performs more memory accesses is the Motion Estimation (ME). The ME requires a large memory bandwidth, which is mostly used for read and write the reference frames in the memory. This operation generates high energy consumption, since memory accesses are one of the main power demanding elements in current digital systems. This problem becomes more evident when battery-powered devices are considered. In this sense, this dissertation proposes algorithmic and architectural solutions for the reference frames compression before they are sent to memory. In this sense, this dissertation proposes algorithmic and architectural solutions for the reference frame compression before they are sent to memory. Thus, reducing the memory accesses and the memory bandwidth for the ME process. In this work, three solutions were developed: DRFC (Differential Reference Frame Coder), DRFVLC (Differential Reference Frame Variable-Length Coder), and the DDRFVLC (Double Differential Reference Frame Variable-Length Coder). All of these solutions follow the same operating flow, differential coding followed by entropy coding. The difference between them is the amount of differential coding used and the entropy coding. The solutions developed achieve high compression rates and high memory bandwidth reduction. These solutions achieve a compression ratio from 50% to 70%, outperforming any lossless reference frame compressor available in the current literature. The hardware architectures for the three algorithms, including the encoder and decoder modules, have been developed. These architectures were described in VHDL and synthesized for ASIC standard cells. The synthesis was done for technologies, 65nm and 180nm, and two operating frequencies, 62.5MHz and 250MHz. The architectures syntheses results shown that DDRFVLC is the most efficient solution, with a power dissipation of 1.13mW to encode HD 1080p videos, and 3.25mW to UHD 4K videos, which is a negligible overhead, since this solution reaches energy savings of 90.36mJ (65.14%) from the external memory access.

Keywords: Video coding; Reference frame compression; Memory bandwidth reduction; Hardware design.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACF	*Autocorrelation Function*
AFLR	*Autocorrelation-Based Frame Lossless Recompression*
APBT	*Adaptive Prefix Bit Truncation*
AVC	*Advanced Video Coding*
CABAC	*Context Adaptive Binary Arithmetic Coding*
CCT	Condições Comuns de Teste
CD	Codificação Diferencial
CD-H	Codificação Diferencial-Horizontal
CD-HI	Codificação Diferencial-Horizontal Inversa
CD-V	Codificação Diferencial-Vertical
CD-VI	Codificação Diferencial-Vertical Inversa
CDI	Codificação Diferencial Inversa
CIF	*Common Intermediate Format*
CTU	*Coding Tree Units*
CU	*Coding Unit*
DCT	*Discrete Cosine Transform*
DCD	Dupla Codificação Diferencial
DCDI	Dupla Codificação Diferencial Inversa
DDR	*Double Data Rate*
DDRFVLC	*Double Differential Reference Frame Variable-Length Coder*
DMMA	*Dual-Mode Memory Addressing*
DPB	*Decoded Image Buffer*
DPCM	*Differential Pulse Code Modulation*
DRAM	*Dynamic Random-Access Memory*
DRFC	*Differential Reference Frame Coder*
DRFVLC	*Differential Reference Frame Variable-Length Coder*
DWT	*Discrete Wavelet Transform*
HACP	*Hierarchical Average and Copy Prediction*
HCC	*Head Code Compression*

HD	*High-definition*
HEVC	*High Efficiency Video Coding*
HMD	*Hierarchical Minimum and Difference*
HM	*HEVC Model*
ITU-T	*International Telecommunication Union*
LCU	*Largest Coding Unit*
MC	*Motion Compensation*
MDA	*Multi-Mode DPCM Scanning and Averaging*
MDLR	*Multi-Directional Lossless Recompression*
MDP	*Multi-Direction Prediction*
ME	*Motion Estimation*
MMSQ	*Min-Max-Scalar-Quantization*
MPEG	*Moving Picture Experts Group*
MSE	*Mean Squared Error*
PD	*Pattern Decision*
PSNR	*Peak-to-Signal Noise Rate*
PU	*Prediction unit*
QP	*Quantization Parameter*
RAM	*Random Access Memory*
RGB	*Red Green Blue*
RGM	*Regrouping Modes*
SAD	*Sum of Absolute Differences*
SBT	*Significant Bit Truncation*
SBTVD	Sistema Brasileiro de TV Digital
SFL	*Semi-Fixed Length Coding*
SPIHT	*Set Partitioning in Hierarchical Trees*
UHDTV	*Ultra High Definition Television*
VCEG	*Video Coding Experts Group*
VGA	*Video Graphics Array*
VHDL	*VHSIC Hardware Description Language*
VLC	*Variable Length Coding*
YCbCr	*Luminance, Chrominance Blue, Chrominance Red*

SUMÁRIO

1 INTRODUÇÃO

Os dispositivos digitais apresentam uma evolução constante, onde produtos como *palm tops* tornaram-se rapidamente obsoletos, e mesmo *smartphones* e *tablets*, que são produtos com elevada capacidade computacional, têm um tempo de vida no mercado muito curto, sendo constantemente atualizados. Naturalmente, esta corrida tecnológica em ritmo acelerado afeta o mercado de bens de eletrônica de consumo, estabelecendo uma crescente demanda por dispositivos móveis mais baratos e mais poderosos, melhor qualidade em serviços de mídia – em especial os serviços de áudio/vídeo cuja qualidade é um diferencial importante para o consumidor final - entre outras características dos dispositivos de computação móvel pessoal.

As aplicações de vídeo digital são um exemplo típico desse cenário. O hábito de compartilhar informações através de texto em uma página da web está constantemente sendo substituída por conteúdo de vídeo em sites, como *Youtube*. Além disso, videoconferências também estão se tornando mais frequentes, à medida que a tecnologia de comunicação se torna mais barata e melhor, oferecendo maior largura de banda, além de vídeos estéreo ou até mesmo vídeo 3D, com múltiplas vistas. Para quantificar esta tendência, um artigo publicado pela Cisco mostra que até o final de 2018 haverá 1,4 dispositivos móveis conectados à internet por habitantes na Terra (CISCO, 2014). Além disso, mais de dois terços do tráfego mundial de dados em dispositivos móveis será de vídeos digitais até 2018. O tráfego de vídeo em dispositivos móveis aumentará 14 vezes entre 2013 e 2018, sendo responsável por 69% do tráfego total de dados em dispositivos móveis até o final do período da previsão (CISCO, 2014).

As principais desvantagens que advêm desta tendência de uso de vídeos de forma crescente é que, com o grande aumento no uso de vídeos digitais de resoluções cada vez maiores, aliado com a necessidade de *displays* de resoluções e taxas de quadros maiores, juntamente com o número de vistas em caso de vídeos 3D com múltiplas vistas, haverá um incremento muito grande na largura de banda

necessária para transmissão de vídeos, bem como uma grande demanda de capacidade de memória para armazená-los. Além disso, o crescente interesse em dispositivos móveis, alimentados por baterias e capazes de lidar com vídeos digitais, traz desafios de energia e processamento de grande porte. Este cenário reforça a necessidade de codificar de forma eficiente os vídeos digitais antes de armazená-los e/ou transmiti-los. Com isso, a codificação de vídeo representa um processo fundamental para tornar o suporte às aplicações multimídia uma tarefa possível.

Os codificadores de vídeo estado da arte geram um fluxo contínuo de dados em conformidade com determinados padrões. O padrão H.264/AVC (ITU-T, 2012) é o padrão dominante no mercado atual. No entanto, com o advento das Ultra-Alta Resoluções (UHD – *Ultra High Definition*), como 4K (3840x2160 pixels) e 8K (7680x4320 pixels), e à crescente demanda por vídeos digitais na internet motivaram o desenvolvimento de um novo padrão de codificação de vídeo mais eficiente que o padrão H.264/AVC. Neste sentido, um novo padrão foi desenvolvido, o *High Efficiency Video Coding* (HEVC) (ITU-T, 2013). O padrão HEVC prevê uma redução de 50% na taxa de bits, mantendo a mesma qualidade visual, quando comparado com o padrão H.264/AVC (OHM, SULLIVAN, *et al.*, 2012). Para atingir esta taxa de redução, o HEVC introduziu uma grande quantidade de inovações no processo de codificação do vídeo, por exemplo, novas estruturas de codificação, transformadas de tamanho variável, unidades de predição de maior tamanho (até 64x64 amostras), entre outros (SULLIVAN, OHM, *et al.*, 2012). No entanto, estas novas ferramentas incorreram em um aumento no esforço computacional do codificador de 10% a 500% (CORREA, ASSUNCAO, *et al.*, 2012), e mais de duas vezes no volume de dados que é acessado em memória, quando comparado com o codificador H.264/AVC. Isto ocorre principalmente devido ao aumento na largura de banda e no armazenamento dos quadros de referência utilizados na codificação do vídeo (SAMPAIO, SHAFIQUE, *et al.*, 2014). Esta grande quantidade de acessos à memória interna e externa do chip, além do aumento do tamanho da memória interna, levam a um elevado consumo de energia nos codificadores de vídeo (SAMPAIO, SHAFIQUE, *et al.*, 2014) (ZATT, SHAFIQUE, *et al.*, 2011).

1.1 Formulação do problema e discussões

A predição inter-quadros é a etapa de processamento mais complexa nos codificadores de vídeo atuais, a qual corresponde a mais de 80% do tempo de

processamento e de consumo de energia nos codificadores (SAMPAIO, SHAFIQUE, *et al.*, 2014). A Estimação de Movimento (ME – *Motion Estimation*) procura a melhor correspondência de um bloco do quadro atual em um conjunto de blocos presentes nos quadros de referência. A pesquisa pode ser executada em todo o quadro de referência, ou em uma área de busca restrita a uma região de maior interesse do quadro. Os quadros de referência são armazenados, tipicamente, na memória externa, enquanto as áreas de busca são armazenadas em memória interna. Devido ao gerenciamento de memória para buscar as amostras da área de busca na memória externa e o aumento no consumo de *leakage* das memórias internas, a ME torna-se o bloco de processamento que mais utiliza memória no codificador (ZATT, SHAFIQUE, *et al.*, 2011) (SHAFIQUE, ZATT, *et al.*, 2012). Como resultado, de 70% a 90% do consumo energético da ME nos codificadores de vídeos atuais, como H.264 e HEVC, é gasto no acesso à memórias internas e externas (*leakage* e dinâmico) (ZATT, SHAFIQUE, *et al.*, 2011) (SAMPAIO, SHAFIQUE, *et al.*, 2014). Quando a codificação em questão é feita para vídeos da classe 3D com múltiplas vistas, este panorama se agrava ainda mais. O codificador de vídeo 3D utiliza, além da predição inter-quadros, a predição inter-vistas. A predição inter-vistas funciona de maneira semelhante à predição inter-quadros, porém, os quadros utilizados na predição são de câmeras distintas, deste modo, eliminando as redundâncias que existem entre câmeras. Além disso, no padrão emergente 3D-HEVC, que é uma extensão do padrão HEVC, é utilizado o formato de dados *Multi-View plus Depth* (MVD) (MULLER, SCHWARZ, *et al.*, 2013). O MVD é composto por múltiplos quadros de texturas e mapas de profundidade associados a esses quadros de texturas, desta forma, aumentando ainda mais a quantidade de informação a ser codificada, uma vez que os mapas de profundidade são codificados pelas predições intra-quadro e inter-quadros.

Algumas soluções que visam minimizar os problemas ocasionados pela comunicação entre a memória externa e o núcleo de processamento são apresentadas na literatura. Estes trabalhos se baseiam em duas técnicas, que são: (1) estratégias de reutilização de dados utilizando memórias caches e *buffers*; e (2) compressão de quadros de referência, antes de serem armazenadas na memória externa.

A reutilização de dados através de uma memória cache interna integrada ao chip trabalha de modo a armazenar os dados que potencialmente serão mais

acessados pela unidade de processamento. Entretanto, além do emprego de uma memória local, o projetista deve, também, se preocupar em garantir a sua eficiência. Para tanto, técnicas como a antecipação da busca de dados e previsão de acessos são amplamente utilizadas.

As soluções que utilizam a compressão de quadros de referência requerem duas etapas a mais no processo de codificação, que são: (1) comprimir o quadro de referência antes de ele ser enviado à memória; e (2) descomprimir o quadro que está na memória externa quando ele for utilizado na codificação. Esta abordagem introduz ganhos de diferentes maneiras, diminuindo o número de acessos à memória, diminuindo o tamanho da memória externa ou reduzindo a largura de banda necessária na comunicação, gerando uma redução no consumo de energia do codificador. A compressão dos quadros de referência é dividida em dois grupos: compressão com perdas e compressão sem perdas de qualidade. A compressão com perdas geralmente utiliza técnicas simples de compressão, porém insere perda na qualidade do vídeo codificado. Já na compressão sem perdas a qualidade não é alterada.

Desta forma, se percebe o grande desafio relacionado à utilização de memórias de forma eficiente nos codificadores de vídeo, principalmente com foco na codificação de vídeos de alta definição em dispositivos móveis. O desenvolvimento de soluções para a redução de largura de banda e acessos à memória externa trazem questões como: analisar as características dos vídeos codificados, a fim de extrair seu comportamento; desenvolver soluções de baixa custo computacional; desenvolvimento de soluções que evitem perdas de informações nos quadros de referência; desenvolvimento de arquiteturas de hardware com alto desempenho e baixo consumo de energia; e desenvolver uma organização de memória externa que permita acesso aleatório aos dados codificados.

1.2 Objetivo do trabalho e organização da dissertação

Este trabalho consiste em explorar soluções algorítmicas e arquiteturais para a compressão de quadros de referência. Tais soluções devem atender a requisitos como: compressão sem perdas de qualidade, altas taxa de compressão, baixo custo computacional, alto desempenho, baixa latência e baixo consumo energético. Desta forma, essas soluções podem introduzir alta economia de consumo de energia em codificadores de vídeo, além de introduzir soluções inovadoras para a literatura.

Essa dissertação está organizada como segue: o Capítulo 2 apresenta as principais ferramentas de codificação de vídeo, bem como o fluxo de codificação de vídeo nos codificadores de vídeo atuais. Além disso, as questões relacionadas aos acessos à memória externa e os desafios que envolvem a comunicação entre memória externa e núcleo de processamento são abordadas; o Capítulo 3 traz os principais trabalhos encontrados na literatura que abordam as técnicas de redução da largura de banda de memória em codificadores de vídeo; o Capítulo 4 apresenta uma análise das distribuições das amostras em vídeos digitais, bem como o comportamento destas distribuições com a utilização da codificação diferencial; o Capítulo 5 apresenta os algoritmos para a compressão de quadros de referência sem perdas desenvolvidos nesse trabalho; o capítulo 6 mostra o desenvolvimento das arquiteturas de hardware para o compressor de quadros de referência e como é feito o acesso aleatório aos dados codificados na memória externa; o Capítulo 7 apresenta os resultados e comparativos de software e hardware das soluções desenvolvidas com os trabalhos relacionados e, por fim, o capítulo 8 conclui esta dissertação e apresenta possíveis trabalhos futuros.

2 CODIFICAÇÃO DE VÍDEO

Aplicações que manipulam vídeos digitais estão presentes hoje em diversos dispositivos, que vão desde aparelhos celulares a televisores digitais de alta definição, com os mais diversos propósitos tais como: entretenimento, educação, comunicação, segurança, entre outros. Esses vídeos digitais são compostos por imagens estáticas dispostas sequencialmente a uma determinada frequência de exibição. As imagens estáticas que compõem um vídeo digital são chamadas de quadros, os quais são compostos por pixels. O pixel é a menor informação que forma uma imagem digital. Para imagens coloridas, cada pixel possui três canais de informação, sendo que o tipo de informação apresentada em cada canal irá depender do espaço de cores utilizado (RICHARDSON, 2002). O espaço de cores mais utilizado para vídeos digitais é o YCbCr, o qual é composto por três elementos, sendo eles, luminância (Y), crominância azul (Cb) e crominância vermelha (Cr). Estes três elementos, quando combinados, podem formar milhares de cores distintas (RICHARDSON, 2002). Este formato facilita a subamostragem de cores, pois as informações de luminância (Y) e crominância (Cb e Cr) são completamente independentes. E o processamento individual dessas informações, é importante devido ao fato de que o sistema visual humano é mais sensível a informações de luminância do que crominância (RICHARDSON, 2002).

A subamostragem de pixel têm como principais formatos o 4:4:4, 4:2:2 e 4:2:0. O formato 4:4:4 consiste em quatro amostras de luminância para quatro amostras de crominância azul e quatro amostras de crominância vermelha. Neste caso, os elementos de crominância são preservados, logo, a subamostragem não é aplicada. No formato 4:2:2, para cada quatro amostras de luminância, há duas amostras de crominância azul e duas amostras de crominância vermelha, reduzindo em 50% o número de amostras de crominância. O formato 4:2:0 é composto por quatro amostras de luminância, uma amostra de crominância azul e uma amostra de crominância vermelha. Neste formato, a redução das amostras de crominância é de 75% (RICHARDSON, 2002). A subamostragem de pixel 4:2:0 é a mais utilizada nos

codificadores de vídeo, porque através da sua utilização se obtém uma alta taxa de redução de informações, sem perda de qualidade significante no vídeo codificado (RICHARDSON, 2010).

Outros fatores que influenciam a qualidade do vídeo são a taxa de amostragem e a resolução do vídeo. A resolução de um quadro é definida pela quantidade de pixels distribuídos na horizontal e na vertical. Existem muitos tamanhos de resolução, tais como CIF (*Common Intermediate Format*), com 352x288 pixels e o VGA (*Video Graphics Array*) com 640x480 pixels, sendo estes, formatos antigos e com poucas aplicações atualmente. As resoluções de maior interesse atualmente são o HD 720p (1280x720 pixels), o HD 1080p (1920x1080 pixels) e o UHD 4K com 3840x20160 pixels. A taxa de amostragem é o número de quadros exibidos durante um período de tempo: quanto maior a taxa de amostragem mais suave será a transição dos quadros, fazendo com que a percepção de movimento seja a mais real possível. A taxa de amostragem geralmente é de 24 a 30 quadros por segundo (RICHARDSON, 2002). Entretanto, esta taxa pode ser maior se o vídeo tiver uma resolução muito alta. Se a resolução e a taxa de amostragem aumentarem, maior será a demanda de processamento computacional do codificador de vídeo.

Como os vídeos digitais são utilizados para os mais diversos propósitos, eles necessitam ser armazenados e eventualmente transmitidos, e para que isso aconteça de forma eficiente, é necessária a utilização de um codificador de vídeo. A Equação 1 mostra a largura de banda (*LB*) necessária para transmitir uma sequência de vídeo não comprimido, considerando subamostragem de pixel 4:2:0. A largura de banda é a taxa de transferência do canal de comunicação, que existe entre a memória e as unidades de processamento.

$$LB = L * A * QPS * B * tx_{sub}\ [bits/s] \quad (1)$$

Na Equação 1, *L x A* é o número de amostras em um quadro, onde *L* representa a direção horizontal e *A* representa a direção vertical; *QPS* representa a quantidade de quadros por segundo da sequência de vídeo; *B* representa a quantidade de bits por amostra de luminância ou crominância (tipicamente 8 bits); tx_{sub} representa o fator de multiplicação para a subamostragem de pixel, sendo $tx_{sub} = 1{,}5$ para subamostragem 4:2:0, onde para a cada 4 amostras de luminância existem 2 amostras de crominância, uma para cada camada (RICHARDSON, 2010).

Um vídeo digital com resolução HD 1080p (1920x1080 pixels), comercialmente conhecida como Full HD, capturado a 30 quadros por segundo e subamostragem de pixel 4:2:0, necessitaria de uma taxa de 712 Mb/s para ser transmitido em tempo real. Uma hora de captura deste vídeo ocuparia aproximadamente 336 GB de espaço de armazenamento sem compressão. Desta forma, pode-se perceber que a manipulação de vídeos digitais sem compressão é uma tarefa inviável, especialmente para aplicações em tempo real e para dispositivos móveis com limitação de processamento e consumo de energia, devido ao poder computacional necessário para as ferramentas de exibição/transmissão e também devido ao espaço para o armazenamento.

Com a tecnologia atual, a manipulação adequada dos vídeos digitais torna-se um desafio. Este empenho em atingir altas taxas de compressão fez com que a pesquisa no meio acadêmico e na indústria aumentasse consideravelmente nos últimos anos. A expectativa é que a pesquisa nesse tema cresça cada vez mais, dado o aumento considerável nas resoluções dos televisores digitais, que hoje já chega a UHD 8K com 7680x4320 pixels. Além disso, estes dispositivos recomendam taxas de processamento de 60 a 120 quadros por segundo, para que se obtenha uma boa qualidade visual (NHK, 2012). Para que esta manipulação ocorra de forma adequada a codificação dos vídeos digitais seguem normas de padronização. Os padrões de codificação de vídeo seguem estas normas. Dentre os padrões atuais destacam-se o H.264/AVC, padrão mais utilizado no mercado, e o HEVC, padrão estado da arte lançado em 2013 (ITU-T, 2013). Nas próximas seções será apresentado um breve histórico dos padrões de codificação, além de uma ideia geral sobre o fluxo de codificação do padrão HEVC, que será importante para a compreensão do escopo deste trabalho.

2.1 Padrões de codificação de vídeo

A padronização de técnicas de codificação de vídeo digital é essencial para permitir que equipamentos de diferentes fabricantes se comuniquem. Deste modo, padrões de codificação de vídeo foram desenvolvidos. Os primeiros esforços para a padronização da codificação de vídeo digital datam do final da década de 80 e início da década de 90, com a criação do padrão H.261 pelo ITU-T (ISO/IEC, 1993). Este padrão foi o responsável por introduzir ferramentas como, estimação de movimento na direção temporal, transformada discreta do cosseno e quantização linear seguida

de codificação de entropia aplicada aos resíduos de predição (RICHARDSON, 2002). Estas técnicas continuam sendo utilizadas pelos padrões atuais.

Após o padrão H.261, surgiu o padrão MPEG-1 da ISO/IEC (ISO/IEC, 1993), seguido do padrão MPEG-2 da ISO/IEC, que também foi padronizado pela ITU-T como H.262 (ITU-T, 2000). O MPEG-2, ou H.262, ainda é amplamente utilizado pelos sistemas de televisão digital. Apesar do sucesso do padrão MPEG-2, o desenvolvimento de novos padrões continuou. O padrão H.263 (ITU-T, 2005) foi lançado e incorporou alguns avanços obtidos pelos padrões MPEG-1 e MPEG-2.

O bom resultado alcançado com o padrão MPEG-2/H.262 motivou os grupos MPEG (*Moving Picture Experts Group*) (ISO/IEC, 2012) e VCEG (*Video Coding Experts Group*) (ITU-T, 2012) a trabalharem em conjunto novamente. Em 1997, estes grupos uniram-se sob o nome JVT (*Joint Video Team*), para desenvolver um novo padrão capaz de atingir taxas de compressão bem superiores às do H.263. Este novo projeto foi intitulado de H.26L.

Em 2001, o grupo MPEG conclui o desenvolvimento do padrão MPEG-4 Parte 2 (ISO/IEC, 2004). A partir deste padrão e do H.26L, o grupo JVT, iniciou um novo projeto. Este novo padrão tinha como objetivo atingir 50% a mais de compressão para a mesma qualidade, quando comparado a padrões anteriores. A versão final deste padrão foi concluída em 2003, sendo intitulado de H.264 pelo grupo VCEG, e MPEG-4 Parte 10 - AVC (*Advanced Video Coding*) pelo grupo MPEG. Em um artigo de 2004, descrevendo o padrão (SULLIVAN, TOPIWALA e LUTHRA, 2004), o coordenador do JVT, Gary Sullivan, decidiu utilizar o nome H.264/AVC para se referir ao padrão de forma equilibrada entre os nomes dados ao padrão pela ITU-T e pela ISO. Desde então, este padrão tem sido foco de incessante pesquisa e desenvolvimento por grupos espalhados ao redor do mundo.

Nos últimos anos os vídeos digitais tiveram um aumento considerável nas suas resoluções e taxa de amostragens. Com isso, um novo padrão começou a ser desenvolvido a partir de 2010 pelo JCT-VC (*Joint Collaborative Team on Video Coding*), sendo este grupo formado por especialistas do VCEG e do MPEG. O *High Efficiency Video Coding* (HEVC) supriu as necessidades de codificação citadas anteriormente com o máximo de eficiência e, para isso, reuniu ferramentas de compressão especializadas para vídeos de altas resoluções atingindo altas taxas de processamento e o dobro da taxa de compressão, para a mesma qualidade do vídeo, quando comparado com o padrão H.264/AVC (OHM, SULLIVAN, *et al.*, 2012).

2.2 O Padrão HEVC

Esta seção apresenta uma breve introdução ao novo padrão de codificação de vídeo, o *High Efficiency Video Coding* (HEVC) (ITU-T, 2013), descrevendo seu fluxo de codificação e inovações, destacando os acessos à memória externa durante o processo da ME, discutindo o impacto disso no consumo energético do codificador.

2.2.1 Fluxo de codificação

O padrão HEVC adota o esquema híbrido de codificação em blocos já consagrado em padrões anteriores, baseado na predição com compensação de movimento, transformadas e codificação de entropia de alta eficiência (SULLIVAN, OHM, *et al.*, 2012). A Figura 1 apresenta o diagrama de blocos de um codificador HEVC.

O processo de codificação do HEVC é mais complexo do que o utilizado no H.264/AVC. Isto se deve ao fato de que o HEVC utiliza uma estrutura flexível de árvore quadrática (*quadtree*), o que permite o uso eficiente de blocos com grandes dimensões e formatos variados nas etapas de predição e transformadas (SULLIVAN, OHM, et al., 2012). O fluxo de codificação no HEVC ocorre da seguinte forma: o quadro do vídeo é primeiramente dividido em um conjunto de blocos de mesmo tamanho, chamados *coding tree units* (CTUs). Uma CTU consiste em um bloco de *M*x*M* amostras de luminância e dos dois blocos de crominância correspondentes, sendo *M* = 16, 32 ou 64 amostras. Os tamanhos maiores tipicamente atingem melhores taxas de compressão (SULLIVAN, OHM, *et al.*, 2012). As CTUs eram também chamadas de *treeblocks* em versões anteriores do padrão HEVC. Na versão atual, o tamanho máximo do bloco de luminância de uma CTU é de 64x64 amostras.

A CTU é composta por uma ou mais unidades básicas de codificação, chamada *Coding Units* (CUs). O conceito de CU permite a sua divisão recursiva em quatro blocos de tamanhos iguais a partir do CTU. Este processo de divisões recursivas acaba por formar uma estrutura de codificação em forma de árvore quadrática composta por blocos de CU, que podem ter dimensões que variam desde 8x8 amostras até o tamanho da própria CTU. A Figura 2 (a) apresenta quatro CTU com as suas respectivas *quadtree*. Nesta figura cada CTU (I, II, III, IV) tem um

tamanho de 64x64. As folhas da *quadtree* representam o tamanho escolhido para a CU. No exemplo em questão, o menor nível (8x8) ocorre na profundidade 4.

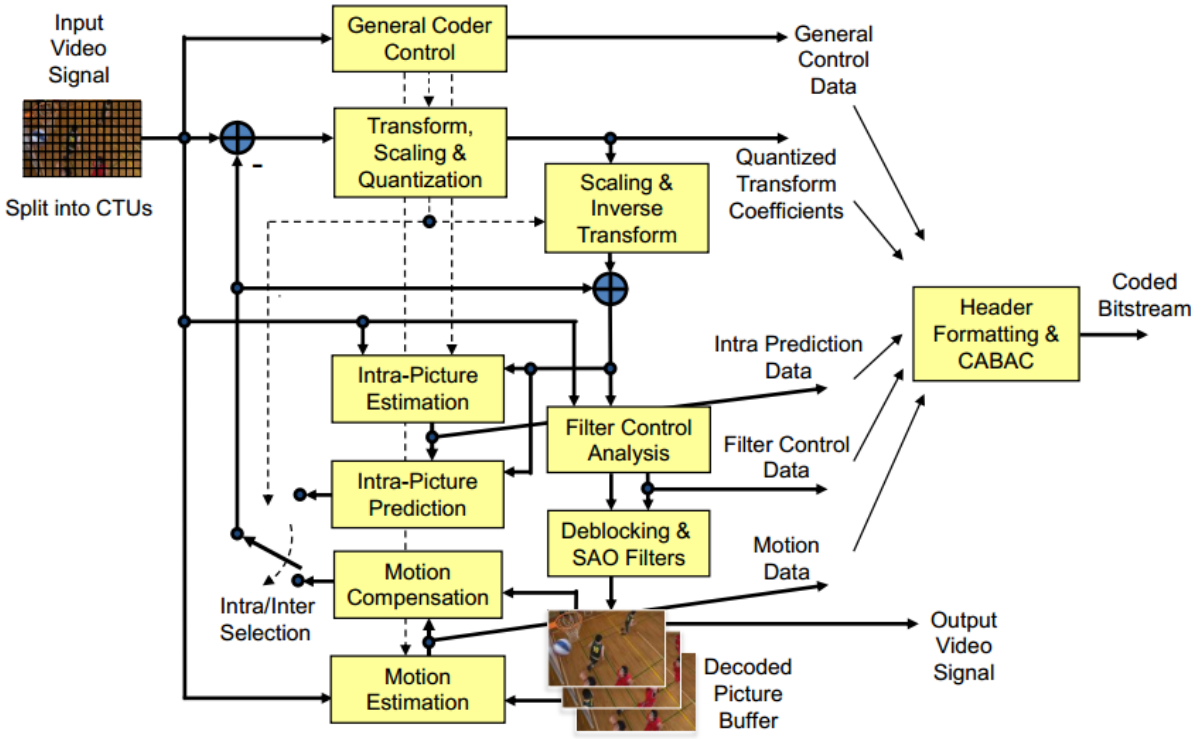


Figura 1 – Codificador HEVC (SULLIVAN, OHM, *et al.*, 2012)

O HEVC provê a utilização de *Prediction Units* (PUs), que são as unidades básicas que transportam as informações dos processos de predição (SULLIVAN, OHM, *et al.*, 2012). Cada CU pode conter uma ou mais PUs dependendo do modo de partição. Existem oito configurações de partições de PUs diferentes, quatro com formatos simétricos 2*N*x2*N*, 2*N*x*N*, *N*x2*N*, *N*x*N* e quatro de formato assimétrico 2*N*x*nU*, 2*N*x*nD*, *nL*x2*N* e *nR*x2*N*. Isso significa que uma CU com partição 2*N*x2*N* contém uma PU. Se a CU tiver partição *N*x*N*, ela terá 4 PUs. Nas demais configurações de partição a CU terá 2 PUs (SULLIVAN, OHM, *et al.*, 2012). Todos estes modos podem ser utilizados na predição inter-quadros. Para a predição intra-quadro só podem ser utilizadas PUs de tamanho 2*N*x2*N* e *N*x*N*. A Figura 2 (b) apresenta todas as possíveis representações das PUs, simétricas e assimétricas.

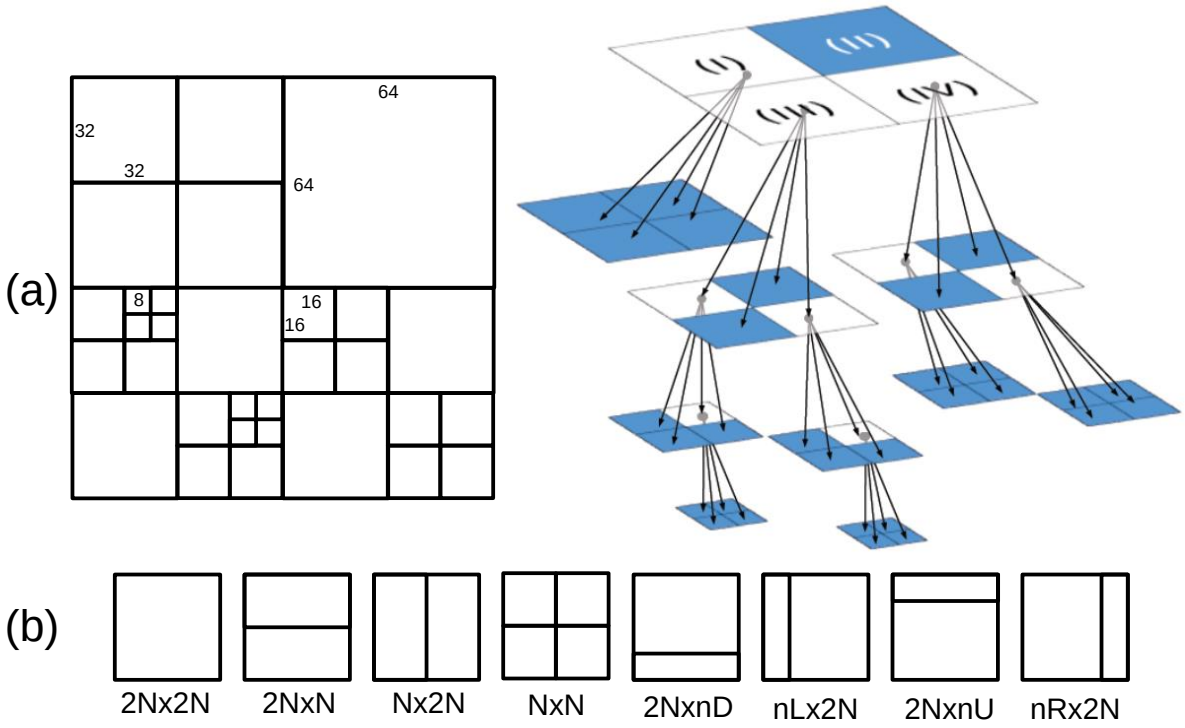


Figura 2 – (a) Quatro CTUs com suas respectivas *quadtree* e (b) divisões das PUs (CORREA, ASSUNCAO, *et al.*, 2014)

2.2.2 Etapa das predições

O processo de predição é o primeiro processo a ser executado por um codificador HEVC, sendo que a predição pode ser de três tipos: intra-quadro (I) e inter-quadros (P) ou (B). Quadros do tipo I são codificados utilizando apenas a predição intra, em quadros do tipo P e B podem ser aplicados tanto o processo de predição intra-quadro quanto no processo de predição inter-quadros. Nos quadros de tipo B cada CTU pode utilizar mais de um quadro como referência para a predição inter-quadros, enquanto nos quadros do tipo P é utilizado apenas um quadro como referência (RICHARDSON, 2010).

O módulo da predição intra-quadro do HEVC é o responsável por reduzir a redundância espacial, assim como é feito no H.264/AVC. No HEVC este módulo sofreu um grande aumento na sua complexidade, uma vez que apresenta 33 modos direcionais de predição intra-quadro mais os modos DC e planar, para os componentes de luminância de cada PU (SULLIVAN, OHM, *et al.*, 2012). No H.264/AVC são utilizados apenas nove modos direcionais para cada macrobloco (SULLIVAN, OHM, *et al.*, 2012). Este é um aumento considerável de complexidade,

pois cada um dos modos deve ser avaliado durante o processo de codificação para a definição do melhor modo de predição.

A predição inter-quadros contém, assim como no H.264/AVC, o módulo mais complexo do codificador de vídeo, a estimação de movimento (ME). A ME também é responsável por gerar os maiores ganhos de compressão no codificador, reduzindo as redundâncias temporais entre quadros vizinhos de uma cena. No HEVC a predição inter-quadros é utilizada em todas avaliações de CU durante o processamento das *quadtree*. Para cada CU na *quadtree*, a predição inter-quadros precisa executar a ME várias vezes, sendo uma vez para o tamanho da partição 2*N*x2*N*, duas vezes para a partição 2*N*x*N*, duas vezes para a *N*x2*N*, quatro vezes para avaliar a partição *N*x*N* e mais quatro vezes para os tamanhos assimétricos, em um total de treze operações de ME por CU (SULLIVAN, OHM, *et al.*, 2012). Todas estas avaliações para serem realizadas exigem um elevado número de cálculos, o que torna a ME o processo mais demorado na codificação de vídeo.

2.2.3 Transformadas e quantização

O HEVC utiliza uma unidade básica para o processo de transformada e quantização, chamada de *Transform Unit* (TU) (SULLIVAN, OHM, *et al.*, 2012). Cada CU pode conter uma ou mais TUs. O módulo de transformadas é responsável por converter as informações do domínio espacial para o domínio das frequências, para que assim, a quantização possa ser aplicada, reduzindo a redundância espacial presente nos resíduos de predição (RICHARDSON, 2010). A transformada principal (*core transform*) utilizada no HEVC é uma aproximação inteira da DCT (*Discrete Cosine Transform*), já consagrada em padrões anteriores de codificação de vídeo. A DCT opera apenas sobre blocos quadrados de resíduos, sendo que os tamanhos de bloco permitidos são de 4x4, 8x8, 16x16 e 32x32 amostras (SULLIVAN, OHM, *et al.*, 2012).

O processo de quantização é aplicado logo após as transformadas, aplicando uma operação de corte aos coeficientes obtidos pelo processo de transformadas. A etapa de quantização recebe os resíduos transformados pela DCT e aplica operações irreversíveis, eliminando ou atenuando dados de alta frequência, menos significativos ao olho humano. A quantização está diretamente ligada a um parâmetro de quantização (*Quantization Parameter* - QP) que indica a intensidade de perdas inseridas pela quantização. Quanto maior o QP, maiores serão os cortes

nos resíduos, o que tende a maiores perdas na qualidade do vídeo codificado. Por outro lado, valores altos de QP contribuem diretamente para a obtenção de altas taxas de compressão.

2.2.4 Codificação de entropia

A Codificação de entropia recebe o resíduo quantizado e aplica algoritmos de codificação sem perdas para manter a qualidade de vídeo e obter uma alta taxa de compressão. As técnicas de codificação utilizadas nesta etapa são fortemente baseadas em análises estatísticas e conseguem representar valores com uma maior densidade de probabilidade com um menor número de bits. A saída desta etapa é o *bitstream* do vídeo codificado. No HEVC, o algoritmo utilizado para a codificação de entropia utiliza o método de Codificação Aritmética Binária Adaptativa ao Contexto (CABAC) (SULLIVAN, OHM, *et al.*, 2012).

2.2.5 Reconstrução dos quadros de referência

O padrão HEVC possui um laço após a etapa de quantização, onde são aplicadas quantizações inversas e transformadas inversas sobre os resíduos, que depois são adicionados aos quadros reconstruídos de acordo com o processo de predição utilizado. Aqui também são aplicados filtros para remover artefatos inseridos pela codificação por blocos. Após a aplicação destes filtros, o quadro reconstruído é armazenado e poderá ser utilizado como referência para a codificação dos próximos quadros do vídeo. No HEVC, os filtros inseridos nesta etapa são o *Sample Adaptive Offset* (SAO), que é aplicado em áreas cujas amostras possuem intensidade homogêneas (SULLIVAN, OHM, *et al.*, 2012) e o *Deblocking Filter* (DF) que diminui o efeito de bloco gerado pela codificação por CUs (SULLIVAN, OHM, *et al.*, 2012). Este processo de reconstrução do quadro é fundamental para que os quadros de referência sejam os mesmos tanto no codificador quando no decodificador, evitando inconsistências no processo de decodificação (*drifting*).

2.3 Estimação de movimento e o acesso à memória externa

A estimação de movimento (ME), contida na predição inter, é a etapa mais complexa do processo de codificação de vídeo, representando entre 60 e 80% do tempo total de computação, tanto no H.264/AVC (KUHN, 1999) quanto no HEVC

(BOSSEN, BROSS, *et al.*, 2012). No entanto, esta também é a etapa que mais contribui para as elevadas taxas de compressão atingidas pelos codificadores atuais. A Figura 3 apresenta o diagrama de blocos do codificador HEVC com as respectivas complexidades associadas a cada módulo do codificador, onde pode ser visto que o módulo mais complexo do codificador é a predição inter-quadros, representando uma taxa de complexidade entre 77% e 81% (IPSL, 2012).

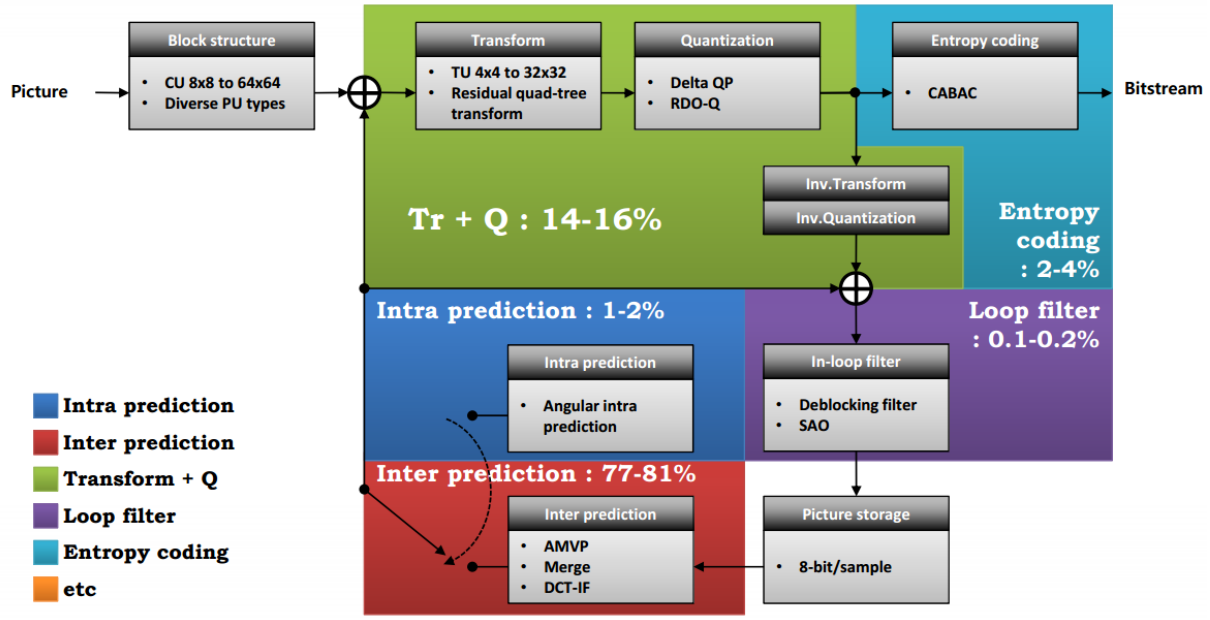


Figura 3 – Complexidade do codificador HEVC por bloco (IPSL, 2012)

A estimação de movimento é a responsável por explorar a redundância temporal presente em uma sequência de quadros. O processo de redução da redundância temporal é realizado para cada bloco do quadro atual (quadro a ser codificado). Este processo consiste em localizar, no quadro de referência, um bloco candidato que mais se assemelha ao bloco do quadro atual. Para fazer a busca pelos blocos é utilizado algum algoritmo de busca como, por exemplo, o *Full Search* (BHASKARAN e KONSTANTINIDES, 1997) ou o *Diamond Search* (KUHN, 1999). Para decidir qual será o melhor bloco candidato no quadro de referência, os algoritmos de ME utilizam algum critério de similaridade, como o SAD (*Sum of Absolute Differences*) (KUHN, 1999). O SAD é um dos critérios de similaridade mais utilizados devido à simplicidade do cálculo e facilidade de implementação em hardware. O cálculo do SAD consiste na soma das diferenças absolutas entre as amostras do bloco atual e dos blocos do quadro de referência. A Equação 2 apresenta o cálculo do SAD, onde R são as amostras do bloco de referência e O as amostras do bloco do quadro original.

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |R_{ij} - O_{ij}| \quad (2)$$

Assim que o bloco é encontrado, a ME deve gerar um vetor indicando o deslocamento deste bloco no quadro de referência, em relação à posição do bloco do quadro atual. Este vetor é chamado de vetor de movimento e deve ser enviado, junto com os resíduos resultantes das diferenças entre o bloco original e o bloco candidato escolhido, para as demais etapas do codificador.

Os vetores de movimento servem para compensar o movimento dentro de um quadro e são, juntamente com o resíduo gerado entre o bloco escolhido e o bloco do quadro atual, codificados e transmitidos. Portanto, os vetores de movimento devem ser escolhidos de forma a minimizar este resíduo e assim, reduzir o número de bits para a codificação deste erro nas etapas posteriores.

Nos codificadores de vídeo atuais, apenas o componente de luminância do bloco costuma ser considerado para a realização da estimação de movimento. Os vetores que são definidos para a luminância são reutilizados para a crominância. Esta estratégia é utilizada porque as informações de crominância são menos relevantes para a composição do vídeo (RICHARDSON, 2010).

O processo de busca da ME é bastante complexo e envolve um número elevado de acessos à memória. Para cada bloco codificado, a ME deverá buscar na memória uma área de busca, que pode ser um quadro de referência inteiro porém, geralmente se restringe a uma parte do quadro referente à região de interesse. Por exemplo, considerando um vídeo HD 1080p (1920x1080 pixels), com blocos de tamanho 64x64 e área de busca 256x256, seria necessário buscar 506 áreas de busca para codificar um quadro apenas. E se o quadro de referência fosse todo armazenado em memória interna, seriam necessários 1,98 MB de memória, sendo este um tamanho muito grande para uma memória interna.

Além desse elevado número de acessos à memória externa realizada pela ME, outro grande problema é a largura de banda envolvida nesta comunicação. Para a configuração apresentada anteriormente, seria necessário uma largura de banda de aproximadamente 950 MB/s. Além disso, essa comunicação pode ser realizada utilizando barramento externo ao chip que é, de maneira geral, ordens de grandeza mais lenta que uma comunicação interna ao chip, finalmente, há a dissipação de potência do circuito devida a alta frequência exigida para a comunicação.

Os quadros de referência ficam armazenados em um *buffer* chamado *Decoded Image Buffer* (DPB). Usualmente, esse *buffer* fica alocado na memória externa devido à grande quantidade de informação: tipicamente, essa memória é uma DRAM (*Dynamic Random-Access Memory*). Devido a isso, a comunicação com a memória domina o consumo de energia nos codificadores de vídeo, uma vez que a potência dissipada, devido à alta frequência exigida pelo acesso e tráfego de dados à memória DRAM, é geralmente superior à potência dissipada pelo núcleo de processamento do codificador de vídeo. Durante o processamento da ME, a energia consumida relacionada às memórias contribui com aproximadamente 90% do total de energia consumida pela ME, sendo que, deste total, metade do consumo é dado pelas memórias externas e a outra metade pelas memórias internas (ZATT, SHAFIQUE, *et al.*, 2011). Em um decodificador HEVC, em torno de 74% do consumo de energia está relacionada aos acessos à DRAM (TIKEKAR, HUANG, *et al.*, 2014).

Além disso, o HEVC permite o uso de vários quadros de referência na ME durante o processo de codificação (BOSSEN, BROSS, *et al.*, 2012). Esta ferramenta gera melhores resultados de codificação ao custo de um aumento significativo de acessos à memória e aumento de complexidade (BOSSEN, BROSS, *et al.*, 2012). A Figura 4 apresenta a largura de banda de memória necessária para a ME em três formatos de vídeos de alta definição 2160p (3840x2160 pixels), 1080p (1920x1080 pixels) e 720p (1280x720 pixels). Os resultados apresentados consideram uma taxa de processamento de 30 quadros por segundo (QPS) e as mesmas configurações de tamanho de área de busca e bloco do exemplo anterior, nenhuma técnica de redução de largura de banda foi considerada, i.e., todos os dados necessários são buscados diretamente na memória externa. Além disso, a Figura 4 mostra as taxas de transferência máxima de três tecnologias de DDR, relacionando os requisitos da ME e as capacidades das memórias.

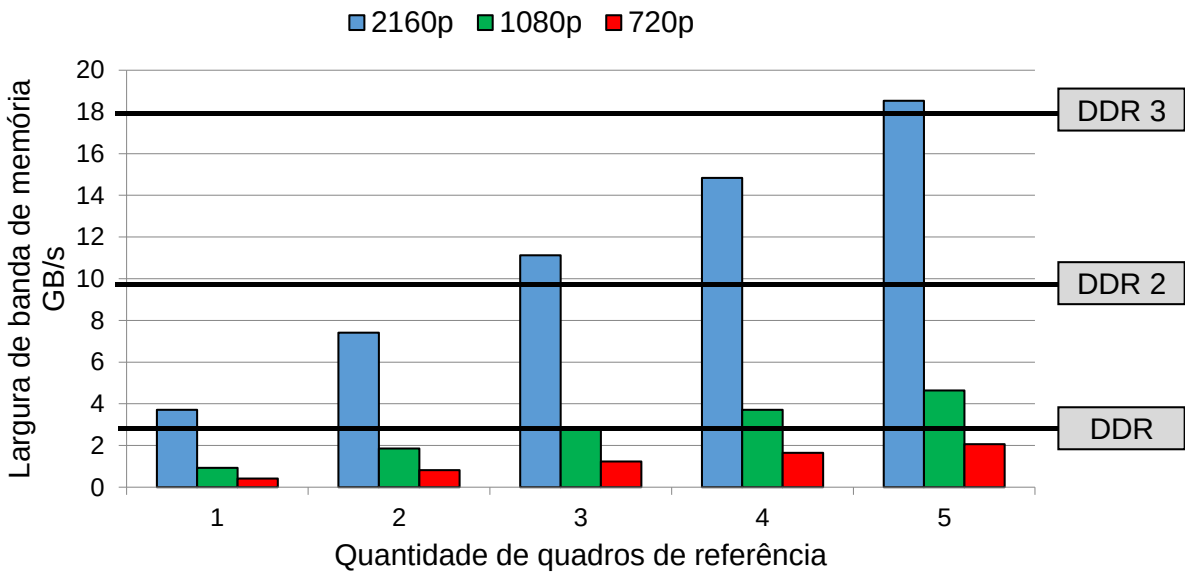


Figura 4 – Requisitos de largura de banda de memória para ME

A partir da Figura 4 pode-se notar que, dependendo do número de quadros de referência utilizado pela ME, nem mesmo as memórias DRR atuais são capazes de atender as demandas de largura de banda de memória exigidas pela ME. Desta forma, o projeto do codificador de vídeo deve considerar que o maior gargalo do sistema se encontra na comunicação entre memória externa e as unidades de processamento. Portanto, há uma grande necessidade por técnicas que reduzam o volume da comunicação com a memória externa durante o processo de codificação, especialmente durante a etapa da ME.

2.4 Considerações finais do capítulo

Este capítulo apresentou alguns conceitos básicos utilizados na codificação de vídeo, além das ferramentas utilizadas pelos codificadores. Estas ferramentas são empregadas para explorar e reduzir as redundâncias de dados nos vídeos digitais, porém apresentam um alto custo em complexidade e acessos à memória. Foi apresentado um breve histórico dos padrões de codificação de vídeo e como os padrões anteriores influenciaram na construção do padrão estado da arte de codificação de vídeo, o padrão HEVC. O fluxo de codificação do HEVC foi apresentado, destacando a importância de cada um dos módulos do codificador para geração do *bitstream* final.

Entre os módulos apresentados, a ME é destaque, pois é uma das ferramentas presentes em todos os padrões de codificação, sendo responsável

pelos maiores ganhos em taxa de compressão. Porém, é o módulo do codificador que apresenta a maior complexidade e o que mais realiza acessos à memória, sendo esse um dos maiores problemas nos codificadores de vídeo atuais.

Além dos problemas relacionados aos acessos à memória externa durante o processo da ME, foi também discutida a importância de reduzir o consumo energético relacionado à memória, principalmente para atender as demandas do processamento em tempo real de vídeos de alta definição, além das restrições de desempenho e consumo de energia apresentados por dispositivos móveis alimentados por bateria.

3 ESTADO DA ARTE

Muitas das aplicações que utilizam codificadores de vídeo são sistemas embarcados, onde a restrição na energia é extremamente severa. Como foi discutido na seção anterior, o projeto de um codificador de vídeo deve buscar a diminuição dos acessos à memória externa, para respeitar as restrições de largura de banda disponível para a comunicação com a memória e diminuir o consumo de energia gerado por esses acessos e tráfego de dados. Entretanto, existem trabalhos na literatura que propõem estratégias para reduzir a largura de banda de memória em codificadores de vídeo. Assim, este capítulo irá apresentar e discutir os trabalhos mais relevantes, que abordam diferentes estratégias para a redução de largura de banda de memória para diferentes padrões de codificação de vídeo.

Estas estratégias se baseiam em duas técnicas principais, que são: (1) estratégias de reutilização de dados utilizando memórias cache; e (2) compressão de quadros de referência, antes de serem armazenadas na memória externa.

A reutilização de dados através de uma memória cache interna, integrada ao chip, trabalha de modo a armazenar os dados que potencialmente serão mais acessados pela unidade de processamento. Entretanto, além do emprego de uma memória local, o projetista também deve se preocupar em garantir a sua eficiência. Para tanto, técnicas como a antecipação da busca de dados e previsão de acessos são amplamente utilizadas (SHAFIQUE, ZATT, *et al.*, 2012). A Figura 5 mostra a utilização da técnica de reuso de dados na comunicação entre a memória externa e as unidades de processamento.

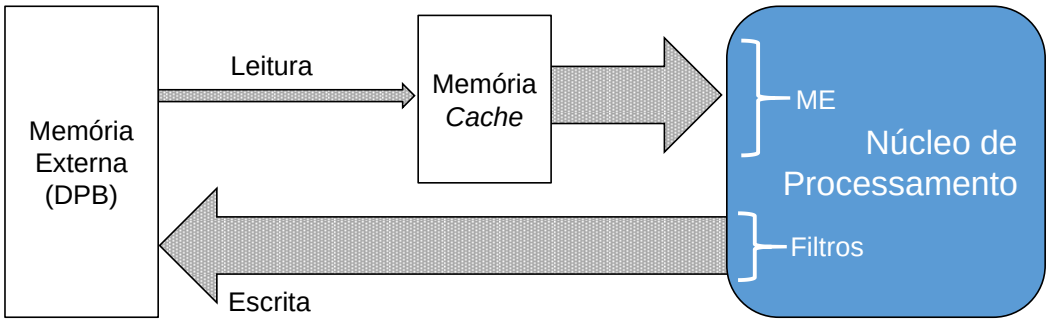


Figura 5 – Comunicação utilizando a técnica de reuso de dados

As soluções que utilizam a compressão de quadros de referência requerem duas etapas a mais no processo de codificação, que são: (1) comprimir o quadro de referência antes de ele ser enviado à memória externa; e (2) descomprimir o quadro que está na memória externa quando ele for utilizado na codificação. Esta abordagem introduz ganhos de diferentes maneiras, diminuindo o número de acessos à memória, diminuindo o tamanho da memória externa e reduzindo a largura de banda necessária na comunicação, gerando uma redução no consumo de energia do codificador. O uso desta técnica pode ser observado na Figura 6, onde é ilustrada a comunicação entre memória externa e as unidades de processamento através da utilização de um compressor/descompressor de quadros de referência.

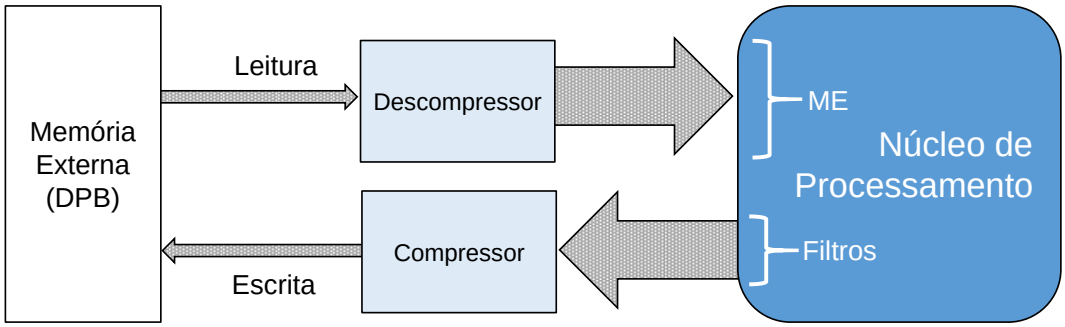


Figura 6 – Comunicação utilizando a compressão de quadros de referência

Nas próximas seções serão apresentados os trabalhos mais relevantes encontrados na literatura, que visam reduzir os problemas gerados na comunicação entre memória externa e o núcleo processamento, de acordo com as duas abordagens apresentadas.

3.1 Reuso de dados

O trabalho (TUAN, CHANG e JEN, 2002) apresenta várias estratégias de reuso de dados considerando, basicamente, dois diferentes níveis de redundâncias de informações: (1) regiões sobrepostas entre blocos candidatos vizinhos dentro de uma mesma área de busca (*Level* A e B) e (2) regiões sobrepostas entre as áreas de busca de blocos atuais vizinhos (*Level* C e D). A Figura 7 apresenta as quatro estratégias, onde as regiões destacadas em cinza escuro representam os dados que são reusados em cada estratégia. A área de busca (AB) (-X, +Y] apresentada na Figura 7, indica que a área de busca suporta um deslocamento de X pixels para a esquerda e Y pixels para a direita do bloco original na direção horizontal, e X pixels para cima e Y pixels para baixo na direção vertical do bloco original. Por exemplo: uma AB (-96, +96] é representado como ABv = ABh = 96 + 96 = 192. Neste caso a

dimensão real de uma área de busca para um bloco N x N seria (ABh + N) * (ABv + N).

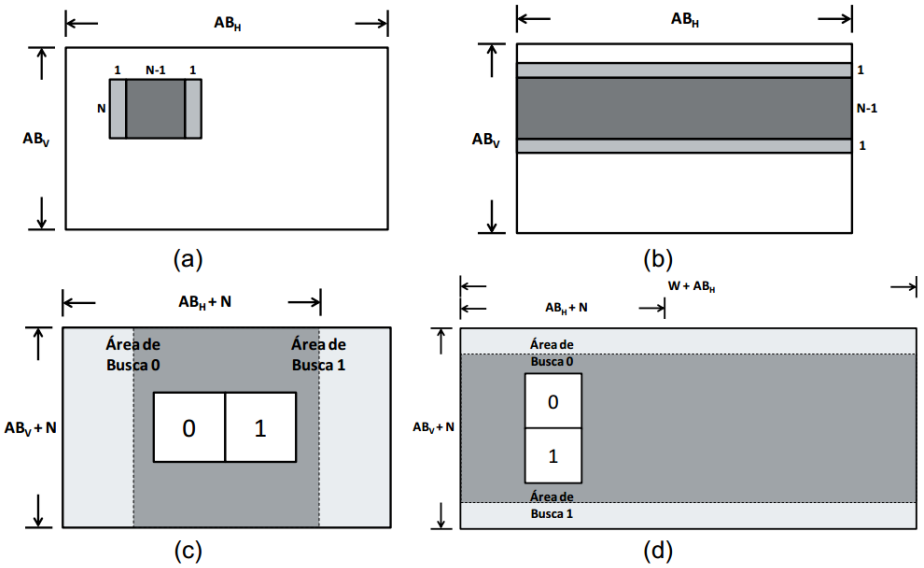


Figura 7 – Estratégias de reuso de dados propostas em Tuan: (a) *Level* A, (b) *Level* B, (c) *Level* C e (d) *Level* D (TUAN, CHANG e JEN, 2002)

O reuso de dados *Level* A, como mostrado na Figura 7 (a), aplica o reuso de dados entre dois blocos candidatos vizinhos. Neste caso, uma região de *N * (N - 1)* é reusada. No Level B Figura 7 (b), o reuso de dados é explorado considerando uma faixa de blocos de candidatos vizinhos horizontalmente dentro da área de busca (AB). Nesta estratégia, uma região de *(N - 1) * (ABh + N - 1)* é reusada. Em ambos *Level* A e *Level* B, o reuso é explorado dentro de uma mesma área de busca.

Nos esquemas *Level* C e *Level* D é explorado um nível mais alto de reuso: as áreas comuns entre áreas de buscas de blocos atuais vizinhos. O *Level* C reaproveita as informações entre áreas de busca entre dois blocos atuais vizinhos. Isto pode ser observado na Figura 7 (c). O número de amostras reusadas com o Level C é igual a *(ABv + N - 1) * (ABh - 1)*. A estratégia Level D aplica a mesma ideia, porém considerando as áreas de pesquisa de todos os blocos vizinhos dentro de todo o quadro (a dimensão W na Figura 7 (d) representa a largura do quadro). Neste caso, o número de amostras reusadas é *(ABv - 1) * (W + ABh - 1)*. O trabalho de (LI, ZHENG e ZHANG, 2007) apresenta uma solução arquitetural utilizando reuso de dados com *Level* D.

A estratégia apresentada no trabalho de Chen (CHEN, HUANG, *et al.*, 2006) é uma expansão do reuso *Level* C. O *Level* C+ considera o reuso de regiões

sobrepostas das áreas de buscas entre blocos atuais vizinhos nas duas direções: horizontal e vertical.

O *Level* C+ apresenta melhores resultados de eficiência quando comparado às estratégias *Level* A-D (TUAN, CHANG e JEN, 2002). A medida de eficiência utilizada pelo autor considera o compromisso entre a largura de banda com a memória principal e o tamanho da memória interna ao chip para guardar localmente os dados reusados. Como a abordagem *Level* C+ exige um processamento fora da ordem usual, o trabalho também propõe algumas ordens de processamento de blocos atuais de modo a não prejudicar o desempenho dos módulos posteriores a ME. A Figura 8 apresenta o reuso de dados obtido com a utilização do *Level* C+, dados quatro blocos atuais e suas respectivas áreas de busca.

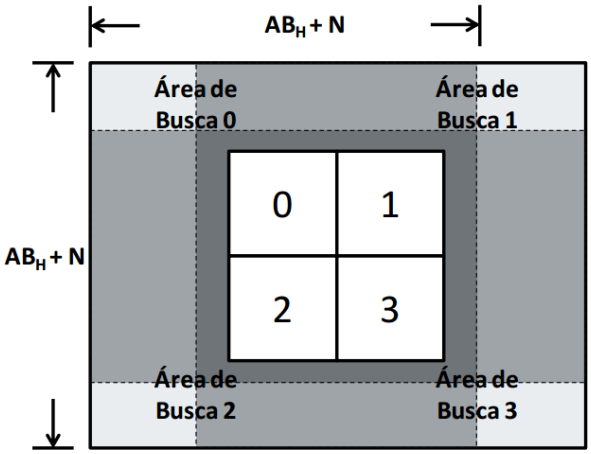


Figura 8 – Reuso de dados com a estratégia Level C+

No *Level* C+, o número de amostras reusadas é *(ABv – nN - 1) * (ABh - 1)*. Esta solução é capaz de atingir uma taxa de redução de 46% na largura de banda de memória quando comparado à estratégia *Level* C, e uma redução de 82% quando comparado ao método tradicional, sem utilização de reuso de dados. No entanto, para que isso aconteça, há um aumento na memória interna de 18%. Esta redução ocorre na comunicação com a memória externa, porém com a utilização de uma memória cache uma nova comunicação é inserida e os custos de acesso a ela, apesar de menores, não podem ser desconsiderados.

O trabalho apresentado em Grellert (GRELLERT, SAMPAIO, *et al.*, 2011) traz uma estratégia de reuso de dados de vários níveis, sendo baseado na sobreposição das áreas dos blocos vizinhos durante o processo de buscas na memória. Este trabalho propõe uma solução com dois níveis de hierarquia de memória: uma cache local e um buffer local. Esta estratégia de reuso de dados com

dois níveis faz com que esta solução atinja uma redução de largura de banda de memória de até 90% para acessos de leitura, quando utilizado na codificação de vídeos HD 1080p. No entanto, é necessária uma memória interna de 185 KBytes para a implementação da cache e do buffer local. Esta solução foi projetada em VHDL e a sua arquitetura apresentou desempenho capaz de processar vídeos em HD 720p a 56 QPS e vídeos HD 1080p a 25 QPS.

3.2 Compressão de quadros de referência

Os algoritmos para a compressão de quadros de referência podem ser divididos em dois grandes grupos: compressão com perdas e compressão sem perdas de qualidade. A compressão com perdas geralmente utiliza técnicas simples de compressão, porém o seu processo insere perdas na qualidade do vídeo codificado, que não podem ser recuperadas no processo de decodificação. Já na compressão sem perdas a qualidade do quadro não é alterada após o processo de codificação, e após o processo de decodificação é possível obter um quadro exatamente idêntico ao original.

As duas próximas subseções apresentam os principais trabalhos encontrados na literatura, com soluções para a compressão de quadros de referência com e sem perdas, respectivamente.

3.2.1 Compressão com perdas de qualidade

A solução proposta em Ivanov (IVANOV e MOLONEY, 2008) se baseia na compressão de blocos com tamanho 2x2 amostras, utilizando diferentes modos de predição. Isto é bastante semelhante ao módulo de predição Intra dos codificadores de vídeo H.264 e HEVC, mas é aplicada de uma forma mais simples. Esta solução atinge uma taxa de compressão constante de 50%, porém com um aumento no tempo computacional de 15% e uma perda significativa de qualidade, que varia de 0,13 a 0,87 dB.

O trabalho de Budagavi (BUDAGAVI e ZHOU, 2008) utiliza um algoritmo de compressão de comprimento fixo baseado em blocos de quantização escalar (SAYOOD, 2005), chamada *Min-Max-Scalar-Quantization* (MMSQ). Foram desenvolvidos dois algoritmos, o MMSQ 6bpp e o MMSQ 5bpp. Esses algoritmos atingem uma redução de 25% a 37,5%, respectivamente, do tamanho da memória usada para armazenar os quadros de referência, e também estima que esse mesmo

percentual possa ser reduzido da taxa de transferência da largura de banda de memória, de acordo com a configuração que é aplicada. Porém, estes algoritmos introduzem uma perda de qualidade de 0,159 dB no MMSQ 5bpp e de 0,043 dB no MMSQ 6bpp. Esta é uma perda considerável, já que a escala do PSNR é logarítmica. Finalmente, o trabalho apresentado por Budagavi foi avaliado apenas para vídeos de baixas resoluções (704x480 pixels). Com isso, os resultados para vídeos de alta resolução (720p ou superiores) não podem ser estimados.

A solução apresentada por (CHENG, TSENG e CHEN, 2009) consiste na utilização do algoritmo de compressão de imagem SPIHT (*Set Partitioning in Hierarchical Trees*) juntamente com a transformada DWT (*Discrete Wavelet Transform*). O algoritmo codifica inicialmente as informações mais importantes através de uma representação de plano de bits por pixel. O algoritmo utiliza uma estrutura de árvores hierárquicas que explora de forma eficiente as propriedades da imagem transformada pela *wavelet* 2D. Esta estratégia é apresentada em três modos, sendo que dois deles são com perdas. O modo *half-size* pode reduzir até 65% dos acessos à memória externa, com perda de 0,1 dB, e o modo *quarter-size* reduz até 76% dos acessos com perda de 1,44dB em qualidade. No modo sem perdas de qualidade, a taxa de compressão atingida é de 62% em média para vídeos CIF.

O trabalho de Gupte (GUPTE, AMRUTUR, *et al.*, 2011) propõe uma solução baseada no algoritmo MMSQ (BUDAGAVI e ZHOU, 2008), proposto anteriormente. A melhoria proposta por Gupte reduz as perdas no processo de codificação do MMSQ. Os erros gerados são armazenados, de modo que estes possam ser retornados aos blocos correspondentes durante o processo da compensação de movimento (MC – *Motion Compensation*). Com esta solução a taxa de redução de largura de banda fica em torno de 23% para vídeos HD 1080p, com uma perda de qualidade de 0,01 dB no quadro de referência codificado.

Esta solução é interessante, mas também gera perdas durante o processo, porque a ME irá utilizar amostras de referência do quadro quantizado, e portanto, o melhor vetor de movimento pode não ser encontrado, simplesmente porque as amostras do quadro de referência não são mais iguais às originais. Além disso, esta solução não é compatível com as arquiteturas que aplicam os processos de ME e MC em conjunto. As arquiteturas que implementam ME e MC combinados podem pular o acesso à memória externa realizada pela MC, armazenando o bloco residual

no momento em que a ME encontra o melhor casamento para o bloco atual, reduzindo a largura de banda de memória no processo de codificação.

O trabalho de (MA e SEGALL, 2011) apresenta um algoritmo de compressão híbrido. Primeiro, a imagem é decomposta em amostras de baixa resolução (LR) e amostras de alta resolução (HR). Em seguida, é aplicada uma codificação residual entre as amostras HR e LR. Após a codificação, uma quantização é aplicada sobre o resíduo. Esta solução apresenta uma taxa de compressão de 31% e uma perda de qualidade de 0,15 dB, para cinco sequências de vídeo HD 1080p avaliadas.

Todas as soluções apresentadas nessa subseção apresentam degradação de qualidade no processo de codificação dos quadros de referência. Isto ocorre, pois os algoritmos desenvolvidos utilizam uma etapa de quantização, descartando, de forma irreversível, algumas informações. Este é um problema importante, uma vez que este erro pode ser cumulativo, e dependendo da configuração que o codificador de vídeo utilizar, um erro nas amostras de referência pode se propagar por diversos quadros codificados a partir destas amostras de referência, causando uma grande perda na qualidade final do vídeo codificado. Outro problema é que o codificador e o decodificador usariam referências diferentes, podendo ocasionar *drifting* no processo de decodificação do vídeo.

3.2.2 Compressão sem perdas de qualidade

O algoritmo *Hierarchical Minimum and Difference* (HMD) é proposto em (LEE, CHUNG, *et al.*, 2009). Esta solução utiliza uma codificação diferencial entre amostras e posteriormente, uma técnica de VLC para codificar os resíduos. O primeiro passo é encontrar a amostra de menor valor em um bloco 2x2. Após, encontra-se a amostra de menor valor em um bloco 4x4. Após encontrar a amostra de menor valor do bloco é realizada a codificação diferencial entre essa amostra e as demais amostras do bloco. Este processo se repete até blocos de tamanho 8x8. O algoritmo VLC utilizado é o *Exp-Golomb*, que utiliza uma tabela estática para armazenar os códigos para os resíduos. Esta tabela pode chegar a 340 KB de tamanho, quando são utilizados vídeos com resolução HD 1080p. Esta técnica atinge uma taxa de redução média de 39% para quadros de luminância. Os resultados foram obtidos através da simulação para sete vídeos com resolução HD 1080p, sendo que foram considerados 30 quadros de cada sequência.

O algoritmo proposto por (KIM e KYUNG, 2010) consiste de dois passos. O primeiro é um método baseado em predição hierárquica média de pixel e de cópia, o *Hierarchical Average and Copy Prediction* (HACP). Este método utiliza três níveis de predição e quatro modos para predição (predição por média horizontal e vertical, predição por cópia horizontal e vertical). O segundo passo envolve o truncamento de bits pouco significativos, o *Significant Bit Truncation* (SBT). Neste método somente os bits significativos dos erros são armazenados, a quantidade de bits significativos é mandada por um cabeçalho, e os bits não significativos são representados por um bit apenas, por exemplo, 00000101 é representado por 0101. Os resultados experimentais mostram uma redução de 60% da largura de banda de memória, em média. Os resultados da implementação de hardware mostram um custo em área de 36K *gates*, com uma taxa de processamento de 14,2 pixels/ciclo o que é suficiente para processar vídeos HD a 30 QPS.

O trabalho desenvolvido em (BAO, ZHOU e GOTO, 2010) apresenta uma solução que realiza a compressão sobre blocos de tamanho 8x8, além do uso de duas memórias cache para armazenar os comprimentos dos blocos e dos quadros comprimidos. Este trabalho utiliza seis modos diferentes para realizar a codificação, sendo que cada modo é composto por uma codificação diferencial utilizando a técnica de *Differential Pulse Code Modulation* (DPCM) (SALOMON, 2007), e um tipo de codificação de tamanho variável (VLC). Esses seis modos são aplicados sobre as amostras de cada bloco 8x8. Após isso, uma decisão deve ser tomada para saber qual dos modos obteve melhor resultado. Essa técnica atinge uma redução na largura de banda de memória de 50% a 60% durante o processo de codificação do vídeo. Esse trabalho foi estendido em (BAO, ZHOU, *et al.*, 2012), o qual foi integrado a uma arquitetura de ME para o padrão H.264/AVC.

No trabalho de (ZHOU, ZHOU, *et al.*, 2011) a solução desenvolvida é aplicada sobre amostras de luminância e crominância. As amostras de luminância são divididas em blocos de 8x4, e as amostras de crominância em blocos de tamanho 4x2. Esses tamanhos foram escolhidos porque o decodificador está configurado para decodificação de vídeos com subamostragem 4:2:0. O algoritmo proposto no trabalho utiliza uma codificação diferencial a partir de um DPCM que é aplicado sobre o bloco. Após o DPCM, os blocos são subdivididos em blocos de tamanho 2x2 para serem codificados utilizando um VLC. O VLC utilizado no trabalho é baseado em códigos *Exp-Golomb*, e de acordo com os valores de máximo e

mínimo de cada bloco 2x2, um tipo de código é utilizado, sendo que os códigos ficam armazenados em uma tabela de códigos de tamanho variável que é interna ao compressor. Nos experimentos foram utilizadas três sequências de vídeo com resolução HD 1080p *e* os resultados apresentam uma taxa de compressão média de 51%.

Em (KUO e LIN, 2012) a solução baseada em codificação em linha é proposta. Esta solução se difere das demais uma vez que as outras utilizam codificação baseada em blocos. Esta solução utiliza quatro técnicas em conjunto para realizar a compressão dos quadros de referência. A primeira, é uma codificação diferencial entre a amostra atual e a amostra da coluna anterior. Em seguida é aplicado um esquema de classificação em dois estágios para classificar todos os resíduos. Após, o resíduo é avaliado e uma consulta ao dicionário de códigos é feita: caso o resíduo esteja no dicionário, o código presente neste dicionário será utilizado; caso não esteja, um esquema de truncamento de bit adaptativo semelhante ao apresentado em (KIM e KYUNG, 2010) é aplicado para gerar o código. Posteriormente, uma codificação de *Huffman* é empregada para otimizar os códigos. No final, uma compressão é aplicada aos códigos gerados pelo algoritmo de *Huffman*. Esta técnica é chamada de *Head Code Compression* (HCC). O HCC tira proveito da similaridade entre os códigos gerados pelo algoritmo de *Huffman*, reaproveitando partes de códigos que são iguais, e desta forma, uma redução maior é obtida. Esta solução atinge uma taxa de compressão média de 66%. Para a obtenção desse resultado, 13 sequências de vídeo com resolução HD 1080p foram utilizadas, sendo que 60 quadros de cada uma das sequências foram considerados.

Em (GUO, ZHOU e GOTO, 2013) é proposto um algoritmo de compressão que utiliza uma predição no domínio espacial com múltiplos modos de direção. Esta abordagem é adotada com o intuito de se adaptar às várias características de uma imagem. Após a etapa de predição, um reagrupamento residual é realizado a fim de melhorar o desempenho da compressão, utilizando para isso uma codificação de comprimento semifixo (*Semi-Fixed Length* – SFL). Os resultados mostram que esta solução pode atingir uma taxa de compressão média de 57% para os vídeos das Condições Comuns de Testes (CCTs) (BOSSEN, 2012). No trabalho (ZHOU, GUO, *et al.*, 2014) os autores apresentam uma discussão sobre o acesso aleatório à memória externa, utilizando esse algoritmo. Além disso, em (ZHOU, GUO, *et al.*, 2014) novos resultados da implementação em hardware são apresentados. Em

(GUO, ZHOU e GOTO, 2014) é apresentada uma extensão desses trabalhos, que inclui esse algoritmo e introduz as arquiteturas com acesso aleatório à memória externa.

Em (LEE, CHEN e YOU, 2014) é utilizado o algoritmo *Autocorrelation-Based Frame Lossless Recompression* (AFLR). Esse algoritmo realiza uma predição de 4 modos distintos nas amostras. Após a etapa de predição, é realizada uma codificação utilizando o algoritmo *Golomb-Rice* adaptativo. Esse trabalho também propõe uma memória DMMA (*Dual-Mode Memory Addressing*), para prover acesso aleatório aos dados codificados na memória externa. A taxa de compressão dessa solução atinge em média 51%. Esse resultado foi obtido utilizando 10 vídeos HD 1080p.

Por fim, o trabalho de (CHEN, LEE e CHEN, 2014) apresenta um algoritmo de compressão multidirecional (*Multi-Directional Lossless Recompression* - MDLR). Esse algoritmo trabalha com blocos de tamanho 16x16 e para cada bloco é executada uma predição diferencial multidirecional (*Multi-Directional Prediction* – MDP), a fim de obter todos os resíduos do bloco. Por último, uma codificação *Golomb-Rice* adaptativa é realizada sobre os resíduos. Esta estratégia atinge uma taxa de compressão média de 42% para vídeos HD 1080p.

3.3 Síntese dos trabalhos relacionados

Três análises podem ser feitas sobre os trabalhos estado da arte. Primeiro, os trabalhos sobre reuso de dados geralmente atingem uma alta taxa de redução de largura de banda entre o núcleo de processamento e a memória externa. Entretanto, cabe salientar que esta abordagem reduz apenas os acessos de leitura da memória, sendo que o número de acessos para a escrita na memória externa permanecem iguais. Além disso, os autores não contabilizam os custos adicionais da inserção da memória cache, necessária para o reuso. A inserção desta cache implica em novos custos para manter e acessar os dados armazenados, tanto em área quanto em consumo de energia.

Segundo, os trabalhos com soluções de compressão com perdas de qualidade podem gerar *drifting* entre codificador e decodificador, e isto resulta em perdas de eficiência de codificação, uma vez que a predição inter-quadros usará as amostras deste quadro de referência (que agora apresenta perdas em relação ao original), degradando a qualidade do vídeo codificado. Nenhum dos trabalhos

citados apresenta dados sobre a perda de qualidade final do vídeo codificado a partir destes quadros de referência com perdas de qualidade. Estas perdas podem ser cumulativas, pois os quadros codificados a partir destas amostras de referência com perdas podem ser usados como referência para a codificação de quadros futuros.

Terceiro, os trabalhos com soluções de compressão sem perdas de qualidade utilizam diversas técnicas de codificação em conjunto para atingirem altas taxas de compressão. Isto implicará em alta dissipação de potência e elevada latência quando estas soluções forem implementadas em hardware. Além disso, a maioria desses trabalhos são implementações para o padrão H.264/AVC, e poucos deles implementam estas soluções em hardware, o que dificulta a avaliação do consumo energético dessas soluções.

Neste sentido, o algoritmo ideal de compressão de quadros de referência deve evitar perdas de qualidade, proporcionando altas taxas de compressão e com baixo custo computacional. Vale ressaltar que o processo de compressão e descompressão de quadros de referência deve ser inserido dentro do fluxo de uma etapa crítica de desempenho dos codificadores de vídeo. Logo, esse processo deve apresentar o menor custo computacional e latência possíveis, evitando a criação de um novo gargalo de desempenho no sistema. Além disso, as soluções de compressão de quadros de referência pode ser utilizada em conjunto com a técnica de reuso de dados, reduzindo ainda mais a largura de banda final.

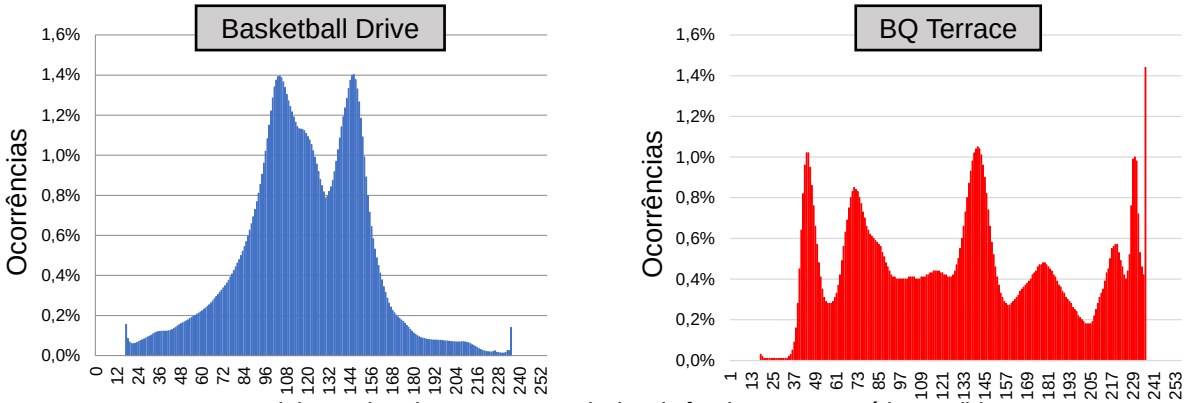
Dessa forma, como pode ser observado na descrição dos trabalhos relacionados, os trabalhos relacionados falham em alguns aspectos como taxa de compressão, custo computacional, quantidade de memória utilizada, entre outros. Aliado a isso, a crescente demanda por vídeos de alta resolução em dispositivos móveis faz com que novas soluções ainda mais eficientes sejam desenvolvidas. Nesse sentido, essa dissertação apresenta soluções para a compressão de quadros de referência sem perdas de qualidade com baixa complexidade, a qual gera redução de largura de banda de memória e redução do consumo de energia no codificador de vídeo. Os próximos capítulos irão apresentar as análises realizadas sobre vídeos de alta definição, e as soluções desenvolvidas para a compressão dos quadros de referência.

4 ANÁLISE DA DISTRIBUIÇÃO DE AMOSTRAS

Neste capítulo será apresentada uma análise de dados realizada com algumas sequências de teste, a fim de compreender melhor o comportamento das amostras que compõem os vídeos digitais. Nesta análise foram consideradas amostras de luminância de oito sequências de vídeo HD 1080p, sendo elas: *Pedestrian Area, Cactus, Tennis, Rolling Tomatoes, Station2, Riverbed, BQ Terrace, Basketball Drive*. Estas sequências foram codificadas e posteriormente decodificadas através do software de referência do padrão HEVC – HM-12.0 (MCCANN, BROSS, *et al.*, 2013) – com o algoritmo de busca TZ (*Test Zone Search*), configuração de acesso aleatório e QPs (*Quantization Parameter*) 22, 27, 32 e 37, de acordo com as condições comuns de testes (BOSSEN, 2012). Este processo de transformação das amostras de entrada se faz necessário porque os quadros de referência são quadros que já foram codificados, i.e., quadros reconstruídos. Esse capítulo está divido em três seções: a seção 4.1 apresenta a análise dos vídeos utilizando a codificação diferencial para três modos de predição: horizontal, vertical e diagonal; a seção 0 apresenta uma nova abordagem para codificação diferencial, a qual utiliza dois modos de predição em sequência, horizontal e vertical. Além disso, é realizado um comparativo entre essas distribuições após a utilização dessas técnicas com base no valor da entropia. A seção 4.3 apresenta as considerações finais do capítulo.

4.1 Análise da distribuição para predição simples

Na Figura 9 são apresentados dois histogramas de amostras de luminância para duas sequências de vídeo completas, *Basketball Drive* (a) e *BQ Terrace* (b), em resolução HD 1080p. A partir dessa figura pode-se perceber que os histogramas apresentam uma distribuição completamente diferente, sendo que essa distribuição varia de acordo com as características dos vídeos. Este tipo de distribuição esparsa, observada nesses dois histogramas, é altamente indesejável para algoritmos de codificação de entropia, especialmente para uma codificação de tamanho variável.



(a) Valor das amostras de luminância reconstruídas (b)

Figura 9 – Histograma das amostras reconstruídas

Para quantificar o valor esperado de informação contida em uma mensagem, neste caso uma sequência de vídeo, a entropia de Shannon pode ser utilizada (SHANNON, 1948). A entropia de Shannon (*H(X)*) fornece um limite para a compressão sem perdas de informação para uma variável aleatória *X* com *n* possibilidades *{x1, x2, ..., xn}* e pode ser definida com na Equação 3.

$$H(X) = -\sum_{i=0}^{n} p(x_i) * \log_b p(x_i) \quad (3)$$

Na Equação 3, *p(xi)* é a função de densidade de probabilidade de resultados *xi*. A base *b* do logaritmo define a unidade da entropia, e.g., usando bit como unidade, o *b* deve ser igual a 2. Além disso, a entropia de Shannon pode ser utilizada para demonstrar que a distribuição original de uma sequência de vídeo não é eficiente para algoritmos de codificação de entropia.

Desta forma, podemos quantificar as distribuições apresentadas na Figura 9 através da entropia de Shannon. Para a distribuição apresentada na Figura 9 (a) são necessários 7,07 bits para armazenar cada amostra, após utilizar um algoritmo ótimo de codificação de entropia. Considerando a distribuição na Figura 9 (b), são necessários 6,84 bits por amostra. Nesses casos, comparando com a sequência de vídeo original, que contém 8 bits por amostra, a taxa de redução de dados seria de 11,55% para a distribuição na Figura 9 (a) e 14,54% na Figura 9 (b).

Entretanto, as amostras de vídeos naturais apresentam elevada redundância espacial. Isto significa que amostras vizinhas tendem a ter valores semelhantes, exceto quando ocorre alguma borda (amostras vizinhas com grandes diferenças de valores). Assim, técnicas como a codificação diferencial (CD) podem ser utilizadas para alcançar taxas de redução de dados mais elevadas. Entretanto, a codificação

diferencial precisa seguir algum modo de predição, que pode ser horizontal, vertical ou algum tipo de ordem diagonal. Esses modos de predição para a codificação diferencial são semelhantes aos modos da predição intra dos padrões H.264/AVC e HEVC (SULLIVAN, OHM, *et al.*, 2012).

A Figura 10 apresenta histogramas de resíduos gerados pela codificação diferencial para três modos de predição: horizontal, vertical e diagonal. As Equações 4, 5 e 6 definem esses modos de predição.

$$Predição\ Horizontal = Amostra[i][j] - Amostra[i][j-1] \quad (4)$$

$$Predição\ Vertical = Amostra[i][j] - Amostra[i-1][j] \quad (5)$$

$$Predição\ Vertical = Amostra[i][j] - Amostra[i-1][j-1] \quad (6)$$

Os histogramas da Figura 10 são apresentados para dois vídeos, *Basketball Drive* e *BQ Terrace*. Esses dois vídeos foram utilizados, pois apresentam uma distribuição bastante distinta entre as amostras, conforme pôde ser observado na Figura 9. O intervalo de resíduos para essa codificação diferencial varia de -255 até 255. Entretanto, o intervalo mostrado na Figura 10 é menor, pois a grande concentração de resíduos está próximo a zero.

A Figura 10 (a) e (b) apresenta histogramas de resíduos para uma predição horizontal, esses resíduos são obtidos através da Equação 4. Com esta técnica simples, a entropia das sequências de vídeo da Figura 10 (a) e (b) diminui para 3,02 e 3,49 bits por amostra, atingindo em média, 62,25% e 56,37% de taxa de compressão respectivamente, no melhor caso. O sentido usado para realizar a CD é um fator importante, uma vez que, resultados diferentes podem ser obtidos para o cálculo da diferença entre amostras em sentidos diferentes. Por exemplo, se houver uma borda horizontal, quando a CD é realizada horizontalmente, os resíduos obtidos ficarão concentrados em torno do valor zero. O mesmo acontece para as bordas verticais quando a CD é aplicada verticalmente.

Esse efeito pode ser visto na Figura 10 (c) e (d) onde a CD utiliza uma predição vertical através da Equação 5. Aplicando a codificação diferencial nesse modo, a entropia para essas sequências de vídeo fica em 3,22 e 3,89 bits por amostra, atingindo em média, 59,75% e 51,37% de taxa de redução para os vídeos *Basketball Drive* e *BQ Terrace*, respectivamente. Comparando esses resultados com os resultados obtidos no sentido horizontal, se percebe que houve uma piora nos resultados utilizando a codificação diferencial no sentido vertical. Isso ocorre porque

as amostras apresentam maior correlação com suas amostras vizinhas na horizontal.

Também foi avaliada a codificação diferencial utilizando uma predição diagonal. Nesse modo de predição os resíduos são obtidos através da Equação 6.

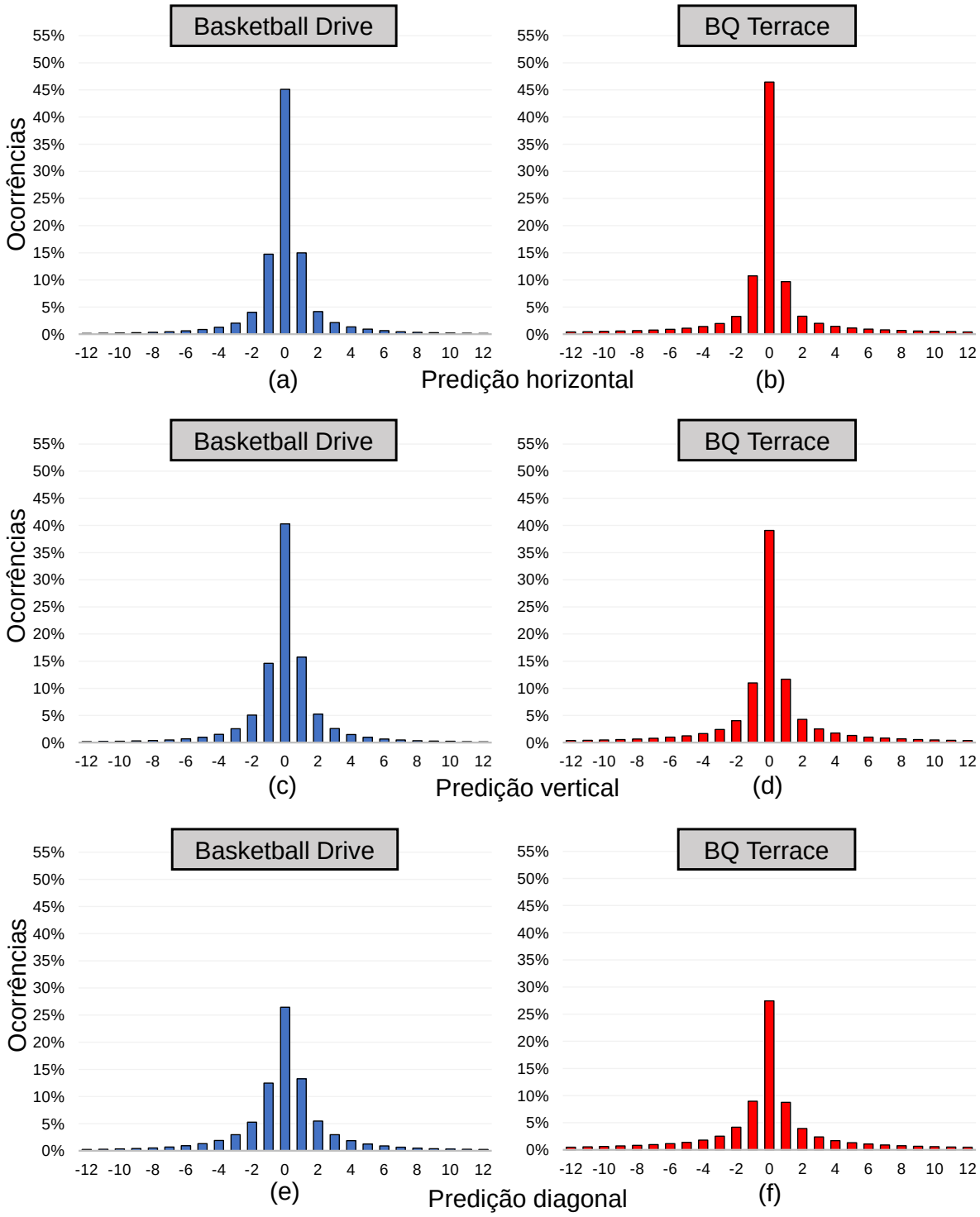


Figura 10 – Histograma de resíduos para predição horizontal, (a) e (b), vertical, (c) e (d), e diagonal, (e) e (f)

Para esse modo a entropia atingida para as sequências Basketball Drive e BQ Terrace ficou em 4,68 e 5,50 bits por amostra, sendo que com essa quantidade de bits por amostra é possível atingir uma taxa de compressão de 41,5% e 31,25% para esses dois vídeos, sendo esse valor bem inferior aos outros dois modos testados. Percebe-se então que o sentido ideal da predição pode variar de acordo com o conteúdo dos vídeos, não se limitando inclusive às direções vertical e horizontal.

4.2 Análise da distribuição para dupla predição

Como visto anteriormente o conteúdo dos vídeos pode variar bastante, o que impede que um único sentido na codificação diferencial seja eficiente em todos os cenários. Uma alternativa a esse fato é o uso de diversos modos (ou sentidos) para aumentar a eficiência da predição. No entanto, essa abordagem exige um custo adicional em bits para indicar o modo de predição utilizado. Além disso, a complexidade do processo aumenta consideravelmente, uma vez que será necessário avaliar qual o melhor modo da codificação diferencial, e para isso, uma codificação de entropia teria que ser realizada para cada modo de predição da codificação diferencial, para então definir qual o modo que atinge os melhores resultados de compressão. Isso acaba inserindo grande latência no processo de codificação dos quadros de referência, o que é extremamente indesejável, principalmente por este processo estar situado no gargalo de processamento dos codificadores de vídeo. Soluções em hardware podem explorar o paralelismo, no entanto, os custos adicionais em hardware e consumo de energia podem tornar a solução menos eficiente.

Como visto na seção anterior, geralmente os melhores modos de predição são o horizontal e o vertical. Isso pode ser percebido em todos os oito vídeos utilizados nos experimentos para os quatro QPs. A forma encontrada para utilizar esses dois modos juntamente foi a utilização de duas codificações diferenciais aplicadas na sequência, desta forma a CD pode ser aplicada na horizontal para manipular bordas horizontais, e com os resíduos obtidos por esse processo, uma segunda CD é aplicada para manipular as bordas verticais. Este processo será chamado de dupla codificação diferencial (DCD). A Figura 11 (a) e (b) apresenta os histogramas de resíduos gerados a partir da DCD. A entropia da distribuição nas Figura 11 (a) e (b) atingem, respectivamente, 2,31 e 3,11 bits por amostra, o que

torna possível atingir uma taxa de redução de dados de 71,12% e 61,13%, respectivamente.

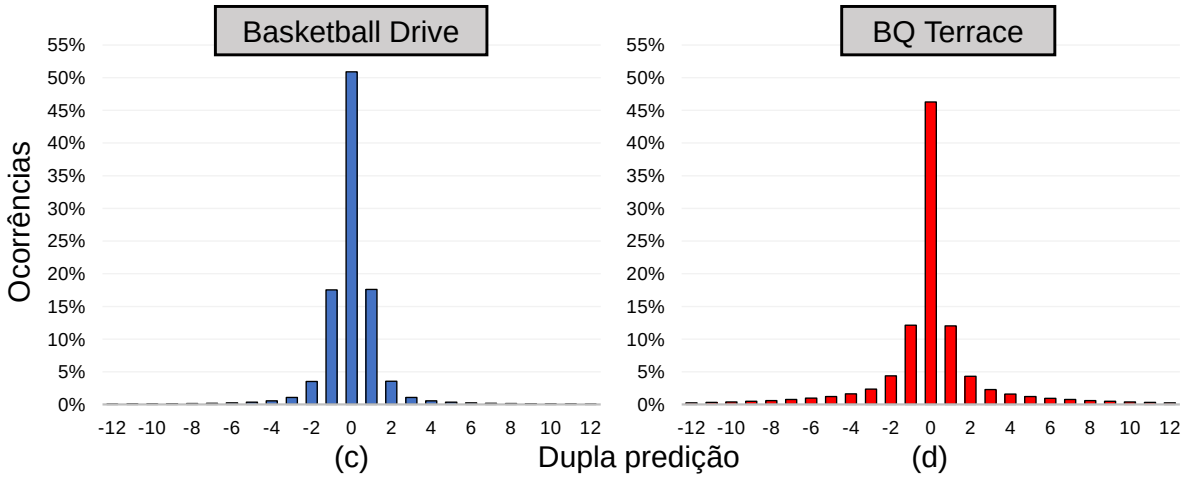


Figura 11 – Histograma de resíduos para codificação diferencial com dupla predição

Esse tipo de codificação diferencial – DCD – é uma inovação importante para a literatura, onde não foi encontrado nenhum trabalho que utilize esse método. Alguns trabalhos estado da arte, como visto no Capítulo 3, utilizam algum tipo de codificação diferencial. Porém, esses trabalhos realizam a codificação diferencial em um único sentido. Alguns desses trabalhos também avaliam mais de um sentido, porém, de forma separada, o que demanda, além de informação lateral para identificar o modo, mais de uma iteração no processo de codificação, ou exigindo recursos adicionais para o paralelismo.

Essa nova abordagem para codificação diferencial acabou gerando melhores resultados quando comparado aos outros três modos de predição apresentados na seção anterior. Isso ocorre porque a DCD codifica tanto as bordas horizontais quanto as bordas verticais em um mesmo quadro, sendo esses os sentidos de borda predominante nos vídeos. Além disso, essa abordagem não insere informação lateral para identificar o modo utilizado na codificação, nem gera custo computacional extra para avaliar qual o melhor modo utilizado.

A Tabela 1 apresenta a entropia das sequências de teste reconstruídas, sem a codificação diferencial e com a utilização da codificação diferencial, para os três modos de predição simples e para a dupla predição. Quanto menor o valor da entropia, maior pode ser a taxa de compressão alcançada. O valor da entropia é apresentado para as oito sequências de teste utilizadas com a média dos quatro QP. Nessa tabela pode ser visto que as sequências reconstruídas sem a codificação

diferencial apresentam uma alta entropia, o que reduz a taxa de compressão que pode ser obtida com essas sequências. Porém, com a utilização da codificação diferencial, independentemente do modo utilizado, percebe-se a melhora na entropia, o que leva a distribuição a atingir melhores taxas de compressão. A menor entropia é alcançada com a utilização da dupla predição, resíduo duplo. Esse valor é menor que qualquer outro modo de predição utilizado, sendo em média 67% menor que a entropia das sequências reconstruídas, o que demostra que a utilização da DCD é uma boa alternativa para ser utilizada em conjunto com uma codificação de entropia, nas soluções para compressão dos quadros.

Tabela 1 – Entropia das sequências de teste com e sem a codificação diferencial

Vídeo	Sem Codificação Diferencial	Predição Horizontal	Predição Vertical	Predição Diagonal	Dupla Predição
Basketball Drive	7,07	3,02	3,22	4,68	2,31
BQ Terrace	6,84	3,29	3,89	5,50	3,11
Cactus	7,55	3,60	3,43	5,95	2,85
Pedestrian Area	6,78	2,89	2,86	4,15	1,87
Tennis	7,21	2,53	2,85	4,03	1,93
Rolling Tomatoes	6,58	1,89	2,29	3,95	1,37
Station2	6,86	3,16	3,24	4,26	2,45
Riverbed	6,60	3,32	4,05	5,77	2,30
Média	6,94	2,96	3,23	4,79	2,27

4.3 Considerações finais do capítulo

Esse capítulo apresentou a análise dos vídeos que serviu de base para o desenvolvimento dos algoritmos propostos nessa dissertação. A codificação diferencial foi apresentada para três modos distintos: horizontal, vertical e diagonal. A dupla codificação diferencial utiliza dois modos em sequência: horizontal e vertical. Um comparativo entre esses modos de codificação diferencial foi realizado com base em oito sequências de teste. Para esse comparativo foi utilizado o valor de entropia de cada distribuição. A dupla codificação diferencial foi a que apresentou os melhores resultados de entropia, sendo em média 67% menor que a entropia das sequências reconstruídas. Entre os modos da codificação diferencial simples, o modo horizontal foi o que apresentou os melhores resultados, atingido uma entropia 57% menor que as sequências reconstruídas.

Nessa análise pôde ser visto que, de forma diferente da distribuição das amostras reconstruídas, a qual varia entre diferentes sequências de vídeo, quando

uma codificação diferencial ou dupla codificação diferencial é utilizada, a distribuição das amostras residuais permanecem concentradas em torno de zero, mesmo para vídeos com diferentes características de movimentação e luminosidade. Esse comportamento permite uma codificação de entropia mais eficiente sobre essas distribuições. Dessa forma, uma alta taxa de redução de dados pode ser atingida com a utilização da codificação diferencial, aliada a uma codificação de entropia eficiente.

5 ALGORITMOS PARA COMPRESSÃO DOS QUADROS DE REFERÊNCIA

Com base no estudo apresentado na capítulo anterior, este trabalho propõe algoritmos para compressão de quadros de referência que utilizam a codificação diferencial, simples e dupla, aliada a uma codificação de entropia eficiente. A codificação de entropia utiliza códigos de tamanho variável que ficam armazenados em uma tabela estática. No total, três algoritmos foram desenvolvidos: o DRFC (*Differential Reference Frame Coder*), o DRFVLC (*Differential Reference Frame Variable-Length Coder*) e o DDRFVLC (*Double Differential Reference Frame Variable-Length Coder*). Todos esses algoritmos seguem o mesmo comportamento, aplicando uma codificação diferencial em nível de blocos seguida de codificação de entropia. As diferenças entre eles estão no tipo de codificação diferencial utilizada e na forma como a codificação de entropia é aplicada.

- Algoritmo DRFC: utiliza codificação diferencial simples horizontal combinada com uma codificação de entropia onde os códigos estáticos têm o mesmo tamanho, sendo que este tamanho varia de 2 a 5 bits, de acordo com a configuração.
- Algoritmo DRFVLC: também utiliza uma codificação diferencial simples horizontal, porém a codificação de entropia utiliza códigos de tamanho variável gerados a partir do algoritmo de *Huffman* (HUFFMAN, 1952), o qual gera códigos mais otimizados para valores com mais ocorrência.
- Algoritmo DDRFVLC: utiliza a dupla codificação diferencial (DCD), a qual elimina bordas horizontais e verticais a partir da aplicação em sequência das codificações horizontal e vertical, e codificação de entropia utilizando códigos de tamanho variável de *Huffman*.

As próximas seções irão explicar em detalhes o processo de codificação e decodificação utilizados por cada um dos três algoritmos desenvolvidos, destacando o fluxo de codificação/decodificação (comum a todos os algoritmos) e as diferentes

abordagens no processo de codificação diferencial e codificação de entropia utilizados por cada algoritmo. Também é apresentada uma seção com uma avaliação dos algoritmos para diferentes configurações de utilização, a fim de definir a configuração com os melhores resultados em termos de taxa de compressão e custo computacional.

5.1 Codificador

O processo de codificação de um bloco consiste em: (i) obter um novo bloco do quadro de referência a ser enviado à memória externa; (ii) transformar a distribuição das amostras do bloco utilizando a codificação diferencial; (iii) aplicar a codificação de entropia para codificar os resíduos gerados pela etapa anterior e (iv) armazenar os resíduos codificados na memória. A Figura 12 apresenta o fluxograma do codificador (DRFC, DRFVLC e DDRFVLC) com as etapas mencionadas acima.

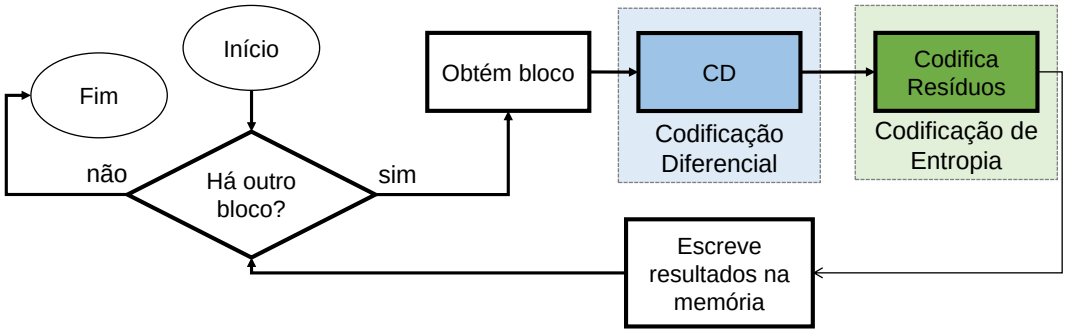


Figura 12 – Fluxograma do processo de codificação (DRFC, DRFVLC e DDRFVLC)

5.1.1 Processo da codificação diferencial

O processo de codificação diferencial (CD) é utilizado para concentrar a distribuição de amostras residuais, e é realizada subtraindo a amostra atual de amostras vizinhas. Nesse trabalho foram utilizados dois tipos de CD: a codificação diferencial simples horizontal (CD-H), e a dupla codificação diferencial (DCD).

A CD-H é definida por três casos, como apresentado na Figura 13, para um bloco BO [*i*][*j*] de 4x4 amostras. O primeiro caso (*i = 0 e j = 0*), ocorre quando a primeira amostra do bloco é processada, onde *i* e *j* designam as linhas e as colunas do bloco, respectivamente. Esta amostra é simplesmente copiada para o bloco residual CD-H (número 33 em negrito na Figura 13). O segundo caso (*i > 0 e j = 0*) é a codificação da primeira coluna do bloco, em que o resíduo é obtido subtraindo a

posição [*i-1*] [*j*] da posição [*i*] [*j*] a partir do BO. O terceiro caso (*j > 0*) é a codificação das linhas do bloco. A Equação 7 generaliza este processo:

$$CD\text{-}H[i][j] = \begin{cases} BO[i][j], & i \wedge j = 0 \\ BO[i][j] - BO[i-1][j], & i>0 \wedge j=0 \\ BO[i][j] - BO[i][j-1], & j>0 \end{cases} \quad (7)$$

É importante notar que no exemplo apresentado na Figura 13, o BO é composto por duas regiões (separadas por uma borda vertical) com valores de amostras bastante diferentes. Neste tipo de cenário, o bloco CD-H residual pode apresentar valores com altas amplitudes, como pode ser visto na Figura 13 (resíduos sublinhados). Nestes casos, a aplicação de uma codificação diferencial na direção vertical pode atenuar a amplitude de grande parte destes resíduos, sem causar impactos significativos nos demais valores.

O processo da dupla codificação diferencial (DCD) é composto por duas codificações diferenciais em sequência, codificação diferencial horizontal (CD-H) e codificação diferencial vertical (CD-V). Desta forma, este processo reduz os altos valores de resíduos que podem ter sido gerados por bordas verticais dentro do bloco. Com isso, o processo de DCD apresenta dois casos distintos. O primeiro é quando a primeira linha (*i=0*) está sendo codificada, nesse caso a linha é somente copiada e seu valor não é alterado. No segundo caso (*i>0*), o resíduo é obtido subtraindo a posição [*i-1*][*j*] da [*i*][*j*]. A definição deste processo é apresentado na Equação 8.

$$CD\text{-}V[i][j] = \begin{cases} CD\text{-}H[i][j], & i = 0 \\ CD\text{-}H[i][j] - CD\text{-}H[i-1][j], & i > 0 \end{cases} \quad (8)$$

É importante notar que a Figura 13 é um exemplo de um bloco de tamanho 4x4, e alguns dos resíduos permanecem com amplitudes elevadas após o processo de DCD. No entanto, para a codificação de blocos maiores, a ocorrência desses valores altos se torna menos significativa devido à grande ocorrência de resíduos com valores próximos a zero, fazendo com que a distribuição dos valores fique mais centrada em zero e consequentemente, aumentando o potencial de eficiência da codificação de entropia. Os algoritmos DRFC e DRFVLC utilizam a codificação diferencial simples horizontal, CD-H, enquanto que o algoritmo DDRFVLC, utiliza o processo de codificação diferencial dupla, DCD.

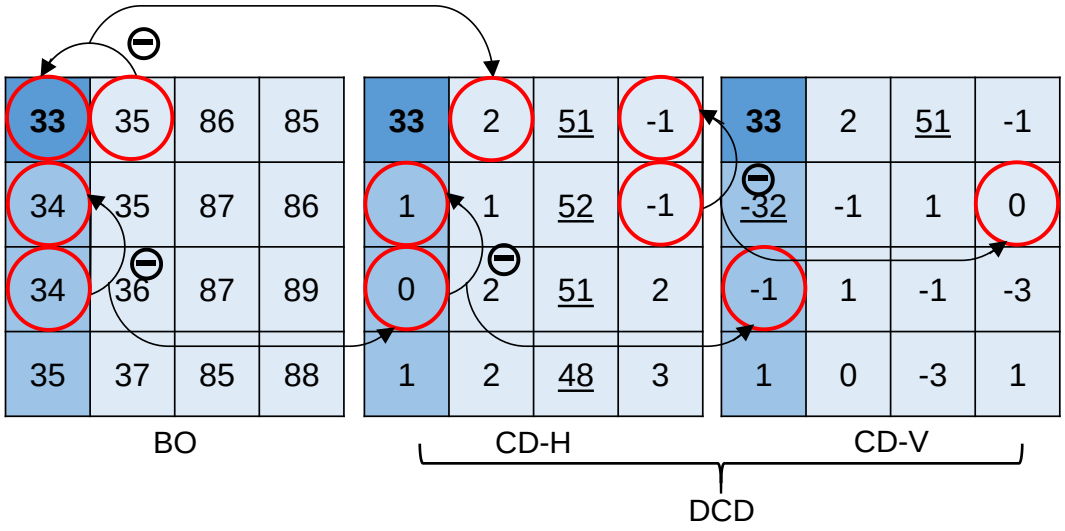


Figura 13 – Exemplo do processo de dupla codificação diferencial

5.1.2 Processo da codificação de entropia

Quando o processo da codificação diferencial termina, os resíduos estão prontos para serem codificados. Neste momento inicia o processo da codificação de entropia. Nesse trabalho foi utilizada uma codificação de entropia que utiliza códigos estáticos. Esses códigos podem ser de tamanho fixo ou de tamanho variável gerados a partir do algoritmo de *Huffman*. Esses códigos são definidos em tempo de projeto e ficam armazenados em uma tabela estática. De acordo com o valor do resíduo um código diferente deve ser utilizado.

No algoritmo DRFC são utilizados códigos de tamanho fixo cujo tamanho, dependendo da configuração, pode ser de 2, 3, 4 ou 5 bits. A Tabela 2 apresenta os resíduos com os respectivos códigos otimizados para o tamanho 4 bits, onde a faixa de representação de resíduos é de [-7, 7]. Pode-se notar que há apenas dois tamanhos diferentes de códigos, 4 ou 12 bits. Códigos de 12 bits serão utilizados quando o valor do resíduo estiver fora do intervalo [-7,7], nesses casos serão utilizados 12 bits (4 bits da exceção mais 8 bits do valor da amostra original). Caso o tamanho do código fixo utilizado fosse 2 bits, o intervalo de representação seria apenas [-1, 1], e nesse caso, os dois tamanhos utilizados seriam 2 bits, do resíduo codificado, ou 10 bits, exceção mais amostra original.

Tabela 2 – Códigos para tamanho fixo de 4 bits do algoritmo DRFC

Resíduo	Código	Resíduo	Código	Resíduo	Código	Resíduo	Código
0	0000	-2	0100	-4	1000	-6	1100
1	0001	3	0101	5	1001	7	1101
-1	0010	-3	0110	-5	1010	-7	1110
2	0011	4	0111	6	1011	Exceção	1111

Nos outros dois algoritmos desenvolvidos, DRFVLC e DDRFLVC, a codificação de entropia utiliza uma tabela estática com códigos de tamanhos variáveis. Esta tabela estática foi desenvolvida utilizando a codificação de *Huffman*.

O processo de codificação de *Huffman* exige uma análise de dois passos, uma para calcular a probabilidade de ocorrência de cada símbolo, criando o dicionário de códigos, e outra para traduzir os símbolos para os novos códigos (SALOMON, 2007). Devido a esses dois passos, uma alta latência é inserida no processo de codificação, tornando esta abordagem ineficiente para os sistemas de codificação de vídeo, onde o desempenho é uma limitação crítica. Para resolver este problema, uma tabela de *Huffman* estática pode ser utilizada. A tabela estática é gerada anteriormente ao processo de codificação, a partir de dados estatísticos da aplicação do algoritmo de *Huffman* sobre várias sequências de teste. Estes dados, no entanto, precisam ser cuidadosamente escolhidos para que a tabela estática atenda as diferentes probabilidades de ocorrência dos símbolos (valores dos resíduos). Nesse trabalho foi utilizado um conjunto de oito sequências de vídeo, com características de movimento e luminosidade diferentes, a fim de gerar uma tabela estática suficientemente robusta para a codificação de todo tipo de vídeo.

A Tabela 3 apresenta os resíduos e os símbolos correspondentes para o intervalo [-32, 32] da tabela estática de *Huffman*. Esta tabela foi gerada para blocos de tamanho 64x64, utilizando o processo de dupla codificação diferencial. Como pode ser visto na Tabela 3, tamanhos menores de código são gerados para símbolos que ocorrem mais frequentemente (os símbolos próximos a zero têm maior ocorrência, como visto no capítulo 4).

Tabela 3 – Tabela com códigos de *Huffman* para o intervalo [-32,32]

Resíduo	Código	Resíduo	Código	Resíduo	Código
-32	10001010001101	-10	101100000	12	1011001011
-31	1000101001110	-9	101100100	13	1011000011
-30	1011000010000	-8	10000101	14	1000101011
-29	1011000010110	-7	10001111	15	10111101110
-28	1011001010100	-6	10111100	16	10111101010
-27	1011110101110	-5	1000110	17	10110001010
-26	1011110110100	-4	100000	18	10001011110
-25	100010100010	-3	101110	19	10001010010
-24	100010101010	-2	1010	20	101111011011
-23	101100001001	-1	111	21	101100101011
-22	101100101001	0	0	22	101100101000
-21	101111010110	1	110	23	101100001010
-20	10001010000	2	1001	24	100010101011
-19	10001010100	3	101101	25	100010100110
-18	10001011111	4	1011111	26	1011110110101
-17	10110001011	5	1000100	27	1011110101111
-16	10111101100	6	10110011	28	1011001010101
-15	10111101111	7	10001110	29	1011000010111
-14	1000101110	8	10000100	30	1011000010001
-13	1011000100	9	101100011	31	1000101001111
-12	1011110100	10	100010110	32	1000101000111
-11	100001101	11	100001100	Exc.	10000111

Os resíduos simples possuem tamanho de 9 bits, enquanto que os resíduos gerados pela codificação diferencial dupla geram resíduos de 10 bits. O processo de codificação de entropia consiste em trocar o resíduo gerado pela codificação diferencial por um código otimizado que está na tabela estática. Por exemplo, no caso do DRFC utilizando os códigos apresentados na Tabela 2 o resíduo 0 é representado pelo código “0000” e o resíduo -7 pelo código “1110”. Esses códigos são armazenados em sequência em uma fila, sem informações de início e fim do código. Assim, para esses dois códigos a fila ficaria: “00001110”.

No caso da tabela com códigos de tamanho variável, utilizando os exemplos da Tabela 3 teríamos o código “0” para o resíduo 0 e o código “10001111” para o resíduo -7. A fila de saída com esses códigos ficaria “010001111”. Nota-se no entanto, uma maior dificuldade no processo de decodificação de códigos de tamanho variável, uma vez que, a priori, o tamanho do código não é conhecido. Dessa forma, o processo de decodificação deverá ser bit a bit. Isso não ocorre quando é utilizado códigos de tamanho fixo, onde o tamanho do código é sempre conhecido. O processo de decodificação será melhor explicado nas próximas seções.

5.2 Decodificador

O decodificador executa o processo inverso da codificação, sendo que os passos são: (i) executar o processo inverso da codificação de entropia, i.e. a partir dos códigos gerar o resíduo correspondente; (ii) executar a codificação diferencial inversa, i.e. nesta etapa o resíduo gerado será somado à amostra anterior; e (iii) gerar o bloco reconstruído para ser utilizado pela ME. A Figura 14 apresenta o fluxograma em alto nível do processo de decodificação. Esse processo é válido para os três algoritmos desenvolvidos (DRFC, DRFVLC e DDRFVLC), sendo que as principais diferenças de cada algoritmo são internas aos módulos e serão detalhadas nas próximas subseções.

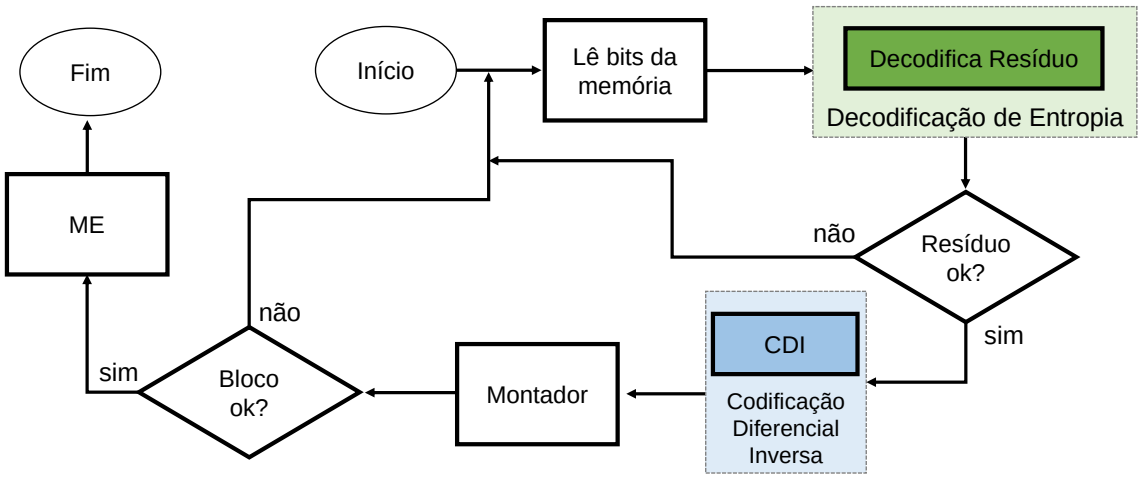


Figura 14 – Fluxograma do processo de decodificação (DRFC, DRFVLC e DDRFVLC)

5.2.1 Processo da decodificação de entropia

A fase de decodificação inicia com a leitura do bloco codificado a partir da memória externa, aplicando o processo inverso da fase de codificação. Neste ponto também há diferenças entre os algoritmos desenvolvidos. Enquanto o algoritmo DRFC tem apenas dois tamanhos de códigos possíveis, os outros dois algoritmos, DRFVLC e DDRFVLC, possuem muitos códigos com tamanhos distintos, sendo esta a principal diferença entre eles, uma vez que códigos VLC geram alta dependência de dados entre os códigos. Na decodificação de tamanho variável, não há nenhuma informação explicita para detectar o início ou o fim do código, exigindo assim uma leitura bit a bit da informação codificada.

Na decodificação no algoritmo DRFC, a leitura dos bits é realizada sempre no tamanho do resíduo codificado: se essa informação representa um código no intervalo, o valor do resíduo já é gerado; caso essa informação seja um código de exceção, é feita a leitura dos próximos 8 bits que contém a informação da amostra original. Esse processo é repetido até o bloco completo ser decodificado. Nos algoritmos DRFVLC e DDRFVLC, sabe-se que os primeiros 8 bits correspondem à primeira amostra original do bloco, desde que foi enviado para a memória sem qualquer transformação ou codificação. Assim, esta amostra é utilizada como ponto de partida para o processo de codificação diferencial inversa (CDI).

Após a leitura dos primeiros 8 bits, referentes à primeira amostra do bloco original, a próxima etapa é responsável por obter um código válido. Para fazê-lo, cada bit lido da memória é armazenado em uma lista, a qual é utilizada como chave de acesso à tabela estática de códigos. Se a lista não estiver na tabela de códigos, o próximo bit lido da memória é concatenado à lista, e então a tabela é novamente consultada. Quando o código for encontrado na tabela, o resíduo original é obtido. Caso o código encontrado seja o código de exceção, os próximos 8 bits referentes a amostra original, no caso do DRFC e do DRFVLC, serão lidos da lista. Caso seja o DDRFVLC, o que será lido da lista será um resíduo de 9 bits. Esta etapa é repetida até que o bloco esteja completamente decodificado.

5.2.2 Processo da codificação diferencial inversa

Quando o bloco de resíduos é completamente decodificado pelo processo de decodificação de entropia, o processo da dupla codificação diferencial inversa (DCDI) é usado para obter o bloco de amostra original. Este processo é separado em dois passos: codificação diferencial simples vertical inversa (CD-VI) e codificação diferencial simples horizontal inversa (CD-HI).

A CD-VI, tem como entrada o bloco residual (BR) gerado pela etapa de codificação de entropia inversa. O processo da CD-VI é definido na Equação 9:

$$CD\text{-}VI[i][j] = \begin{cases} BR[i][j], & i = 0 \\ BR[i][j] + CD\text{-}VI[i-1][j], & i > 0 \end{cases} \quad (9)$$

O processo da CD-VI tem dois casos. O primeiro caso (*i = 0*) apenas copia a primeira linha armazenada no bloco residual, BR. Esta linha é utilizada para obter os resíduos restantes. Em seguida, no segundo caso (*i > 0*), as outras linhas são processadas somando o resíduo BR[*i*][*j*] com o resíduo CD-VI[*i*] [*j-1*].

Após completar o processo de CD-VI, a codificação diferencial horizontal inversa (CD-HI) é aplicada. O processo CD-HI recebe como entrada o resíduo obtido pela CD-VI. A CD-HI é definida por três casos, como apresentado na Figura 15, para um bloco BR [*i*][*j*] de 4x4 amostras. O primeiro caso (*i = 0 e j = 0*), ocorre quando a primeira amostra do bloco é processada. Essa amostra é simplesmente copiada para o bloco residual CD-HI (número 33 em negrito na Figura 15). O segundo caso (*i > 0 e j = 0*) é a decodificação da primeira coluna do bloco, em que o resíduo é obtido somando o valor da posição CD-VI[*i*][*j*] com o valor da posição CD-HI[*i - 1*] [*j*]. O terceiro caso (*j > 0*) é a decodificação das linhas do bloco. Neste caso, o resíduo é obtido somando o valor da posição CD-VI[*i*] [*j*] com o valor CD-HI[*i*] [*j - 1*]. A Equação 10 define o processo da codificação diferencial horizontal inversa.

$$CD\text{-}HI[i][j] = \begin{cases} CD\text{-}VI[i][j], & i \wedge j = 0 \\ CD\text{-}VI[i][j] + CD\text{-}HI[i-1][j], & i>0 \wedge j=0 \\ CD\text{-}VI[i][j] + CD\text{-}HI[i][j-1], & j > 0 \end{cases} \quad (10)$$

A Figura 15 ilustra o processo da dupla decodificação inversa (DCDI) com um exemplo para um bloco de 4x4 amostras. Pode-se perceber que o bloco residual (BR) utilizado nessa figura é o mesmo bloco CD-V apresentado na Figura 13. Isto foi feito para demonstrar que o resultado gerado na decodificação, bloco CD-HI, é o mesmo bloco BO da Figura 13 e portanto, se percebe que não há perdas de informações durante o processo.

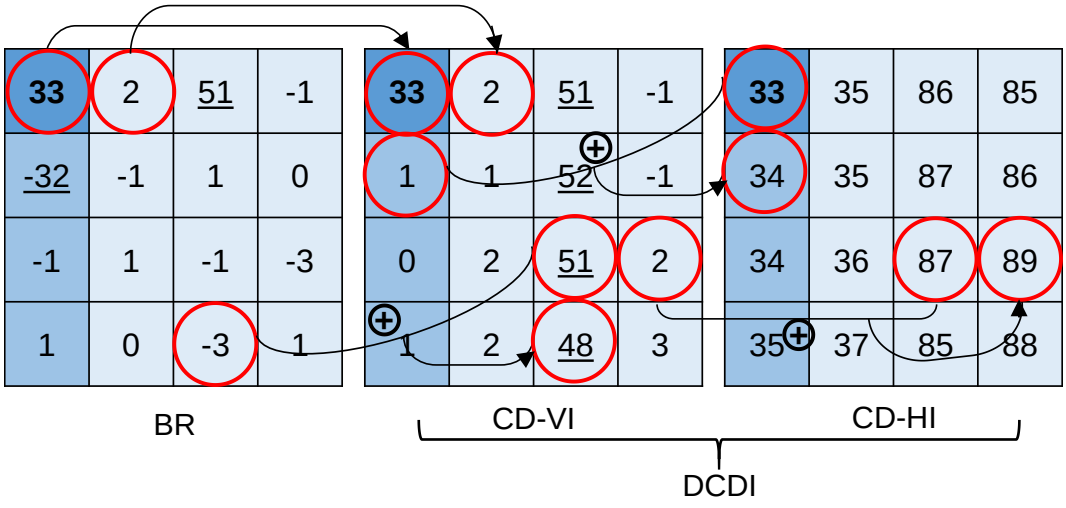


Figura 15 – Exemplo do processo da dupla codificação diferencial inversa

5.3 Exploração algorítmica

A taxa de compressão atingida pelos algoritmos depende das suas configurações. Parâmetros como o tamanho do bloco e o tamanho da tabela estática para armazenar os códigos usados no processo de codificação influenciam diretamente os resultados obtidos. Para encontrar o melhor *tradeoff* entre taxa de compressão e custo computacional foram avaliadas 114 configurações diferentes, variando o tamanho do bloco (B) e o tamanho da tabela estática de códigos (TC) para os três algoritmos.

Para realizar a compressão, a tabela estática com os códigos deve ser armazenada em uma memória ROM. Além disso, na etapa de decodificação, o tamanho da tabela e a quantidade de códigos com diferentes tamanhos trazem grandes impactos em custo computacional, além de custos distintos de área e consumo de energia para a implementação em hardware. Isso se deve ao fato de que na decodificação de um código VLC não há nenhuma informação explícita para detectar o limite do início ou do fim do código em um fluxo de entrada, o que leva a uma procura bit a bit, ou a uma versão paralela para testar todos os tamanhos ao mesmo tempo. Logo, pensando em custo computacional, tabelas de códigos menores são preferíveis.

Além disso, os resíduos gerados pela dupla codificação diferencial variam de -510 a 510 e desta forma, para armazenar esta tabela completa seriam necessários 1021 entradas, uma para cada resíduo possível. No entanto, apenas os resíduos em torno de zero possuem alta ocorrência, como discutido no Capítulo 4. Assim, é possível reduzir a tabela de código de modo que os resíduos que tenham baixa ocorrência não sejam armazenados de forma direta. Para fazer isso sem perda de informações, um código de exceção é necessário para representar os resíduos que estão fora do intervalo estabelecido. Assim, quando o resíduo está dentro do intervalo, o código armazenado na tabela de códigos é usado; caso contrário, o código de exceção é utilizado, seguido pelo valor do resíduo usando representação de comprimento fixo.

Nesta avaliação, foram analisadas 12 tabelas de códigos com tamanhos diferentes. Para o algoritmo DRFC foram avaliados quatro intervalos: [-1, 1] (TC 1), [-3, 3] (TC 3), [-7, 7] (TC 7) e [-15, 15] (TC 15). Para os algoritmos DRFVLC e DDRFVLC oito intervalos da tabela de códigos foram consideradas: [-4, 4] (TC 4), [-

8, 8] (TC 8), [-16, 16] (TC 16), [-32, 32] (TC 32), [-64, 64] (TC 64), [-128, 128] (TC 128), [-255, 255] (TC 255) e [-510, 510] (TC 510). Além disso, seis tamanhos de blocos diferentes foram investigados (4x4, 8x8, 16x16, 32x32, 64x64 e 128x128). Para estes experimentos foi utilizado o conjunto de vídeos para testes descrito no Capítulo 4, considerando 4 valores distintos de QP: 22, 27, 32 e 37. Desta forma, cada um dos experimentos apresentados nas Figura 16, 17 e 18, representam o valor da taxa de compressão média para 32 simulações, composta por 8 sequências de vídeo para os 4 valores de QPs. Os resultados completos das taxas de compressão utilizados para gerar estes gráficos podem ser vistos no Apêndice A.

A Figura 16 apresenta os resultados dos 24 experimentos distintos realizados para o algoritmo DRFC. Nota-se na Figura 16 que o tamanho do bloco utilizado na codificação pouco interfere na taxa de compressão. Nesse caso, o que causa impacto na taxa de compressão é a quantidade de bits utilizados nos códigos, sendo que a configuração TC 1, que contém apenas três resíduos (-1, 1 e 0) codificados, e TC 3, que codifica os resíduos no intervalo [-3, 3], atingem sempre taxas de compressão maiores que os experimentos TC 7 e TC 15, considerando o mesmo tamanho de bloco utilizado. A configuração TC 3 atinge as melhores taxas de compressão, sendo que cada valor desse intervalo é representado por 3 bits. Além disso, essa é uma das configurações mais estáveis, sendo que a diferença entre a taxa de compressão do bloco 8x8 para e do bloco 128x128 é de apenas 1,15% (bloco 8x8: 51,33% e bloco 128x128: 52,48%).

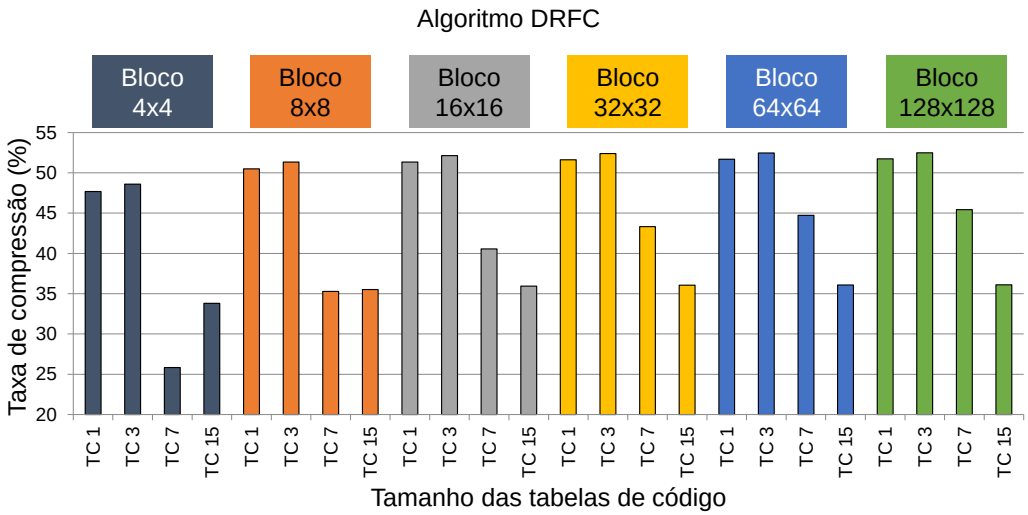


Figura 16 – Taxa de compressão para os 24 experimentos do algoritmo DRFC

O gráfico na Figura 17 mostra os resultados da taxa de compressão da exploração algorítmica realizada com o algoritmo DRFVLC. Para esse algoritmo foram realizados 42 experimentos. Os experimentos variaram no tamanho de bloco, de 4x4 até 128x128, e no tamanho das tabelas de códigos (TC), de TC 4 até TC 255, sendo TC 255 a tabela completa sem códigos de exceção. Nesse algoritmo é utilizada codificação diferencial simples e uma codificação de entropia com códigos de tamanho variável. Nota-se, na Figura 17, dois importantes comportamentos. Primeiro, a taxa de compressão aumenta conforme o tamanho do bloco aumenta. Isso já era esperado devido à alta similaridade entre amostras vizinhas, a qual é melhor explorada em blocos maiores.

Outro fato que também contribui para esse aumento na taxa de compressão em blocos maiores é que os resíduos vizinhos à primeira amostra do bloco geralmente possuem valores altos, e devido a isso, acabam tendo um código longo, que em blocos pequenos, afeta de maneira mais significativa a taxa de compressão.

O segundo ponto importante é que a taxa de compressão a partir da configuração TC 16 fica praticamente estabilizada. A taxa de variação entre TC 16 (65,09%) e TC 255 (65,31%) no bloco 32x32, por exemplo, é de apenas 0,22%, sendo que esta diferença é pequena, independente do bloco utilizado. Desta forma, utilizar a configuração TC 16, que compreende resíduos com valores no intervalo [-16, 16], é uma boa opção, uma vez que a tabela para armazenar esses resíduos é pequena, o que traz benefícios para a implementação em hardware. Essa abordagem atingiu uma alta taxa de compressão em blocos grandes, a partir de 32x32, atingindo mais de 65% de taxa de compressão.

A Figura 18 apresenta os resultados dos experimentos em termos de taxa de compressão para cada uma das configurações do algoritmo DDRFVLC. A maior taxa de compressão, para todos os tamanhos do bloco, é atingida com a utilização da tabela que contém todos os resíduos possíveis (TC 510). No entanto, para os tamanhos de bloco acima de 32x32, as tabelas de códigos na faixa igual ou superior a TC 16 apresentam taxa de compressão semelhantes. Por exemplo, comparando o TC 16 e TC 510 para o bloco de 32x32, a diferença na relação de compressão é de apenas 0,08% (TC 16: 71,16%; TC 510: 71,24%). Desta forma, estes experimentos indicam que tabelas menores podem ser utilizadas, com impactos quase insignificantes sobre as taxas de compressão.

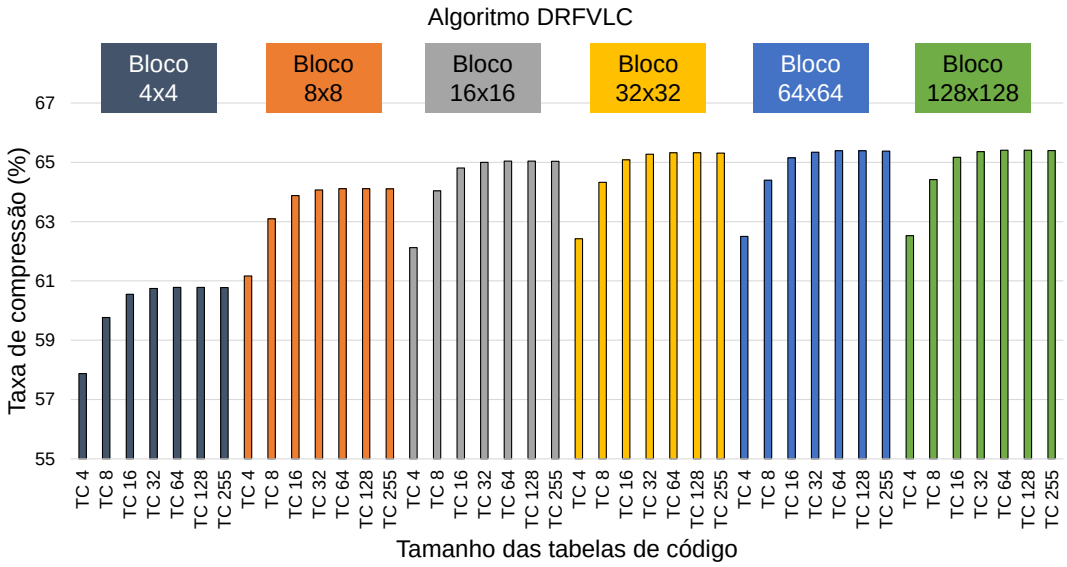


Figura 17 – Taxa de compressão para os 42 experimentos do algoritmo DRFVLC

Além disso, tabelas menores proporcionam menor consumo de recursos de hardware, conforme discutido anteriormente, quando comparado com configurações com tabelas maiores.

Quando o tamanho de bloco é analisado, os resultados mostram que para blocos de tamanho superior a 32x32, os resultados são muito semelhantes. Por exemplo, a configuração TC 32 com o bloco de tamanho 64x64 apresenta uma taxa de compressão 0,23% menor do que o bloco de tamanho 128x128 (64x64: 71,84%; 128x128: 72,07%). Portanto, para definir o melhor tamanho do bloco para o DDRFVLC, é necessário considerar os aspectos relacionados com a integração do compressor de quadros de referência com o codificador de vídeo e a organização de memória exigida (fragmentação de memória e de acesso aleatório).

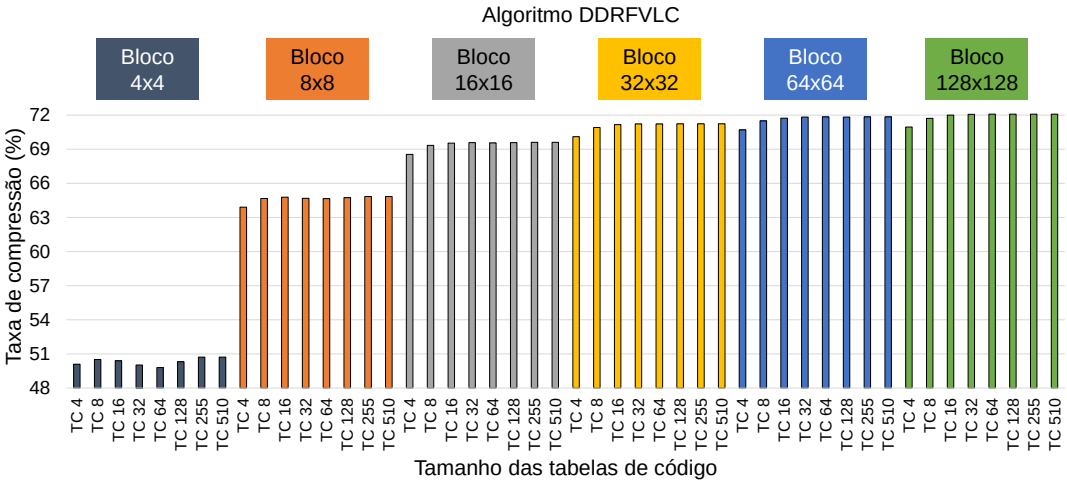


Figura 18 – Taxa de compressão para os 48 experimentos do algoritmo DDRFVLC

5.4 Considerações finais do capítulo

Neste capítulo foi apresentado o processo de codificação e decodificação dos três algoritmos desenvolvidos, destacando as suas diferenças, apresentando de forma detalhada os processos da codificação diferencial e da codificação de entropia utilizada pelos algoritmos. Os três algoritmos desenvolvidos para a compressão de quadros de referência, não inserem perdas de informação durante o processo de codificação e decodificação. Desta forma, a qualidade do vídeo codificado não é alterada.

O algoritmo DRFC é o algoritmo mais simples entre os três desenvolvidos neste trabalho. Esse algoritmo utiliza uma codificação diferencial simples com uma codificação de entropia que utiliza códigos de tamanho fixo, sendo o foco desse algoritmo a baixa utilização de recursos de hardware. A configuração TC 3 do DRFC, que compreende uma tabela com códigos de 3 bits representando informações de resíduos no intervalo de [-3, 3], foi a que apresentou as melhores taxas de compressão, e por isso essa configuração será utilizada no projeto de hardware desse algoritmo.

No algoritmo DRFVLC, a codificação diferencial simples é utilizada em conjunto com uma codificação de entropia de tamanho variável. Para isso, foi utilizado o algoritmo de *Huffman* para gerar os códigos de tamanho variável. Com esses códigos, foi possível explorar melhor os resíduos gerados pela codificação diferencial, que são centrados no valor zero.

O algoritmo DDRFVLC, introduz uma significativa contribuição para a literatura, uma vez que esse algoritmo utiliza uma codificação diferencial dupla, a qual explora a similaridade entre amostras vizinhas tanto na horizontal quanto na vertical. Deste modo, esse algoritmo é menos sensível a ocorrência de bordas dentro do bloco, a qual acaba gerando resíduos de altas frequências que diminuem a performance da codificação de entropia.

O tamanho de tabela que apresentou melhores resultados entre taxa de compressão e custo computacional é o TC 16, sendo que esse tamanho também apresentou bons resultados para o algoritmo DDRFVLC. Devido a isso, o projeto de hardware desses algoritmos irá utilizar essa configuração.

A taxa de compressão atingida pelos três algoritmos são altas, variando de 50% com o DRFC até 72% com o DDRFVLC. Entretanto, o custo computacional

desses algoritmos também varia. Com base nos resultados das taxas de compressão dos algoritmos, os tamanhos das tabelas de códigos definidas para serem utilizadas nas arquiteturas são TC 3 para o DRFC, que armazena 8 códigos, e TC 16 para os algoritmos DRFVLC e DDRFVLC, que armazena 34 códigos. A tabela TC 3 compreende o intervalo de resíduos [-3, 3], enquanto que a TC 16 compreende o intervalo [-16, 16]. Para os resíduos fora desses intervalos é utilizado um código de exceção. Devido a esses diferentes tamanhos de tabelas, o custo computacional das soluções pode variar bastante. Isso irá se refletir diretamente nos resultados arquiteturais, onde as soluções mais simples podem apresentar um desenvolvimento arquitetural bastante eficiente, principalmente o DRFC, que não utiliza comprimento variável nos códigos.

Dessa forma, para avaliar qual o algoritmo mais eficiente, deve-se desenvolver a arquitetura de hardware e analisar o consumo energético da solução. A eficiência energética é que vai mostrar qual a melhor solução: a solução que mais reduzir o consumo de energia nos acessos à memória, considerando o acréscimo no sistema gerado pela própria arquitetura do codec, será considerada a melhor.

No próximo capítulo serão apresentadas as arquiteturas de hardware desenvolvidas para os três algoritmos. Além disso, será apresentada a solução desenvolvida para o acesso aleatório aos dados codificados na memória, e a integração do codec com um sistema de codificação de vídeo.

6 ARQUITETURAS DE HARDWARE DESENVOLVIDAS

Neste capítulo serão apresentas as arquiteturas projetadas para os três algoritmos desenvolvidos neste trabalho e apresentados no capítulo anterior. Como o fluxo de codificação dos algoritmos é o mesmo, será apresentado primeiro o *template* arquitetural do codificador e decodificador e em seguida, serão discutidas as características específicas da implementação arquitetural de cada algoritmo. Além disso, será apresentada a estratégia utilizada para realizar os acessos aleatórios aos dados codificados, e como é feita a integração do codec desenvolvido com um codificador de vídeo HEVC. A seção 6.1 apresenta e discute as questões da integração do codec de quadros de referência e o codificador de vídeo, apresentando também a solução adotada para prover o acesso aleatório aos dados codificados na memória externa. As arquiteturas do codec foram descritas em linguagem de descrição de hardware VHDL, sendo esses módulos, codificador e decodificador, apresentados nas seções 6.2 e 6.3, respectivamente.

6.1 Integração do sistema e acesso aleatório à memória externa

A Figura 19 apresenta uma versão simplificada do codificador de vídeo HEVC (excluindo a predição intra que não é foco deste trabalho), incluindo o codec do compressor de quadros de referência. O codec situa-se entre a memória externa e o núcleo da ME. Assim, qualquer comunicação entre estes módulos é intermediada pelo codec. Todos os dados escritos na memória externa são codificados pelo codificador. Quando a ME solicita dados da memória externa, estas informações devem ser decodificadas pelo decodificador.

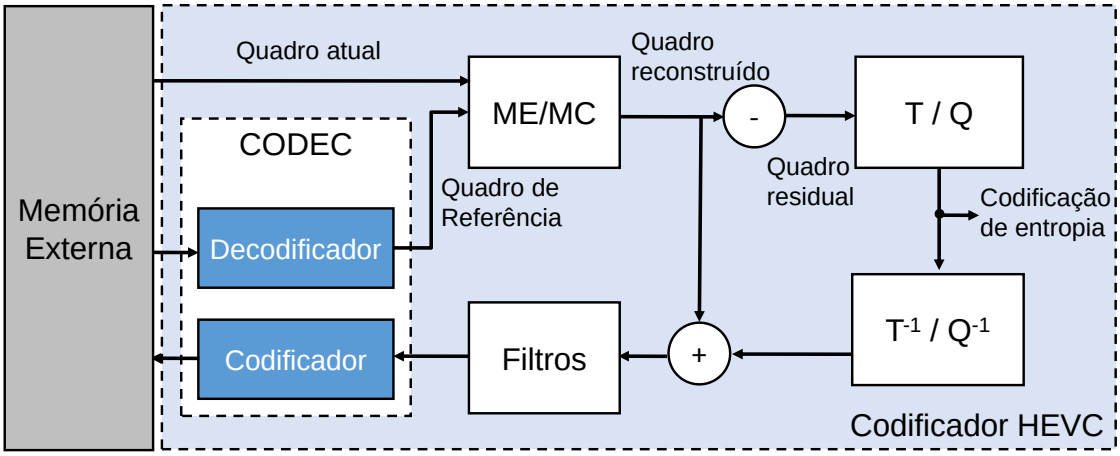


Figura 19 – Compressor de quadros de referência integrado a um sistema de codificação de vídeo

Os algoritmos de busca de ME podem realizar pesquisas em diferentes direções dentro dos quadros de referência, com o intuito de encontrar o bloco de menor resíduo em relação ao bloco que está sendo codificado. Devido a isso, este sistema requer uma organização de memória externa capaz de suportar o acesso aleatório aos blocos codificados, a fim de atender as necessidades da ME.

O tamanho do bloco utilizado no sistema de codificação é importante e tem um impacto significativo no desempenho do codificador de vídeo. O codificador de vídeo HEVC baseia-se em CTUs (*Coding Tree Unit*) com tamanho até 64x64 amostras, sendo esse tamanho o que apresenta melhores resultados de compressão no codificador HEVC (SULLIVAN, OHM, *et al.*, 2012). Isso significa que a janela de pesquisa da ME é deslocada 64 amostras para a direita cada vez que uma nova CTU começa a ser processada (CHEN, HUANG, *et al.*, 2006). Desse modo, utilizando blocos maiores do que 64x64 amostras, as informações excedentes terão que ser armazenadas. Por outro lado, se usarmos blocos menores que 64x64 amostras, estaremos limitando o desempenho do codificador de vídeo. Além disso, blocos grandes como 64x64, apresentam baixa fragmentação de memória e melhor aproveitamento dos blocos de memória externa.

Devido a esses fatos e aos bons resultados de compressão obtidos, as arquiteturas dos compressores de quadros de referência desenvolvidas utilizam blocos com 64x64 amostras. Assim, o tamanho de bloco escolhido atinge altas taxas de compressão, como visto no Capítulo 5, e possibilita uma fácil integração com o codificador HEVC, sem custos adicionais com memórias temporárias.

A Figura 20 apresenta a organização proposta para a memória externa, otimizada para vídeos de resolução de HD 1080p. No entanto, esta mesma estratégia pode ser utilizada com foco em outras resoluções. Duas regiões da memória externa são usadas para endereçamento: a memória regular e a memória auxiliar. A memória regular é dividida em 510 partições, onde cada partição é usada para armazenar um bloco de 64x64 amostras codificadas. Desta forma, todo o quadro de referência (1920x1088 amostras) pode ser armazenado.

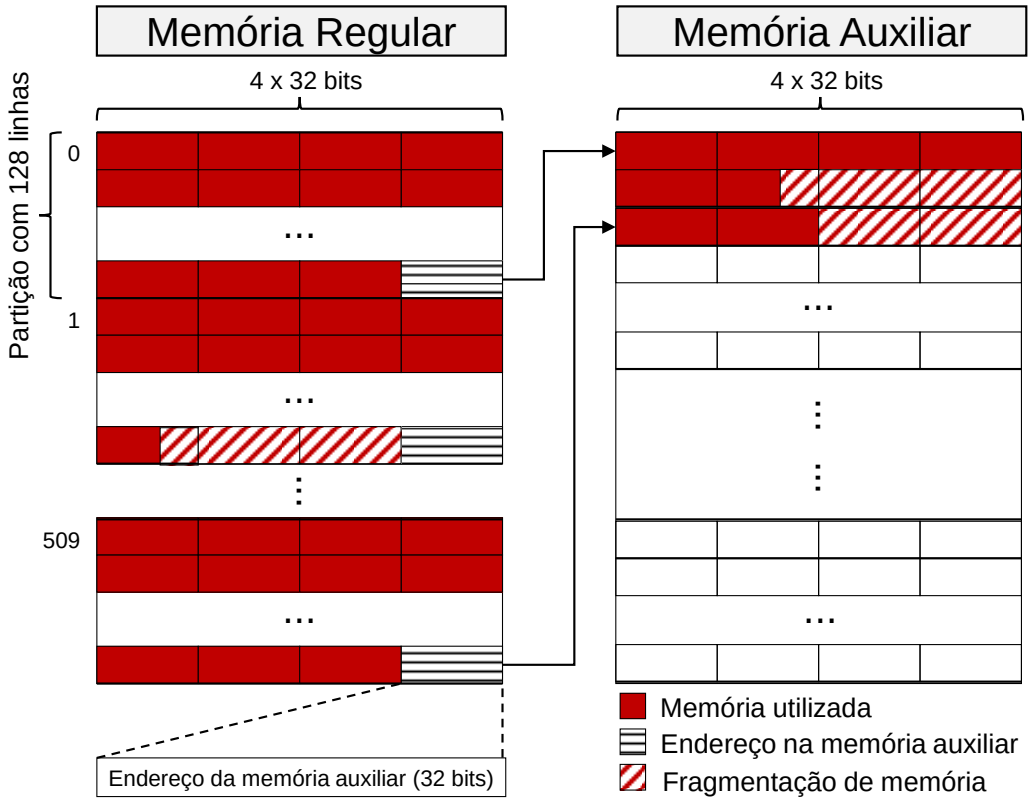


Figura 20 – Organização da memória externa

Cada partição é composta de 512 palavras de 32 bits, com quatro palavras por linha, num total de 128 linhas por partição. Um total de 511 palavras são utilizadas para armazenar as amostras codificadas e uma palavra adicional é usada para armazenar o endereço de uma região na memória auxiliar, caso seja necessária. Quando o bloco codificado ultrapassa o limite de 511 palavras na memória regular, o excesso de informações é enviado à memória auxiliar. O endereço para esta área na memória auxiliar fica escrito na última palavra da partição da memória regular, representado pelos retângulos texturizados (linhas horizontais) na Figura 20.

A memória auxiliar tem partições de tamanho variável, onde cada partição é composta por um número inteiro de linhas de 128 bits, sendo o tamanho mínimo de

uma linha com quatro palavras de 32 bits. Assim, a fragmentação nessa memória é reduzida, uma vez que a memória auxiliar não tem tamanho fixo de partição.

A Figura 21 mostra o histograma de palavras de 32 bits utilizadas pelos blocos codificados. O histograma foi gerado considerando as mesmas oito sequências de vídeo utilizadas nos experimentos apresentados no capítulo 4 e foi gerado através do algoritmo DDRFVLC para blocos 64x64 e tabela TC 32. Este estudo forneceu suporte para definir os tamanhos da memória externa e das partições. Em média, 347 palavras de 32 bits são suficientes para armazenar um bloco codificado nesta configuração. No entanto, esta quantidade de palavras gera um elevado número de acessos à memória auxiliar. Devido a isso, definiu-se um número maior de palavras para ser utilizado. Com 512 palavras por partição, 88,7% dos blocos codificados são armazenados apenas na memória regular, e 11,3% dos blocos codificados usam alguma área na memória auxiliar. Se nenhuma técnica de compressão é usada, seriam necessárias 1.024 palavras de 32 bits para armazenar um bloco de 64x64. Utilizando o esquema proposto, obtemos uma redução média de 44% em memória externa física (considerando a fragmentação de memória) usada para armazenar os quadros de referência. No entanto, em termos de largura de banda de memória externa, esta redução é de cerca de 68%, conforme demonstrado no capítulo 5.3.

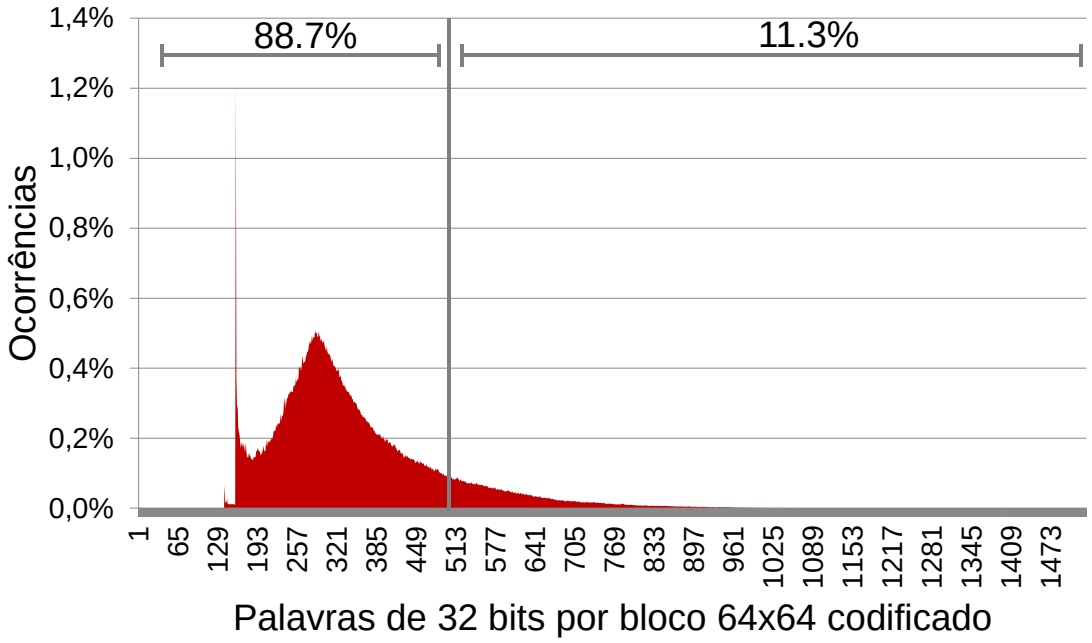


Figura 21 – Histograma de palavras de 32 bits por bloco 64x64 codificado

Uma análise do pior caso teórico neste sistema foi realizada. Para isso, foi considerado que cada amostra codificada utiliza o tamanho máximo em bits para ser codificado. Por exemplo, no caso do DDRFVLC a maior amostra codificada tem 18 bits de tamanho, 9 bits do resíduo mais 9 bits do código de exceção. Desta forma, todas as partições da memória regular seriam utilizadas por completo e mais uma área na memória auxiliar. No total, esse quadro em resolução HD 1080p ocuparia 4,5MB de memória externa. No entanto, esse tamanho não seria um problema em termos de perda de dados, nem de organização de memória externa, porque com a utilização da memória auxiliar, que tem partições de tamanho variável, estas informações podem ser facilmente alocadas. Vale salientar que este pior caso teórico nunca ocorreu em nenhum dos testes realizados, onde mais de 5 milhões de blocos 64x64, das 8 sequências de vídeo, foram avaliados.

Devido à utilização de memória compartilhada, e por permitir acesso aleatório a blocos individuais, esta abordagem pode ser utilizada em um sistema com múltiplas unidades do Codec de quadros de referência funcionando em paralelo. Desta forma, o desempenho do Codec pode aumentar em tantas vezes quanto as unidades do Codec forem utilizadas, sem inserir um novo gargalo no sistema. Além disso, o sistema proposto pode ser utilizado em codificadores HEVC que utilizem as ferramentas de paralelização Tiles (SULLIVAN, OHM, *et al.*, 2012) ou WPP (*Wavefront Parallel Processing*) (SULLIVAN, OHM, *et al.*, 2012) sem a necessidade de modificações.

6.2 Arquiteturas do codificador

A Figura 22 apresenta o *template* arquitetural de hardware desenvolvido para o codificador. As arquiteturas que implementam os três algoritmos desenvolvidos utilizam este mesmo *template* arquitetural, sendo que todas operam sobre blocos de tamanho 64x64. Os módulos que apresentam variações significativas, dependendo do algoritmo implementado, são: o banco de registradores de entrada, a codificação diferencial e a tabela estática. Nessa figura, o módulo da codificação diferencial é destacado com a codificação diferencial dupla (CDD) e a codificação diferencial simples (CDS). O primeiro módulo utilizado é o Banco de Registradores. Esse módulo é utilizado para armazenar as amostras de entrada que serão utilizadas na codificação diferencial. Nos algoritmos de codificação diferencial simples, DRFC e DRFVLC, esse módulo é composto por

apenas dois registradores de 8 bits. No caso da dupla codificação diferencial, DDRFVLC, esse módulo deverá armazenar uma coluna inteira de um bloco, e portanto, serão necessários 64 registradores de 8 bits cada.

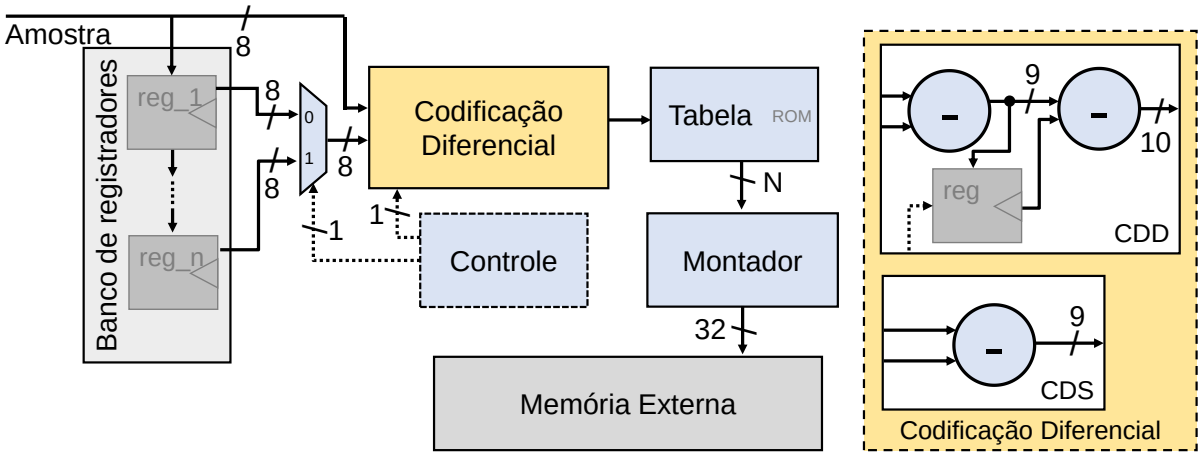


Figura 22 – *Template* arquitetural do codificador

Para cada bloco codificado, o primeiro registrador é definido para zero. Isto é feito para preservar o valor da primeira amostra do bloco. Deste modo, não é necessário um processo especial para a primeira amostra. As entradas para este módulo são sempre números positivos, uma vez que as amostras de luminância variam de 0 a 255, utilizando 8 bits por amostra. O multiplexador faz a seleção entre a amostra da mesma coluna ou da coluna vizinha, e após, envia a amostra selecionada para o módulo da codificação diferencial.

O módulo da codificação diferencial é composto por um ou dois subtratores, como pode ser visto na Figura 22. Na codificação diferencial simples (CDS) é utilizado um subtrator de 9 bits para não ocorrer overflow no cálculo. O CDS realiza a subtração entre a amostra atual e a amostra selecionada pelo multiplexador. Já na codificação diferencial dupla (CDD), dois subtratores são utilizados em série, um subtrator de 9 bits e um subtrator de 10 bits. Além disso, um registrador de 9 bits é utilizado para armazenar o primeiro resíduo. A primeira codificação diferencial é realizada entre a amostra atual e a amostra selecionada pelo multiplexador. A segunda codificação diferencial é realizada entre o resíduo atual e o resíduo anterior, que é armazenado no registrador de 9 bits. Os resíduos gerados pela codificação diferencial são enviados para o módulo Tabela.

Na Tabela, o resíduo é traduzido para os códigos correspondentes. No DRFC os códigos são de tamanhos fixos, ao passo que no DRFVLC e DDRFVLC são utilizados códigos de *Huffman* de tamanho variável (VLC). Este módulo é

composto por uma pequena memória ROM que armazena os códigos de tamanho fixo ou VLC, e o código de exceção utilizado para codificar os resíduos fora do intervalo especificado, uma vez que nenhum dos algoritmos considera a tabela completa. Se o código de exceção é utilizado, o resíduo é apenas anexado ao código de exceção. A tabela também contém o tamanho, em bits, dos códigos para cada amostra. Essa informação é utilizada pelo módulo Montador para escrever corretamente o código no seu buffer interno. O tamanho da saída (*N*) vai depender dos códigos utilizados pelo módulo. O Apêndice B apresenta os códigos utilizados nas tabelas estáticas para os três algoritmos.

O módulo Montador é utilizado para normalizar a saída do codificador para a memória externa, uma vez que o codificador trabalha com codificação de comprimento variável, e memórias comerciais funcionam com palavras de tamanho fixo. Desta forma, sempre que 32 bits contínuos de informações são escritos no buffer circular, esta palavra é escrita na memória externa. O Montador foi projetado como um buffer circular, sendo o tamanho deste buffer definido pela soma do tamanho do maior código VLC com 32 que é o tamanho da saída. Assim, se o maior código possuir 18 bits o buffer circular deverá ter 50 bits, porque o pior caso envolve escrever a palavra mais longa (18 bits) com a saída do codificador pronta para ser escrita na memória externa (32 bits).

O Controle possui uma máquina de estados finitos que define as amostras certas a serem subtraídas na codificação diferencial, os estados seguem os casos apresentados nas Equações 7 e 8. Além disso, o Controle especifica o tamanho do bloco utilizado. Dessa forma o início e o fim do bloco são conhecidos, essas são informações importantes, porque após a última amostra do bloco ser codificada, esta informação é escrita na memória de referência; mesmo que o montador não esteja com a palavra de 32 bits completa (os outros bits não serão utilizados). Este processo gera uma fragmentação na memória externa; no entanto, permite que seja utilizado o acesso aleatório à memória, uma vez que não existem dependências de dados entre os blocos codificados, conforme apresentado na seção anterior.

Enquanto a operação de escrita é realizada, o espaço restante no buffer é utilizado para receber mais entradas do codificador. Ao fazer isso, o sistema não fica parado durante as operações de escrita na memória. Todo esse processo de codificação foi projetado para usar um ciclo de relógio por amostra.

6.3 Arquiteturas do decodificador

A Figura 23 apresenta o diagrama de blocos do *template* arquitetural projetado para o decodificador, utilizado para os três algoritmos. Inicialmente, os blocos codificados são obtidos a partir da memória externa. O componente Parser é responsável por normalizar a entrada para o módulo Tradutor. O Parser recebe uma palavra de 32 bits como entrada e tem um buffer interno que funciona como uma fila para armazenar as entradas. Este buffer é necessário, uma vez que as palavras oriundas da memória externa apresentam um comprimento variável. Este componente recebe também informações da unidade de Controle sobre o tamanho real da amostra codificada. Esta informação é importante, uma vez que é utilizada para atualizar o buffer interno desse componente, a fim de gerar a próxima palavra de forma correta. A saída do Parser (*N*) é um código com tamanho do maior código presente na tabela. Este código é enviado para o módulo Tradutor. Além disso, o Parser emite um sinal para a unidade de Controle, informando se o buffer tem espaço suficiente para uma nova entrada.

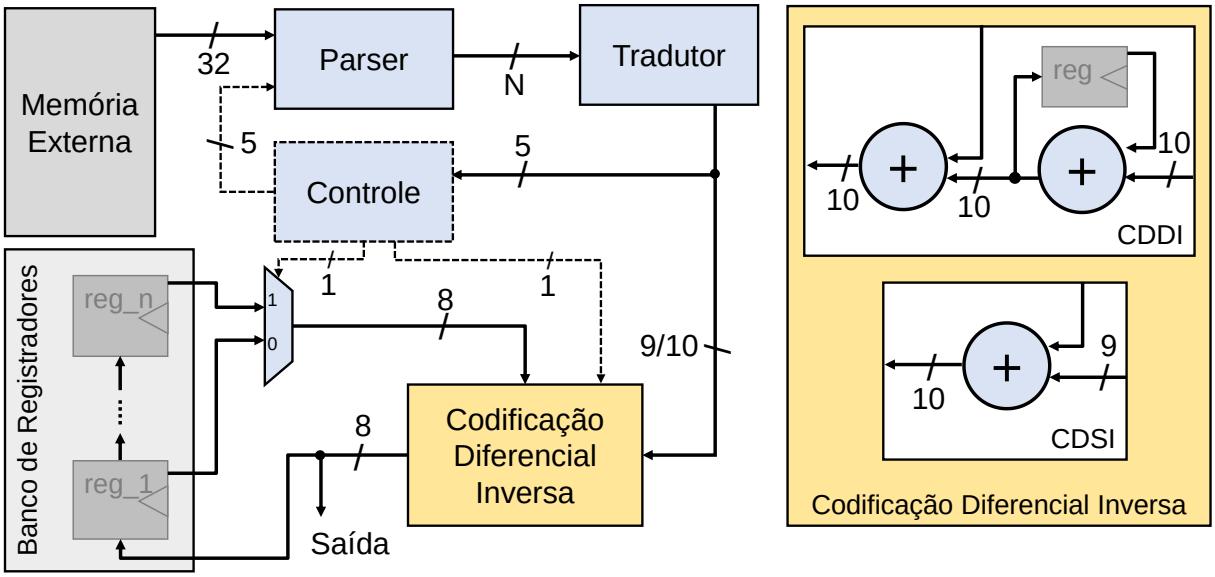


Figura 23 – *Template* arquitetural do decodificador

O processo de decodificação do código é executado pelo módulo Tradutor. Esse módulo recebe a entrada (*N*) que representa um resíduo codificado. A entrada é decodificada de acordo com a tabela de decodificação. Esta tabela está localizada no módulo Tradutor e foi projetada usando memórias ROM, assim como no codificador. A tabela de decodificação também contém o tamanho, em bits, da palavra codificada. Essas informações são retornadas ao Parser através da unidade

de Controle, de modo que o *buffer* interno ao Parser é atualizado para a posição correta e, em seguida, uma nova amostra pode ser decodificada. A saída do módulo Tradutor é o resíduo com um tamanho de 9 ou 10 bits, o que depende da codificação diferencial utilizada, simples ou dupla, e esse resíduo é enviado ao módulo da codificação diferencial inversa.

A Figura 23, mostra o módulo da codificação diferencial inversa, esse módulo utiliza uma codificação diferencial simples inversa (CDSI) ou a codificação diferencial dupla inversa (CDDI), sendo também esses dois módulos apresentados na Figura 23.

A CDSI é composta por um somador de 9 bits, o qual executa a soma entre o resíduo atual e a amostra decodificada selecionada pelo multiplexador. A CDDI é composta por dois somadores em série, um somador de 10 bits e um somador de 9 bits. A codificação diferencial dupla inversa é realizada em dois passos: o primeiro passo executa a soma entre o resíduo decodificado e o resíduo anterior, que está armazenado no registrador. No segundo passo, a soma é realizada entre o resíduo atual e a amostra selecionada pelo multiplexador. Após isso, a amostra original é gerada e enviada para a saída e para o módulo do Banco de Registradores.

O módulo do Banco de Registradores é utilizado para armazenar duas amostras decodificadas, uma da mesma coluna e outra da coluna vizinha, no caso dos algoritmos de codificação diferencial simples, ou uma coluna inteira no caso de utilizar a dupla codificação diferencial. Todo este processo, assim como no codificador, foi projetado para usar um ciclo de relógio por amostra.

6.4 Considerações finais do capítulo

Nesse capítulo foi apresentada a estratégia de integração do codec de quadros de referência com o codificador de vídeo. Essa estratégia envolve a comunicação entre a ME e a memória externa, apresentando em detalhes como é realizado o acesso aleatório aos blocos codificados armazenados na memória externa. Dessa forma, todos os dados escritos na memória externa são codificados pelo codificador. Quando a ME solicita dados da memória externa estas informações devem ser decodificadas pelo decodificador. Por estar inserido em um gargalo de desempenho dos codificadores de vídeo, existem grandes desafios relacionados ao desenvolvimento de um codec de quadros de referência que reduza de forma efetiva a largura de banda, e consequentemente, o consumo energético na comunicação

com a memória externa, sem inserir latência ou perdas de dados no processo de codificação.

A importante questão do tamanho do bloco utilizado na codificação de vídeo foi abordada e serviu como definidor na escolha do tamanho de bloco utilizado pelo codec de quadros de referência. Uma vez que o codificador de vídeo HEVC utiliza CTUs com tamanho de até 64x64 amostras, e considerando os bons resultados de compressão obtidos com os algoritmos desenvolvidos, as arquiteturas desenvolvidas para o compressor de quadros de referências utilizam blocos com 64x64 amostras.

As arquiteturas do codec foram projetadas pensando na eficiência da utilização dos recursos de hardware e do consumo de energia. Isto se reflete na quantidade de registradores usados, nos tamanhos das tabelas estáticas e na quantidade de tamanhos dos códigos VLC, sendo que esse último tem grande impacto no decodificador, uma vez que códigos VLC geram alta dependência de dados entre os códigos. Na decodificação de tamanho variável, não há nenhuma informação explícita para detectar o início ou o fim do código, exigindo assim uma leitura bit a bit da informação codificada, inserindo grande dependência de dados, ou um processamento paralelo de todos tamanhos, inserindo alta dissipação de potência. Desta forma, tabelas menores com código de exceção trazem grandes benefícios.

No próximo capítulo serão apresentados os resultados obtidos para os algoritmos desenvolvidos, tanto em software quanto em hardware. Para as implementações em software, os resultados de taxa de compressão serão apresentados em detalhes. Os resultados de síntese das arquiteturas desenvolvidas também serão apresentados, onde as questões relacionadas ao consumo de energia e eficiência energética das soluções serão discutidas em detalhes. Além disso, uma comparação completa com os trabalhos estado da arte, envolvendo resultados de software e hardware será apresentada.

7 RESULTADOS E COMPARAÇÕES COM TRABALHOS RELACIONADOS

Os algoritmos apresentados foram desenvolvidos com o objetivo de comprimir os quadros de referência de forma eficiente e sem perdas de qualidade, reduzindo o consumo de energia resultante da comunicação entre memória externa e a ME, através da redução do tráfego de dados. Os resultados de taxa de compressão das soluções desenvolvidas serão detalhados na seção 7.1. Esses resultados foram obtidos através da codificação de 17 sequências de vídeo HD 1080p, considerando informações de luminância e crominância dos quadros. No entanto, para avaliar os ganhos com a compressão, é obrigatória que a implementação dos módulos codificador e decodificador seja energeticamente eficiente. Desta forma, as arquiteturas de hardware dedicadas que foram apresentadas no capítulo anterior foram sintetizadas para tecnologia ASIC *standard cells*, sendo esses resultados apresentados na seção 7.2. Na seção 7.3, são apresentados os resultados de redução de potência nos acessos à memória para as soluções desenvolvidas. Por fim, na seção 7.4, são realizadas as comparações dos resultados de software e hardware com os trabalhos mais relevantes encontrados na literatura.

7.1 Taxa de compressão dos algoritmos

Nos capítulos anteriores foi discutido que algumas configurações dos algoritmos geraram melhores resultados de taxa de compressão e custo computacional do que outras configurações. Desta forma, nessa seção serão apresentados os resultados detalhados dessas configurações. O algoritmo DRFC utiliza uma tabela de códigos com tamanho fixo de 3 bits. Essa tabela tem apenas 8 posições (TC3), e cobre o intervalo de representação de resíduos [-3,3]. Os algoritmos DRFVLC e DDRFVLC utilizam uma tabela com códigos de tamanho variável. Esses códigos possuem uma variação de tamanho de 1 bit até 18 bits. A tabela estática desses dois algoritmos possui 34 posições (TC16) e cobre o intervalo

[-16,16]. O tamanho das tabelas foi definido após a análise com os experimentos apresentados na seção 5.3. O tamanho de bloco utilizado pelos três algoritmos é de 64x64 amostras, sendo esse o tamanho que melhor se integrou ao sistema, gerando baixas taxas de fragmentação na memória externa. Além de apresentar bons resultados de taxa de compressão.

Para os resultados detalhados de taxa de compressão dos algoritmos, um conjunto com 17 sequências HD 1080p foram avaliados. Os experimentos foram realizados utilizando sequências de vídeo reconstruídos obtidos através do software HM-12.0, considerando a configuração de acesso aleatório, algoritmo de busca *TZ Search* e os parâmetros de quantização (QPs) 22, 27, 32 e 37, de acordo com as condições comuns de teste (CCT) do HEVC (BOSSEN, 2012). Como as CCTs definem apenas cinco sequências HD 1080p, outras 12 sequências foram usadas para estender os experimentos.

As Tabela 4, 5 e 6 apresentam os resultados de taxa de compressão de forma detalhada para os três algoritmos desenvolvidos, DRFC, DRFVLC e DDRFVLC, respectivamente. Os resultados são apresentados para cada uma das 17 sequências de vídeo avaliadas, separados pelo valor do QP e pelos componentes de luminância e crominância (Y - luminância; Cb e Cr - crominância).

A Tabela 4 apresenta os resultados detalhados de software do algoritmo DRFC, onde é possível observar que a taxa de compressão varia entre os vídeos. Isso acontece porque as sequências de vídeo apresentam características de textura distintas, com diferentes quantidades de regiões homogêneas e regiões texturizadas. Assim, ao comprimir áreas homogêneas, uma taxa de compressão melhor é alcançada, uma vez que os resíduos obtidos pelo algoritmo estão perto de zero. Este fato pode ser observado na sequência *Man in the Car*, que é quase totalmente dominado por regiões homogêneas. No entanto, regiões texturizados tornam a compressão mais difícil, uma vez que as diferenças entre as amostras tendem a estarem mais distantes de zero, e com isso as taxas de compressão diminuem, como pode ser observado, por exemplo, na sequência *BQ Terrace*.

Além disso, os componentes de crominância apresentam maior taxa de compressão em comparação com os componentes de luminância. Isso ocorre porque os componentes de crominância são compostos principalmente de regiões homogêneas. Além disso, com o aumento do valor QP, a taxa de compressão melhora e o desvio padrão diminui. Isso acontece porque QPs maiores produzem

vídeos codificados com texturas mais suaves – o QP maior atenua altas frequências – aumentando a eficiência da codificação diferencial.

Tabela 4 – Taxa de compressão do DRFC para os 17 vídeos avaliados

Vídeo	QP 22 – (%)			QP 27 – (%)			QP 32 – (%)			QP 37 – (%)			Média QPs – (%)		
	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
*Basketball Drive	46,6	59,2	57,9	48,7	59,7	58,3	49,9	60,2	58,6	51,0	60,5	58,8	49,0	59,9	58,4
*BQ Terrace	24,4	58,3	60,8	33,4	59,5	61,3	36,0	60,3	61,6	37,7	60,8	61,8	32,9	59,7	61,4
*Cactus	37,4	54,1	55,3	40,2	56,7	56,0	42,3	57,7	56,5	60,4	60,5	60,6	45,1	57,3	57,1
*Kimono	48,8	60,6	60,7	49,9	61,2	61,4	51,3	61,5	61,9	53,0	61,8	62,1	50,8	61,3	61,5
*Park Scene	34,7	53,4	57,1	37,3	56,1	59,9	40,2	58,1	61,3	43,4	59,5	61,8	38,9	56,7	60,0
Média CCT	**38,4**	**57,1**	**58,3**	**41,9**	**58,6**	**59,4**	**43,9**	**59,5**	**60,0**	**49,1**	**60,6**	**61,0**	**43,3**	**59,0**	**59,7**
Média 4:2:0		**44,8**			**47,6**			**49,2**			**53,0**			**48,7**	
Des. Pad. CCT	**8,8**	**2,8**	**2,1**	**6,4**	**1,9**	**2,0**	**5,8**	**1,4**	**2,1**	**7,9**	**0,7**	**1,2**	**6,6**	**1,7**	**1,7**
Blue Sky	40,0	42,3	49,7	40,1	43,2	51,1	40,1	44,5	52,5	40,0	46,4	54,3	40,0	44,1	51,9
In to Tree	47,8	48,9	49,2	51,6	53,0	53,1	54,6	54,2	54,5	57,9	55,1	55,6	53,0	52,8	53,1
Man in Car	60,7	62,4	62,4	60,8	62,4	62,4	60,9	62,4	62,4	61,0	62,4	62,4	60,9	62,4	62,4
Pedestrian Area	50,9	60,3	61,0	51,6	60,8	61,2	52,0	61,1	61,5	52,2	61,5	61,6	51,7	60,9	61,3
Riverbed	48,4	62,0	62,1	50,4	62,2	62,3	52,9	62,3	62,4	54,6	62,3	62,5	51,6	62,2	62,3
Rolling Tomatoes	60,6	62,0	62,1	61,0	62,1	62,1	61,0	62,0	62,0	61,0	62,1	62,0	60,9	62,1	62,1
Station 2	44,5	61,5	61,3	47,4	62,0	61,8	49,8	62,2	62,0	51,9	62,3	62,2	48,4	62,0	61,8
Sun Flower	48,3	48,6	55,3	48,7	49,3	56,1	49,0	50,0	56,8	49,3	50,5	57,4	48,8	49,6	56,4
Tennis	51,0	61,1	60,2	51,6	61,5	61,0	52,5	61,6	61,4	53,5	61,8	61,6	52,1	61,5	61,0
Tractor	50,4	57,4	57,2	51,2	58,3	57,5	52,0	59,1	57,9	52,8	59,7	58,2	51,6	58,6	57,7
People On Street	34,1	58,8	60,1	36,6	59,6	60,5	38,2	59,9	60,8	38,9	60,2	61,0	37,0	59,6	60,6
Traffic	40,3	53,1	57,6	41,4	55,5	59,1	42,6	57,4	60,1	43,7	58,8	60,7	42,0	56,2	59,4
Média Todos	**45,2**	**56,7**	**58,2**	**47,2**	**57,8**	**59,1**	**48,5**	**58,5**	**59,7**	**50,7**	**59,2**	**60,3**	**47,9**	**58,0**	**59,3**
Média 4:2:0		**49,3**			**50,9**			**52,1**			**53,7**			**51,5**	
Des. Pad. Todos	**9,1**	**5,6**	**3,9**	**7,7**	**5,1**	**3,3**	**7,3**	**4,7**	**2,9**	**7,4**	**4,3**	**2,4**	**7,5**	**4,9**	**3,1**

* vídeos da CCT

Os comportamentos observados na Tabela 4, como variação na taxa de compressão dos vídeos e a melhor taxa de compressão atingida pelos componentes de crominância, são também observados na Tabela 5, que traz os resultados de software do DRFVLC, e na Tabela 6 que apresenta os resultados de software do DDRFVLC. Isto se deve ao fato de que os algoritmos seguem a mesma forma de codificação, com codificação diferencial e codificação de entropia baseada em dicionário de códigos. A variação ocorre na taxa de compressão obtida por cada um.

Os resultados apresentados na Tabela 4 mostram que o DRFC atinge uma taxa de compressão média de 51,5%, considerando todos os vídeos e subamostragem de cor 4:2:0. Para os vídeos das CCTs a taxa de compressão é 2,8% menor, ficando em média 48,7%. Quando são consideradas apenas as informações de luminância, a taxa de compressão atingida pelo DRFC é de 47,9% para todos os vídeos e de 43,3% para os vídeos das CCTs.

A Tabela 5 apresenta os resultados de software da taxa de compressão para o algoritmo DRFVLC. É possível observar nessa tabela que a taxa de compressão obtida é maior quando comparado aos resultados do DRFC. Esse resultado é

independente do vídeo ou do QP utilizado. Esses dois algoritmos utilizam o mesmo tipo de codificação diferencial e o mesmo tamanho de bloco, porém, o DRFVLC utiliza códigos de tamanho variável de *Huffman*, sendo esse o motivo principal da melhora nas taxas de compressão. Com os códigos de tamanho variável, o algoritmo obtém melhor aproveitamento dos resíduos de valor zero e próximos de zero. Esses valores de resíduo dominam a distribuição de um vídeo, como pôde ser observado nos histogramas apresentados no capítulo 4.

A taxa de compressão atingida pelo DRFVLC variou de 57,2% a 65% para quadros 4:2:0 nos vídeos das CCTs, sendo que a média foi de 61,2% para esses vídeos. Isso representa uma taxa de compressão 12,5% maior que o DRFC. Para quadros de luminância, a taxa de compressão atingida foi de 53,5% para os vídeos das CCTs e de 59,1% quando todos os vídeos são considerados. Isso representa um aumento de 11,2% quando comparado ao DRFC.

Tabela 5 – Taxa de compressão do DRFVLC para os 17 vídeos avaliados

Vídeo	QP 22 – (%)			QP 27 – (%)			QP 32 – (%)			QP 37 – (%)			Média QPs – (%)		
	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
*Basketball Drive	57,9	76,5	75,4	60,3	77,2	75,8	61,7	78,0	76,4	62,9	78,7	77,0	60,7	77,6	76,1
*BQ Terrace	34,5	74,1	79,5	42,3	76,1	80,7	46,3	77,8	81,4	49,6	78,8	82,3	43,2	76,7	81,0
*Cactus	48,3	67,3	71,1	51,5	71,8	72,2	53,3	73,7	72,9	55,2	84,4	84,6	52,1	74,3	75,2
*Kimono	59,2	74,4	77,1	60,2	76,6	79,7	61,6	78,2	81,5	63,4	79,8	82,8	61,1	77,2	80,3
*Park Scene	46,0	66,6	70,7	49,2	70,5	75,3	52,1	73,1	78,4	55,1	75,4	80,4	50,6	71,4	76,2
Média CCT	**49,2**	**71,8**	**74,8**	**52,7**	**74,4**	**76,7**	**55,0**	**76,2**	**78,1**	**57,2**	**79,4**	**81,4**	**53,5**	**75,4**	**77,8**
Média 4:2:0		**57,2**			**60,3**			**62,4**			**65,0**			**61,2**	
Des. Pad. CCT	**9,0**	**4,0**	**3,4**	**6,9**	**2,8**	**3,1**	**5,9**	**2,3**	**3,3**	**5,2**	**2,9**	**2,6**	**6,7**	**2,3**	**2,4**
Blue Sky	54,1	59,8	67,3	54,0	60,7	68,4	53,6	61,6	69,5	53,5	63,0	71,0	53,8	61,3	69,1
In to Tree	60,0	58,4	59,8	66,7	63,4	64,5	70,3	66,6	67,4	73,7	69,3	70,2	67,7	64,4	65,5
Man in Car	78,9	82,9	84,7	80,4	83,8	84,6	80,5	84,3	84,8	80,5	84,5	85,0	80,1	83,9	84,8
Pedestrian Area	63,1	77,3	78,7	64,0	78,4	79,3	64,4	79,4	79,9	64,6	80,4	80,4	64,0	78,8	79,6
Riverbed	54,8	76,1	81,5	55,6	79,8	84,0	57,9	81,9	85,0	60,3	82,8	85,5	57,2	80,1	84,0
Rolling Tomatoes	74,3	80,4	82,1	76,5	81,2	81,5	76,2	81,6	81,4	75,7	81,6	81,3	75,7	81,2	81,6
Station 2	54,5	76,2	77,0	57,2	79,0	79,1	59,8	80,2	80,6	62,6	81,7	82,4	58,5	79,2	79,8
Sun Flower	55,9	56,1	65,8	55,8	56,8	66,9	55,8	57,4	67,9	56,2	58,1	68,7	55,9	57,1	67,3
Tennis	64,8	78,7	78,6	65,3	79,7	79,7	66,0	80,5	80,6	67,2	81,0	81,1	65,8	80,0	80,0
Tractor	59,9	68,3	69,6	59,3	69,0	70,0	59,7	70,3	70,9	60,3	71,7	71,9	59,8	69,8	70,6
People On Street	43,1	73,9	76,9	46,1	74,3	77,5	48,3	74,6	77,8	49,8	75,3	78,3	46,8	74,5	77,6
Traffic	50,5	66,4	73,8	51,8	69,5	76,5	53,2	72,2	78,5	54,4	74,5	80,1	52,5	70,6	77,2
Média Todos	**56,5**	**71,4**	**74,7**	**58,6**	**73,4**	**76,2**	**60,0**	**74,8**	**77,3**	**61,5**	**76,5**	**79,0**	**59,1**	**74,0**	**76,8**
Média 4:2:0		**62,0**			**64,0**			**65,4**			**66,9**			**64,6**	
Des. Pad. Todos	**10,4**	**7,7**	**6,3**	**9,7**	**7,4**	**5,8**	**9,0**	**7,2**	**5,5**	**8,6**	**7,2**	**5,2**	**9,3**	**7,2**	**5,5**

* vídeos da CCT

Na Tabela 6 são apresentados os resultados de taxa de compressão do DDRFVLC. Como pode ser visto nesta tabela, o DDRFVLC atinge altas taxas de compressão independente do QP utilizado. A taxa de compressão média atingida para vídeos das CCTs é de 67,9%, para quadros 4:2:0, e 62,4% para quadros de

luminância. Essa taxa de compressão é 8,9% maior quando comparado ao DRFVLC. Considerando todos os 17 vídeos esta taxa de compressão é ainda maior, atingindo 70,7% para quadros 4:2:0 e 67,5% para quadros de luminância, o que representa um aumento de 8,4% em relação ao DRFVLC. Esse melhor desempenho obtido pelo DDRFVLC se deve ao fato de que essa solução utiliza a dupla codificação diferencial, a qual reduz as altas frequências tanto de bordas verticais quanto de bordas horizontais, gerando resíduos com maior incidência de zeros ou de valores próximos a zero.

O custo computacional desses dois algoritmos é praticamente o mesmo, pois o tamanho de bloco e o tamanho da tabela são os mesmos. No entanto, o DDRFVLC realizada uma codificação diferencial a mais que no DRFVLC. Além disso, os valores de desvio padrão são bastante semelhantes aos valores do DRFVLC. Isso mostra que a taxa de compressão aumentou para todos os vídeos de maneira uniforme, indicando que o DDRFVLC produz resultados consistentes.

Tabela 6 – Taxa de compressão do DDRFVLC para os 17 vídeos avaliados

Vídeo	QP 22 – (%)			QP 27 – (%)			QP 32 – (%)			QP 37 – (%)			Média QPs – (%)		
	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
*Basketball Drive	64,5	79,3	78,6	69,1	79,8	79,0	70,5	80,5	79,6	71,5	81,1	80,2	68,9	80,2	79,4
*BQ Terrace	39,1	76,0	80,8	52,5	78,5	81,9	57,0	80,1	82,6	59,4	80,9	83,3	52,0	78,9	82,2
*Cactus	55,3	70,2	75,5	60,9	76,2	76,8	63,8	77,9	77,4	66,7	83,5	83,7	61,7	77,0	78,4
*Kimono	68,9	76,7	78,3	70,0	78,7	80,6	71,3	80,2	82,2	72,8	81,2	83,2	70,8	79,2	81,1
*Park Scene	52,0	70,3	73,6	56,4	74,2	77,6	60,4	76,6	80,0	65,0	78,6	81,6	58,5	74,9	78,2
Média CCT	**56,0**	**74,5**	**77,4**	**61,8**	**77,5**	**79,2**	**64,6**	**79,1**	**80,4**	**67,1**	**81,1**	**82,4**	**62,4**	**78,0**	**79,8**
Média 4:2:0		**62,6**			**67,3**			**69,6**			**72,0**			**67,9**	
Des. Pad. CCT	**10,4**	**3,6**	**2,5**	**6,9**	**2,0**	**1,9**	**5,6**	**1,5**	**1,9**	**4,8**	**1,6**	**1,3**	**6,9**	**1,9**	**1,5**
Blue Sky	59,2	60,4	68,3	59,9	63,1	71,0	60,6	66	73,4	62,1	68,6	75,6	60,4	64,5	72,1
In to Tree	63,4	51,8	53,6	71,0	58,8	60,2	75,4	64,3	65,5	78,3	67,7	69,0	72,0	60,7	62,1
Man in Car	80,1	83,4	85,1	81,8	84,3	84,9	82,2	84,8	85,1	82,2	84,9	85,3	81,6	84,4	85,1
Pedestrian Area	72,8	80,6	81,6	74,0	81,8	82,2	74,8	82,6	82,7	75,4	83,2	83,1	74,2	82,1	82,4
Riverbed	64,7	77,3	81,4	66,8	80,7	84,2	69,2	82,7	85,2	71,4	83,4	85,6	68,0	81,0	84,1
Rolling Tomatoes	76,8	81,2	82,8	79,2	82,5	82,5	79,3	82,9	82,6	79,0	83,1	82,7	78,6	82,4	82,7
Station 2	63,0	77,7	77,7	66,2	80,2	80,1	68,4	81,4	81,6	70,4	82,4	82,9	67,0	80,4	80,6
Sun Flower	70,9	69,7	73,1	71,0	70,7	74,4	71,0	71,4	75,6	71,2	72,1	76,5	71,0	71,0	74,9
Tennis	72,4	80,8	80,7	73,1	82,0	81,9	74,3	82,8	82,8	76,0	83,3	83,3	73,9	82,2	82,2
Tractor	67,9	71,8	73,0	68,5	73,4	73,9	69,4	74,9	75,0	70,6	76,0	75,8	69,1	74,0	74,4
People On Street	55,0	70,4	74,2	57,3	72,1	76,3	58,7	74,0	78,9	60,9	76,4	80,2	58,0	73,2	77,4
Traffic	58,4	66,2	72,1	60,4	68,5	74,3	62,3	70,2	76,5	64,3	73,1	78,7	61,4	69,5	75,4
Média Todos	**63,8**	**73,2**	**75,9**	**66,9**	**75,6**	**77,8**	**68,7**	**77,2**	**79,2**	**70,4**	**78,8**	**80,6**	**67,5**	**76,2**	**78,4**
Média 4:2:0		**67,4**			**70,2**			**71,9**			**73,5**			**70,7**	
Des. Pad. Todos	**9,8**	**8,0**	**7,1**	**7,9**	**6,9**	**5,8**	**7,1**	**6,1**	**4,8**	**6,4**	**5,3**	**4,2**	**7,6**	**6,5**	**5,4**

* vídeo da CCT

Como pode ser visto nessa seção, a taxa de compressão dos algoritmos apresenta comportamentos semelhantes, i.e. a sequência que apresenta a menor taxa de compressão é a mesma para os três algoritmos, *BQ Terrace*. Do mesmo modo, a sequência que apresenta a maior taxa de compressão também é a mesma,

Man in Car. Além disso, esse comportamento similar também pode ser visto nos resultados de taxa de compressão para crominância, onde todos algoritmos apresentaram melhores resultados quando comparados aos resultados de luminância. Esse comportamento se deve ao fato de que essas soluções tem o mesmo fluxo de codificação. Porém, o DDRFVLC é o algoritmo que apresenta as maiores taxas de compressão, pois é a solução que utiliza códigos VLC com dupla codificação diferencial. Entretanto, o DDRFVLC apresenta custo computacional maior que as demais, e a implementação em hardware dessa solução implica em custos mais elevados em área e potência.

7.2 Resultados de síntese

As arquiteturas dos três algoritmos foram descritas em VHDL e sintetizadas para ASIC em tecnologia *standard cells*. A síntese foi gerada visando as tecnologias 180nm e 65nm da TSMC, com 1,8V e 1V de tensão de alimentação, respectivamente, utilizando a ferramenta *Cadence Encounter RTL Compiler* (CADENCE DESIGN SYSTEMS, 2014).

Os resultados das sínteses *standard cells* são apresentados na Tabela 7. Essa tabela apresenta os resultados de área, em *gates*, e de potência total (*Leakage* + dinâmica) para as arquiteturas do codificador, decodificador e codec (codificador e decodificador) para duas frequências operacionais: 62,5MHz (HD 1080p a 30 qps) e 250MHz (HD 1080p a 120 QPS ou 4K a 30 QPS). Com essas frequências as arquiteturas atingem codificação em tempo real para sequências de vídeo HD 1080p e 4K. Os resultados são apresentados separadamente para o codificador, o decodificador e para o codec completo, que integra as arquiteturas.

Conforme se pode observar nos resultados apresentados na Tabela 7, a arquitetura do DRFC apresenta a menor dissipação de potência e a menor área, quando comparado às outras arquiteturas. A arquitetura do DRFVLC apresenta resultados bem próximos aos resultados do DRFC, mesmo utilizando códigos VLC. Isto se deve ao fato de que a tabela com os códigos VLC possui tamanho reduzido, sendo que apenas o intervalo de resíduos de [-16, 16] é representado. Para os demais resíduos é utilizado o código de exceção. A arquitetura do DDRFVLC é a que apresenta a maior ocupação de área e a maior dissipação de potência, sendo até 3,2 vezes maior que o DRFC tanto em área quanto em potência. Esse resultado é devido à grande quantidade de registradores utilizado pelo DDRFVLC, essa

diferença está principalmente no banco de registradores, uma vez que o DDRFVLC deve armazenar uma coluna inteira do bloco original (512 bits) para realizar a dupla codificação diferencial, enquanto que nos outros algoritmos apenas dois registradores de 8 bits são suficientes.

Tabela 7 – Resultados de síntese das arquiteturas de hardware desenvolvidas

		Tecnologia	**180nm**		**65nm**	
		Frequência	**62,5MHz**	**250MHz**	**62,5MHz**	**250MHz**
DRFC	Codificador	Potência - mW	0,766	1,83	0,089	0,253
		Área - gates	1191	1349	1920	1999
	Decodificador	Potência - mW	2,13	4,57	0,183	0,535
		Área - gates	2500	2559	2251	2235
	Codec	**Potência - mW**	**2,896**	**6,4**	**0,272**	**0,788**
		Área - gates	**3691**	**3908**	**4171**	**4234**
DRFVLC	Codificador	Potência - mW	0,956	2,17	0,123	0,381
		Área - gates	1388	1649	2034	1963
	Decodificador	Potência - mW	2,24	5,09	0,173	0,511
		Área - gates	3161	3171	2938	2940
	Codec	**Potência - mW**	**3,196**	**7,26**	**0,296**	**0,893**
		Área - gates	**4549**	**4820**	**4973**	**4903**
DDRFVLC	Codificador	Potência - mW	3.87	11	0,529	1,513
		Área - gates	5036	5299	4685	4771
	Decodificador	Potência - mW	5,44	14	0,610	1,740
		Área - gates	6784	6813	5736	5738
	Codec	**Potência - mW**	**9,31**	**25**	**1,139**	**3,253**
		Área - gates	**11820**	**12112**	**10421**	**10509**

A Figura 24 apresenta um gráfico com a relação de consumo de energia (pJ) por amostra processada das arquiteturas desenvolvidas. Os resultados apresentados são para a tecnologia de 65nm, que foi a tecnologia mais nova utilizada nas sínteses. Quanto menor esse valor mais eficiente é a solução. Os resultados são apresentados para as duas frequências utilizadas nas sínteses, 62,5MHz e 250MHz. Esse comparativo mostra que o DRFVLC, por ter uma dissipação de potência semelhante ao DRFC, apresenta também bons resultados de eficiência energética. Entretanto, o DDRFVLC tem maior potência dissipada, maior área e eficiência energética menor que as demais soluções. Isso ocorre devido à maior utilização de registradores nessa arquitetura. Porém, esses registradores são essenciais para o funcionamento correto da dupla codificação diferencial. No entanto, é importante destacar que estes resultados não devem ser avaliados isoladamente, tendo em vista os ganhos de taxa de compressão dos quadros de

referência proporcionados pelo uso destas arquiteturas dentro do codificador de vídeo.

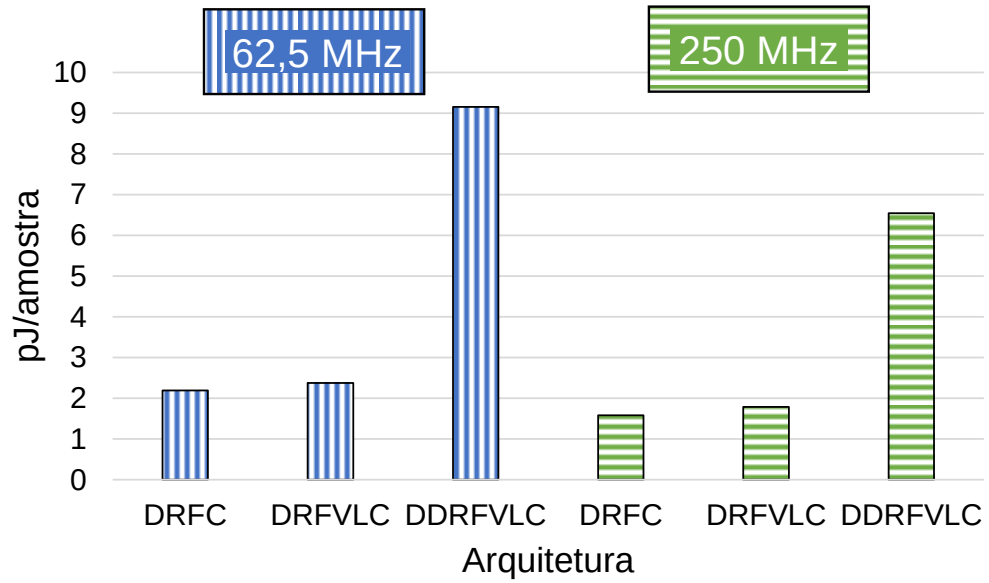


Figura 24 – Consumo de energia por amostra processada das arquiteturas desenvolvidas para síntese 65nm

Apesar da grande diferença na dissipação de potência entre as arquiteturas desenvolvidas, esses resultados causam um acréscimo mínimo em um sistema de codificação de vídeo completo. Por exemplo, um projeto do codificador H.264/AVC (LIU, SONG, *et al.*, 2009), desenvolvido em tecnologia de 180nm, que processa vídeos HD 1080p a 30 QPS tem um dissipação de potência de 1,4W. O acréscimo inserido com o DDRFVLC para a mesma tecnologia e taxa de processamento seria inferior a 1%, porém, com redução de 70% da largura de banda de memória. Desta forma, é possível observar que as arquiteturas dos codecs de quadros de referência desenvolvidas inserem um acréscimo insignificante para o sistema de codificação de vídeo, seja em termos de custo de hardware, desempenho ou dissipação de potência.

7.3 Redução de energia no acesso à memória externa

Com o objetivo de avaliar o impacto das arquiteturas na redução da dissipação de potência para o acesso à memória externa durante o processo de codificação de vídeo, esta seção apresenta uma estimativa de redução de largura de banda para memórias DDR (*Low Power Double Data Rate*) em um codificador de vídeo. Os resultados de redução de potência alcançados pelas soluções foram

estimados utilizando uma memória de 2 Gb (LPDDR2 SDRAM) (MICRON, 2013) com base em dados de aplicações para sequências de vídeo HD 1080p com taxa de amostragem de 30 quadros por segundo (QPS). Essa memória apresenta consumo energético de 279,27pJ por byte lido e 278,16pJ para escrever um byte na memória, tais informações foram extraídas da ferramenta *Power Calculator* (MICRON, 2013) desenvolvida pela *Micron Technology*. Os valores de largura de banda de memória utilizados na estimativa já incluem a fragmentação de memória inserida pela estratégia de integração e acesso aleatório desenvolvida. A largura de banda de memória média, com os quatro valores de QP para as oito sequências utilizadas nos testes, foram consideradas nessa estimativa.

A Tabela 8 apresenta os resultados de consumo de energia para leitura e escrita na memória LPDDR2, sem a compressão de quadros de referência e utilizando a compressão com os três algoritmos desenvolvidos. Os resultados apresentados na Tabela 8 foram gerados para a síntese das arquiteturas na tecnologia de 65nm. Neste cenário, as soluções mostraram uma redução no consumo energético de leitura e escrita na memória de 47,10% com o DRFC, 58,23% com o DRFVLC e 64,20% com o DDRFVLC, incluindo o *overhead* de consumo dos codecs. Isso representa uma redução real no consumo de energia relacionada ao acesso à memória de mais de 22 mJ quando o DDRFVLC é utilizado.

Tabela 8 – Estimativa de redução do consumo de energia em acessos à memória

	65nm – 62,5MHz			
	Sem Compressão	DRFC	DRFVLC	DDRFVLC
Leitura (mJ)	17,38	9,05	7,11	5,65
Escrita (mJ)	17,30	9,02	7,08	5,62
Overhead codificador (mJ)	-	0,089	0,123	0,610
Overhead decodificador (mJ)	-	0,183	0,173	0,529
Overhead memória auxiliar (mJ)	-	0,004	0,004	0,004
Dissipação total (mJ)	**34,68**	**18,34**	**14,48**	**12,41**
Redução (mJ)	-	**16,33**	**20,19**	**22,26**
Redução (%)	-	**47,10**	**58,23**	**64,20**

A Tabela 8 mostra que a maior redução é obtida com a arquitetura do DDRFVLC, sendo essa a solução mais eficiente, mesmo apresentando o maior consumo energético entre as soluções desenvolvidas. Isso se explica pelo fato de que o consumo energético necessário para ler ou escrever uma amostra na memória externa é muito maior do que a energia necessária para processar essa amostra no

codec de quadros de referência. Desta forma, a solução que apresentou maiores taxas de compressão é também a solução mais eficiente.

7.4 Comparativo com trabalhos relacionados

Nesta seção, os resultados de software e hardware do DDRFVLC, apresentados anteriormente, são comparados com os trabalhos relacionados mais relevantes encontrados na literatura. Esses trabalhos relacionados foram apresentados e discutidos no capítulo 3.

7.4.1 Resultados de software

A Tabela 9 mostra o comparativo dos resultados de software entre o DDRFVLC e todos os trabalhos relacionados. Na comparação com os trabalhos foram considerados os resultados da taxa de compressão, a degradação da qualidade do vídeo e o custo computacional, em uma avaliação qualitativa. Na taxa de compressão são considerados os resultados médios de luminância dos quatro QPs de todas as 17 sequências utilizadas na seção 7.1.

Tabela 9 – Comparação de resultados de software com os trabalhos relacionados

Trabalho	Taxa de compressão (%)	ΔPSNR (dB)	Custo computacional
DDRFVLC	**67,5**	**0,00**	**2 CD + 1 VLC**
DDRFVLC [1]	**62,4**	**0,00**	**2 CD + 1 VLC**
Yvanov, (2008)	50,0	0,47	7 PD + 1 Quantização
Budagavi, (2008)	25,0 – 37,5	0,04 – 0,15	1 Quantização
Cheng, (2009)	54,8 – 76,6	0,0 – 1,4	2 DWT + 1 SPIHT
Gupte, (2011)	23,0	0,01	1 Quantização
Ma, (2011)	31,0	0,01	1 CD + 1 Quantização
Lee, (2009)	48,0	0,00	1 HMD + 1 VLC
Kim, (2010)	53,3	0,00	4 HACP + 1 SBT
Bao, (2010)	60,2	0,00	6 DPCM + 6 VLC
Zhou, (2011)	51,0	0,00	1 DPCM + 1 VLC
Kuo, (2012)	66,4	0,00	1 TC + 1 APBT + 1 HCC + 2 VLC
Guo, (2013) [1]	57,6	0,00	4 MDA + 15 RGM + 1 SFL
Zhou, (2014) [1]	57,6	0,00	4 MDA + 9 RGM + 1 SFL
Lee, (2014) [1]	51,2	0,00	4 ACF + 1 VLC
Chen, (2014) [1]	42,9	0,00	4 MDP + 1 VLC

[1] trabalhos avaliados com os vídeos das CCT

Como pode ser observado na Tabela 9, o DDRFVLC apresenta a maior taxa de compressão entre os trabalhos sem perdas de qualidade e, além disso, o DDRFVLC é um dos trabalhos de menor custo computacional. Os trabalhos de Yvanov (2008), Budagavi (2008), Gupte (2011) e Ma (2011) utilizam técnicas que apresentam menor custo computacional. No entanto, esses trabalhos tem uma taxa

de compressão muito baixa. Além disso, esses trabalhos apresentam técnicas de compressão com perdas, que introduzem degradação da qualidade no vídeo codificado.

Diferente desses trabalhos, Cheng (2009) apresenta uma maior taxa de compressão; no entanto, pode introduzir alta degradação de qualidade de acordo com o perfil de operação utilizado. Soluções com perdas de qualidade não são desejáveis para serem utilizadas em compressão de quadros de referência. Essas perdas podem introduzir degradação da qualidade de forma expressiva no processo de codificação, uma vez que o erro pode ser propagado durante a predição inter-quadros. Além disso, é introduzida uma diferença entre os quadros de referência da codificação e da decodificação.

No entanto, os trabalhos de Lee (2009) e Chen (2014) apresentam soluções sem perdas de qualidade. Embora esses trabalhos não exijam um alto custo computacional para serem implementados, a taxa de compressão alcançada por essas soluções fica abaixo de 50%. Os trabalhos de Kim (2010), Zhou (2011), Guo (2013) e Lee (2014), apresentam uma boa taxa de compressão de dados, variando de 51% a 57%. No entanto, o DDRFVLC atinge uma taxa de compressão até 20% maior quando comparado a esses trabalhos.

Quando comparado com Bao (2010), é possível perceber que o DDRFVLC apresenta um custo computacional menor, com uma taxa de compressão 15% superior. Esse trabalho usa seis DPCMs independentes, seguidos por seis VLCs, e uma decisão final para selecionar o resultado com a menor taxa de bits. Além disso, os resultados da taxa de compressão foram gerados utilizando apenas 10 quadros de cinco vídeos HD 1080p, o que não representa todo o cenário de testes, podendo distorcer as reais taxas de compressão da solução.

Kuo (2012) apresenta resultados de taxa de compressão semelhantes ao DDRFVLC. Porém, essa solução apresenta alto custo computacional devido a utilização de várias técnicas de compressão utilizadas em conjunto, tais como: codificação de dicionário, truncamento de bits e códigos VLC, com grande dependência de dados entre as técnicas. Além disso, essa é uma solução de codificação em linha, o que não é desejável em compressão de quadros de referência, uma vez que a acessibilidade aos blocos codificados na memória é dificultada.

Para garantir uma comparação justa com os quatro últimos trabalhos da Tabela 9, que só consideram vídeos recomendados pelas CCTs do HEVC, a Tabela 9 também apresenta a taxa de compressão do DDRFVLC considerando apenas os vídeos das CCTs (DDRFVLC[1]). Neste cenário, a solução proposta atinge uma redução de 62,4% (apenas luminância), enquanto os trabalhos relacionados atingem no máximo 57,6% de redução. Resumindo, o DDRFVLC tem a maior taxa de compressão de dados entre todos os trabalhos relacionados sem perdas. Além disso, apresenta um baixo custo computacional (duas DC e um VLC).

7.4.2 Resultados de hardware

A Tabela 10 apresenta uma comparação entre o DDRFVLC e os trabalhos relacionados que apresentam resultados de hardware. Os resultados de redução de potência apresentados na Tabela 10 seguem o mesmo critério utilizado para estimar a redução de potência apresentada na Tabela 8, a qual indica a redução de potência decorrente dos acessos à memória, considerando a sobrecarga introduzida pela solução, porém, o número de sequências de vídeo utilizados foi ampliado, contemplando todas sequências apresentadas na Tabela 6. A Tabela 10 traz também resultados de eficiência energética, a qual indica o consumo de energia para codificar uma amostra (energia consumida/amostra). Além desses resultados, são também apresentados os resultados de síntese dos trabalhos relacionados. Os resultados do DDRFVLC são comparados com seis trabalhos relacionados, sendo esses os únicos entre todos os trabalhos que apresentam resultados de hardware.

Tabela 10 – Comparação de resultados de hardware com os trabalhos relacionados

Trabalho	Tecnologia (nm)	Gates (K)	Potência (mW)	Frequência (MHz)	Vazão (QPS)	Eficiência Energética (pJ/amostra)	Redução de Energia (mJ [%])
DDRFVLC	**180**	**11,8**	**9,31**	**62,5**	**1080p@30**	**74,83**	**14,09 [40,64]**
DDRFVLC	**65**	**10,4**	**1,13**	**62,5**	**1080p@30**	**9,15**	**22,26 [64,20]**
DDRFVLC	**65**	**10,5**	**3,25**	**250**	**1080p@120 / 4K@30**	**6,54**	**90,36 [65,14]**
Cheng, (2009)	180	27	3,78	10	CIF@30	621,45	0,23 [13,51]
Gupte, (2011)	65	14	1,35	250	1080p@30	21,70	5,28 [15,21]
Kim, (2010)	180	36	17,5	180	1080p@30	140,66	0,98 [2,83]
Guo, (2013)	65	100	N/A	300	4K@120	N/A	N/A
Zhou, (2014)	90	34,5	20	300	4K@60	20,09	119,79 [43,18]
Lee, (2014)	180	28	9,35	62,5	1080p@30	75,15	8,40 [24,24]

Em Cheng (2009) é apresentada uma solução sem perdas de qualidade com baixa dissipação de potência e boa redução de potência da memória. No entanto, essa solução apresenta uma baixa taxa de processamento (CIF 30 qps/10 MHz), o que reduz a sua relevância em cenários das aplicações multimídia atuais, uma vez que hoje em dia até mesmo dispositivos móveis podem processar vídeos de alta definição. O trabalho de Gupte (2011) apresenta maior dissipação de potência quando comparado ao DDRFVLC para a mesma taxa de processamento (HD 1080p a 30 qps) na mesma tecnologia de síntese (180nm). Desta forma, o DDRFVLC apresenta maior eficiência energética que esse trabalho.

O trabalho de Kim (2010) apresenta uma alta dissipação de potência e uma alta frequência para a mesma taxa de processamento, quando comparado ao DDRFVLC. Devido a isso, esse trabalho apresenta baixa taxa de redução de consumo de energia quando uma memória moderna como LPDDR2 é considerada. Desta forma, restringindo a sua utilização em aplicações modernas. O trabalho de Lee (2014) apresenta uma solução com a mesma frequência, taxa de processamento e uma dissipação de potência semelhante ao DDRFVLC. Porém, o DDRFVLC atinge uma redução de energia 16% maior, que se deve à maior taxa de compressão atingida pelo DDRFVLC.

Em Guo (2013) é apresentada uma arquitetura com frequência e taxa de processamento mais elevada que o DDRFVLC. No entanto, a arquitetura desenvolvida apresenta limitações nos modos de predição do algoritmo original apresentado no trabalho. Estas limitações foram aplicadas para reduzir o custo de implementação da arquitetura. Assim, os resultados da taxa de compressão dessas arquiteturas são desconhecidas, i.e., diferentes dos resultados apresentados na Tabela 9. Além disso, os resultados de dissipação de potência não são apresentados, mas, devido à sua alta frequência (4,8 vezes maior do que DDRFVLC) e alta ocupação de área (10 vezes mais elevado que o DDRFVLC), é esperado que essa solução apresente alta dissipação de potência. O trabalho de Zhou (2014) é uma implementação mais eficiente do trabalho de Guo (2013). No entanto, o DDRFVLC tem uma eficiência energética 3 vezes maior do que esse trabalho para o processamento de vídeos 4K. Além disso, o DDRFVLC proporciona uma redução em consumo de energia 20% maior do que Zhou (2014).

O codec DDRFVLC apresenta os melhores resultados em uso de hardware, dissipação de potência e energia, em comparação com todos os trabalhos

relacionados encontrados na literatura. Considerando a síntese de 65nm, apenas 10,4K *gates* são usados, com uma dissipação de potência de 1,13 mW. Além disso, o DDRFVLC atinge taxa de processamento para vídeos UHD 4K (3840x2160 amostras) a 30 qps, ou HD 1080p (1920x1080 amostras) a 120 qps, com uma dissipação de potência de 3,25 mW. Em termos de energia, o DDRFVLC consome 6,54pJ por amostra para essa taxa de processamento. É importante destacar que esse desempenho pode ser multiplicado através da adição de outros módulos de codecs de hardware DDRFVLC, operando em paralelo. Isso é possível devido à organização de memória proposta e a codificação de blocos de forma independente, que é realizada pelo codec DDRFVLC.

8 CONCLUSÕES E TRABALHOS FUTUROS

Essa dissertação apresentou três soluções algorítmicas e arquiteturais para a compressão dos quadros de referência em codificadores de vídeo digitais. Através dessa compressão foi possível reduzir de forma considerável os acessos à memória e a largura de banda de memória necessária durante o processo da Estimação de Movimento (ME). A comunicação entre a ME e a memória externa é o que demanda a maior dissipação de potência nos codificadores de vídeo atuais, representando um gargalo crítico nesse sistema. As três soluções apresentadas, DRFC, DRFVLC e DDRFVLC, atingem alta taxa de processamento, alta economia de energia, não inserem perdas de qualidade no vídeo e apresentam baixo custo para implementação em hardware. As soluções utilizam duas técnicas de codificação em conjunto, a codificação diferencial entre amostras vizinhas para transformar os valores das amostras em resíduos próximos a zero, explorando a alta similaridade entre as amostras, e uma codificação de entropia eficiente, que possui uma tabela com códigos estáticos previamente elaborados.

O algoritmo DRFC utiliza codificação diferencial simples combinada com uma codificação de entropia onde os códigos estáticos tem o mesmo tamanho, 3 bits, que ficam armazenados em uma pequena tabela com 8 posições. Essa tabela cobre o intervalo de resíduos [-3, 3] e possui um código para as exceções que ficam fora desse intervalo. O algoritmo DRFVLC também utiliza uma codificação diferencial, porém, a codificação de entropia utiliza códigos de tamanho variável, gerados a partir do algoritmo de *Huffman*, o qual gera códigos mais otimizados para valores com maior ocorrência. A tabela de códigos utilizada pelo DRFVLC possui 34 posições e cobre o intervalo de representação dos resíduos [-16, 16]. O algoritmo DDRFVLC também utiliza uma tabela de códigos estáticos para esse intervalo, porém, inova na utilização de duas codificações diferencias em sequência, para sentidos distintos, a qual possibilita a eliminação de bordas horizontais e verticais. Os três algoritmos apresentaram o melhor compromisso entre custo computacional e

taxa de compressão quando utilizaram blocos com 64x64 amostras, sendo que esse é o tamanho de CTU do HEVC que gera os melhores resultados de codificação.

As soluções desenvolvidas atingem altas taxas de compressão dos quadros de referência, e consequentemente, de largura de banda para o acesso à memória. O DRFC reduz em média 47,9% das informações, o DRFVLC atinge uma taxa de compressão média de 59,1%, e no DDRFVLC a taxa de compressão média é de 67,5%. Essa redução em largura de banda é obtida tanto para leitura quanto para escrita na memória externa, sendo que estas soluções podem ser aplicadas a outros codificadores de vídeo como H.264/AVC e MPEG-2.

Este trabalho também apresentou o projeto arquitetural completo para os três algoritmos desenvolvidos, incluindo os módulos codificador e decodificador. As arquiteturas foram descritas em VHDL e sintetizadas para ASIC com *standard cells*. A síntese foi gerada para duas tecnologias, 180nm e 65nm, e para duas frequências operacionais, 62,5MHz (HD 1080p a 30 qps) e 250MHz (HD 1080p a 120 qps ou UHD 4K a 30 qps). Os resultados das sínteses das arquiteturas mostraram que o DDRFVLC é a solução mais eficiente, dissipando uma potência de 1,13mW para codificar vídeos HD 1080p, e 3,25mW para vídeos UHD 4K, o que é um *overhead* insignificante, uma vez que essa solução atinge uma redução de energia de 90,36mJ (65,14%) a partir da redução dos acessos à memória externa.

Considerando todos os resultados apresentados, é possível inferir que a solução apresentada nessa dissertação apresenta impactos positivos sobre o consumo de energia de um sistema completo de codificador de vídeo, já que o acesso à memória externa é um dos principais gargalos nos sistemas digitais atuais.

Quando comparado com os trabalhos relacionados estado da arte, essa é a solução de compressão de quadros de referência, sem perdas de qualidade, que atinge a maior taxa de compressão. Além disso, os resultados de síntese das arquiteturas de hardware mostram que a solução desenvolvida apresenta as maiores taxas de redução de potência relativas ao acesso à memória externa em comparação com todos os trabalhos relacionados encontrados na literatura.

Como trabalhos futuros, planeja-se: (1) integrar o codec de quadros de referência com uma estratégia de reuso de dados, alcançando uma taxa de redução de largura de banda superior a obtida apenas com compressão; (2) integrar esse sistema em uma arquitetura de um codificador HEVC, gerando um sistema completo, e (3) explorar esse sistema em um codificador de vídeo 3D, onde as

questões relacionadas ao acesso à memória externa são consideravelmente ampliadas, dado o aumento do volume de dados devido a utilização de múltiplos vistas e mapas de profundidade.

REFERÊNCIAS

BAO, X. et al. An Advanced Hierarchical Motion Estimation Scheme With Lossless Frame Recompression and Early-Level Termination for Beyond High-Definition Video Coding. **IEEE Transactions on Multimedia**, v. 14, n. 2, p. 237-249, Abril 2012.

BAO, X.; ZHOU, D.; GOTO, S. **A lossless frame recompression scheme for reducing DRAM power in video encoding**. International Symposium on Circuits and Systems (ISCAS). Paris, França: IEEE. 2010. p. 677-680.

BHASKARAN, V.; KONSTANTINIDES, K. **Image and Video Compression Standards - Algorithms and Architectures**. 2ª Edição. ed. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

BOSSEN, F. **Common Test Conditions and Software Reference Configurations**. Joint Collaborative Team on Video Coding. Genebra, p. 1-3. 2012. (JCTVC-I1100).

BOSSEN, F. et al. HEVC Complexity and Implementation Analysis. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1685-1696, 2012.

BUDAGAVI, M.; ZHOU, M. **Video coding using compressed reference frames**. IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2008. Las Vegas, NV, USA: [s.n.]. 2008. p. 1165 -1168.

CADENCE DESIGN SYSTEMS. Encounter RTL Compiler, June 2014. Disponivel em: <http://www.cadence.com>.

CHEN, C.-Y. et al. Level C+ data reuse scheme for motion estimation with corresponding coding orders. **IEEE Transactions on Circuits and Systems for Video Technology**, Abril 2006. 553-558.

CHEN, Y.-G.; LEE, Y.-H.; CHEN, C.-C. **An efficient multi-directional lossless recompression for video coding systems**. International Symposium on Bioelectronics and Bioinformatics. Chung Li: IEEE. 2014. p. 1-4.

CHENG, C.-C.; TSENG, P.-C.; CHEN, L.-G. Multimode Embedded Compression Codec Engine for Power-Aware Video Coding System. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 19, n. 2, p. 141-150, 2009.

CISCO. **Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018**. [S.l.]. 2014.

CORREA, G. et al. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1899-1909, Dezembro 2012.

CORREA, G. et al. Complexity scalability for real-time HEVC encoders. **Journal of Real-Time Image Processing**, p. 1-16, Janeiro 2014.

GRELLERT, M. et al. **A multilevel data reuse scheme for Motion Estimation and its VLSI design**. IEEE International Symposium on Circuits and Systems (ISCAS). [S.l.]: [s.n.]. 2011. p. 583 -586.

GUO, L.; ZHOU, D.; GOTO, S. **Lossless embedded compression using multi-mode DPCM & averaging prediction for HEVC-like video codec**. European Signal Processing Conference. Marrakech: IEEE. 2013. p. 1-5.

GUO, L.; ZHOU, D.; GOTO, S. A New Reference Frame Recompression Algorithm and Its VLSI Architecture for UHDTV Video Codec. **IEEE Transactions on Multimedia**, v. 16, n. 8, p. 2323-2332, Dezembro 2014.

GUPTE, A. et al. Memory Bandwidth and Power Reduction Using Lossy Reference Frame Compression in Video Encoding. **IEEE Transactions on Circuits and Systems for Video Technology**, Fevereiro 2011. 225-230.

HUFFMAN, D. A. **A Method for the Construction of Minimum-Redundancy Codes**. Proceedings of the IRE. [S.l.]: [s.n.]. 1952. p. 1098-1101.

IPSL. **Complexity and Performance Analysis of HEVC encoder**. Image Processing Systems Laboratory - Kwangwoon University. [S.l.], p. 1-57. 2012.

ISO/IEC. **International Organization for Standardization. ISO/IEC 11172-2 MPEG-1 (1993): Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video**. [S.l.]. 1993.

ISO/IEC. **International Organization for Standardization. ISO/IEC 14496-2 MPEG-4 Part 2 (2004): Information technology - Coding of audio-visual objects - Part 2: Visual**. [S.l.]. 2004.

ISO/IEC. Moving Picture Experts Group (MPEG), 2012. Disponivel em: <http://mpeg.chiariglione.org/>. Acesso em: Junho 2012.

ITU-T. **International Telecommunication Union. Recommendation H.262 (02/00): Information technology - Generic coding of moving pictures and associated audio information: Video**. [S.l.]. 2000.

ITU-T. **International Telecommunication Union. Recommendation H.263 (01/05): Video coding for low bit rate communication**. [S.l.]. 2005.

ITU-T. **International Telecommunication Union. Recommendation H.264 (01/12): Advanced video coding for generic audiovisual services**. [S.l.]. 2012.

ITU-T. Video Coding Experts Group (VCEG), 2012. Disponivel em: <http://www.itu.int/ITU-T/studygroups/com16/>. Acesso em: Junho 2012.

ITU-T. **Recommendation H.265: High Efficiency Video Coding, Audiovisual and Multimedia Systems**. [S.l.]. 2013.

IVANOV, Y. V.; MOLONEY, D. **Reference Frame Compression Using Embedded Reconstruction Patterns for H.264/AVC Decoder**. International Conference on Digital Telecommunications. [S.l.]: [s.n.]. 2008. p. 168-173.

KIM, J.; KYUNG, C.-M. A Lossless Embedded Compression Using Significant Bit Truncation for HD Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 20, n. 6, p. 848-860, Junho 2010.

KUHN, P. **Algorithms, Complexity Analysis and Vlsi Architectures for Mpeg-4 Motion Estimation**. São Paulo, Brasil: Kluwer Academic Publisher, 1999.

KUO, H.-C.; LIN, Y.-L. A Hybrid Algorithm for Effective Lossless Compression of Video Display Frames. **IEEE Transactions on Multimedia**, v. 14, n. 3, p. 500-509, Junho 2012.

LEE, S.-H. et al. Lossless Frame Memory Recompression for Video Codec Preserving Random Accessibility of Coding Unit. **IEEE Transactions on Consumer Electronics**, v. 55, n. 4, p. 2105-2113, Novembro 2009.

LEE, Y.-H.; CHEN, C.-C.; YOU, Y.-L. Design of VLSI architecture of autocorrelation-based lossless recompression engine for memory-efficient video coding systems. **Circuits, Systems, and Signal Processing**, v. 33, n. 2, p. 459-482, Fevereiro 2014.

LI, D.-X.; ZHENG, W.; ZHANG, M. Architecture Design for H.264/AVC Integer Motion Estimation with Minimum Memory Bandwidth. **IEEE Transactions on Consumer Electronics**, v. 53, n. 3, p. 1053-1060, Agosto 2007.

LIU, Z. et al. HDTV1080p H.264/AVC Encoder Chip Design and Performance Analysis. **IEEE Journal of Solid-State Circuits**, v. 44, n. 2, p. 594-608, Fevereiro 2009.

MA, Z.; SEGALL, A. **Frame Buffer Compression for Low-power Video Coding**. IEEE International Conference on Image Processing. [S.l.]: [s.n.]. 2011. p. 757-760.

MCCANN, K. et al. **High Efficiency Video Coding Software Repository**, 2013. Disponivel em: <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>. Acesso em: Março 2014.

MICRON. Micron MT47H32M16HR: 512 Mb DDR2 SDRAM, 2013. Disponivel em: <http://www.micron.com>. Acesso em: 2013.

MICRON. Micron Technical Note: Calculating Memory System Power for DDR2, 2013. Disponivel em: <http://www.micron.com>. Acesso em: 2013.

MULLER, K. et al. 3D High-Efficiency Video Coding for Multi-View Video and Depth Data. **IEEE Transactions on Image Processing**, v. 22, n. 9, p. 3366-3378, Setembro 2013.

NHK. Japan Broadcasting Corporation: Advanced Television Systems Research Division, 2012. Disponivel em: <http://www.nhk.or.jp/strl/english/index.html>. Acesso em: Junho 2012.

OHM, J.-R. et al. Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1669-1684, Dezembro 2012.

RICHARDSON, I. E. G. **Video Codec Design Developing Image and Video Compression Systems**. [S.l.]: Wiley, 2002.

RICHARDSON, I. E. G. **The H.264 Advanced Video Compression Standard**. 2ª Edição. ed. [S.l.]: John Wiley and Sons, 2010.

SALOMON, D. **Data Compression - The Complete Reference**. 4ª Edição. ed. London, United Kingdom: Springer, 2007.

SAMPAIO, F. et al. **dSVM:** Energy-efficient distributed Scratchpad Video Memory Architecture for the next-generation High Efficiency Video Coding. Design, Automation and Test in Europe Conference and Exhibition. [S.l.]: [s.n.]. 2014. p. 1-6.

SAYOOD, K. **Introduction to Data Compression**. 3ª Edição. ed. USA: Morgan Kaufmann, 2005.

SHAFIQUE, M. et al. **Adaptive Power Management of On-Chip Video Mamory for Multiview Video Coding**. IEEE/ACM/EDA Design Automation Conference. [S.l.]: [s.n.]. 2012. p. 866-875.

SHANNON, C. A Mathematical Theory of Communication. **The Bell System Technical Journal**, v. 27, n. 3, p. 379-423, Julho 1948.

SULLIVAN, G. J. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649-1668, Dezembro 2012.

SULLIVAN, G.; TOPIWALA, P.; LUTHRA, A. **The H.264/AVC Advanced Video Coding Standard - Overview and Introduction to the Fidelity Range Extensions**. SPIE conference on Applications of Digital Image Processing XXVII. [S.l.]: [s.n.]. 2004. p. 454-474.

TIKEKAR, M. et al. A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications. **IEEE Journal of Solid-State Circuits**, v. 49, n. 1, p. 61-72, Jan 2014.

TUAN, J.-C.; CHANG, T.-S.; JEN, C.-W. On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 12, n. 1, p. 61-72, Janeiro 2002.

ZATT, B. et al. **Run-Time Adaptive Energy-Aware Motion and Disparity Estimation in Multiview Video Coding**. Design Automation Conference. San Diego: [s.n.]. 2011. p. 1026-1031.

ZHOU, D. et al. A 530 Mpixels/s 4096x2160@60fps H.264/AVC High Profile Video Decoder Chip. **IEEE Journal of Solid-State Circuits**, v. 46, n. 4, p. 777-788, Abril 2011.

ZHOU, D. et al. **Reducing power consumption of HEVC codec with lossless reference frame recompression**. International Conference on Image Processing. Paris: IEEE. 2014. p. 2120-2124.

APENDICE A – RESULTADOS DAS TAXAS DE COMPRESSÃO DOS ALGORITMOS

As Tabela 11 e 12 apresentam os resultados de taxa de compressão (%) para os experimentos realizados com o algoritmo DRFC. Os resultados apresentados nessas tabelas são a média dos quatro QPs utilizados (22, 27, 32, 37). A Tabela 11 apresenta os resultados para os blocos 4x4, 8x8 e 16x16, enquanto que a Tabela 12 apresenta os resultados dos demais tamanhos de blocos, 32x32, 64x64 e 128x128.

Tabela 11 – Taxa de compressão média do DRFC para blocos 4x4, 8x8 e 16x16

Tamanho do bloco	**4x4**				**8x8**				**16x16**			
Intervalo	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**
Pedestrian Area	47,04	48,45	25,1	34,08	49,4	50,89	34,76	35,79	49,99	51,51	40,17	36,23
Rolling Tomatoes	63,42	57,18	27,71	34,88	66,65	59,98	37,76	36,57	67,49	60,7	43,41	37,01
Station2	38,09	44,63	24,44	33,98	40,39	47,24	34,01	35,67	41,12	48,04	39,38	36,09
Tennis	48,64	48,83	25,54	34,31	51,24	51,31	35,26	36,02	51,94	51,94	40,71	36,44
Cactus	42,67	42,38	25,98	31,41	44,69	44,41	33,07	32,93	45,19	44,91	36,93	33,3
Riverbed	30,92	44,47	26,85	34,48	34,84	48,75	36,82	36,43	36,56	50,52	42,39	37,00
Basketball Drive	44,76	45,63	23,38	32,35	47,51	48,08	32,73	33,99	48,38	48,76	37,99	34,40
Man in Car	65,84	57,11	27,65	34,9	69,16	59,95	37,73	36,64	69,98	60,64	43,36	37,07
Média	**47,67**	**48,59**	**25,83**	**33,80**	**50,49**	**51,33**	**35,27**	**35,51**	**51,33**	**52,13**	**40,54**	**35,94**

Tabela 12 – Taxa de compressão média do DRFC para blocos 32x32, 64x64 e 128x128

Tamanho do Bloco	**32x32**				**64x64**				**128x128**			
Intervalo	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**	**[-1,1]**	**[-3,3]**	**[-7,7]**	**[-15,15]**
Pedestrian Area	50,12	51,65	43,01	36,33	50,14	51,69	44,47	36,36	50,12	51,69	45,22	36,36
Rolling Tomatoes	67,71	60,88	46,37	37,12	67,76	60,92	47,89	37,15	67,78	60,93	48,66	37,15
Station2	41,38	48,31	42,21	36,2	41,48	48,41	43,66	36,22	41,52	48,45	44,39	36,22
Tennis	52,13	52,09	43,57	36,55	52,18	52,13	45,04	36,57	52,19	52,14	45,78	36,57
Cactus	45,31	45,03	38,93	33,38	45,34	45,06	39,95	33,4	45,33	45,06	40,46	33,4
Riverbed	37,37	51,3	45,31	37,17	37,62	51,56	46,73	37,22	37,75	51,68	47,45	37,24
Basketball Drive	48,68	48,95	40,75	34,5	48,8	49,02	42,17	34,53	48,85	49,04	42,89	34,54
Man in Car	70,19	60,82	46,31	37,19	70,23	60,85	47,82	37,21	70,24	60,86	48,59	37,22
Média	**51,61**	**52,38**	**43,31**	**36,06**	**51,69**	**52,46**	**44,72**	**36,08**	**51,72**	**52,48**	**45,43**	**36,09**

As Tabela 13, 14 e 15 apresentam os resultados de taxa de compressão para os experimentos realizados com o DRFVLC. Esses resultados são apresentados por bloco, onde para cada bloco foi explorado 7 tamanhos diferentes para a tabela de códigos, que variaram do intervalo [-4,4] (TC 4) até a tabela completa com todos os resíduos (TC full). A Tabela 13 apresenta os resultados obtidos para os blocos 4x4 e 8x8, a Tabela 14 apresenta os resultados para os blocos 16x16 e 32x32, e na Tabela 15 são apresentados os resultados para os blocos 64x64 e 128x128.

Tabela 13 – Taxa de compressão média do DRFVLC para blocos 4x4 e 8x8

Tamanho do Bloco	**4x4**							**8x8**						
TC	**4**	**8**	**16**	**32**	**64**	**128**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**full**
Pedestrian Area	57,1	59,1	60,0	60,3	60,3	60,3	60,3	60,0	62,1	63,1	63,4	63,4	63,4	63,4
Rolling Tomatoes	70,7	71,0	71,1	71,1	71,1	71,1	71,1	74,2	74,5	74,6	74,6	74,6	74,6	74,6
Station2	50,1	53,0	54,3	54,6	54,6	54,6	54,6	53,0	56,0	57,3	57,7	57,6	57,6	57,6
Tennis	58,5	60,6	61,5	61,7	61,7	61,7	61,7	61,6	63,8	64,7	64,9	64,9	64,9	64,9
Cactus	52,7	54,9	55,8	56,1	56,3	56,2	56,2	55,2	57,5	58,5	58,8	59,0	58,9	58,9
Riverbed	45,6	49,1	50,4	50,6	50,5	50,5	50,5	50,2	53,5	54,6	54,8	54,8	54,8	54,8
Basketball Drive	53,6	55,6	56,3	56,5	56,7	56,7	56,7	56,7	58,7	59,4	59,7	59,9	59,9	59,9
Man in Car	74,7	75,0	75,1	75,1	75,1	75,1	75,1	78,5	78,8	78,9	78,9	78,9	78,9	78,9
Média	57,9	59,8	60,6	60,7	60,8	60,8	60,8	61,2	63,1	63,9	64,1	64,1	64,1	64,1

Tabela 14 – Taxa de compressão média do DRFVLC para blocos 16x16 e 32x32

Tamanho do Bloco	**16x16**							**32x32**						
TC	**4**	**8**	**16**	**32**	**64**	**128**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**full**
Pedestrian Area	60,7	62,8	63,8	64,1	64,1	64,1	64,1	60,9	63,0	64,0	64,3	64,3	64,3	64,3
Rolling Tomatoes	75,1	75,4	75,4	75,4	75,5	75,5	75,5	75,3	75,6	75,7	75,7	75,7	75,7	75,7
Station2	53,9	56,8	58,2	58,5	58,5	58,5	58,5	54,2	57,1	58,4	58,8	58,8	58,8	58,8
Tennis	62,5	64,6	65,5	65,7	65,7	65,7	65,7	62,7	64,9	65,8	66,0	66,0	66,0	66,0
Cactus	55,8	58,2	59,1	59,5	59,6	59,6	59,5	56,0	58,3	59,3	59,6	59,8	59,7	59,7
Riverbed	52,1	55,2	56,3	56,4	56,3	56,3	56,3	52,9	56,0	57,0	57,0	57,0	57,0	57,0
Basketball Drive	57,6	59,6	60,3	60,6	60,7	60,8	60,8	57,9	59,9	60,6	60,9	61,0	61,1	61,1
Man in Car	79,4	79,7	79,8	79,8	79,8	79,8	79,8	79,6	79,9	80,0	80,0	80,1	80,1	80,0
Média	**62,1**	**64,0**	**64,8**	**65,0**	**65,0**	**65,0**	**65,0**	**62,4**	**64,3**	**65,1**	**65,3**	**65,3**	**65,3**	**65,3**

Tabela 15 – Taxa de compressão média do DRFVLC para blocos 64x64 e 128x128

Tamanho do Bloco	**64x64**							**128x128**						
TC	**4**	**8**	**16**	**32**	**64**	**128**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**full**
Pedestrian Area	60,9	63,0	64,0	64,3	64,3	64,3	64,3	60,9	63,0	64,0	64,3	64,3	64,3	64,3
Rolling Tomatoes	75,3	75,6	75,7	75,7	75,7	75,7	75,7	75,3	75,6	75,7	75,7	75,7	75,7	75,7
Station2	54,3	57,2	58,5	58,9	58,9	58,9	58,9	54,3	57,3	58,6	58,9	58,9	58,9	58,9
Tennis	62,7	64,9	65,8	66,0	66,0	66,0	66,0	62,7	64,9	65,8	66,0	66,0	66,0	66,0
Cactus	56,0	58,3	59,3	59,6	59,8	59,8	59,7	56,0	58,3	59,3	59,6	59,8	59,8	59,7
Riverbed	53,2	56,2	57,2	57,2	57,2	57,2	57,2	53,3	56,3	57,3	57,3	57,3	57,3	57,3
Basketball Drive	58,0	60,0	60,7	60,9	61,1	61,2	61,2	58,0	60,0	60,7	61,0	61,1	61,2	61,2
Man in Car	79,7	80,0	80,1	80,1	80,1	80,1	80,1	79,7	80,0	80,1	80,1	80,1	80,1	80,1
Média	**62,5**	**64,4**	**65,2**	**65,3**	**65,4**	**65,4**	**65,4**	**62,5**	**64,4**	**65,2**	**65,4**	**65,4**	**65,4**	**65,4**

Os resultados das taxas de compressão dos experimentos realizados com o DDRFVLC estão apresentados nas Tabelas 16, 17 e 18. Para esse algoritmo foram testados 8 tamanhos de tabelas para cada tamanho de bloco, totalizando 48 experimentos.

Tabela 16 – Taxa de compressão média do DDRFVLC para blocos 4x4 e 8x8

Tamanho do Bloco	**4x4**								**8x8**							
Tabela	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**
Man in car	60,7	60,8	60,7	59,7	60,1	60,0	59,9	59,9	75,8	75,6	75,4	75,6	75,7	75,7	75,7	75,7
Rolling Tomatoes	58,1	58,1	58,1	58,0	57,2	58,6	58,7	58,7	72,8	72,6	72,4	72,3	72,3	72,6	72,6	72,6
Pedestrian area	52,3	52,5	52,4	52,2	52,0	52,8	53,2	53,2	66,7	67,1	67,2	67,0	67,1	67,3	67,3	67,3
Tennis	52,2	52,5	52,4	52,1	51,8	51,7	52,4	52,4	66,4	67,0	67,0	66,9	66,9	66,8	67,0	67,0
Basketball Drive	46,9	47,5	47,4	47,1	46,3	46,5	47,2	47,2	60,6	61,5	61,6	61,6	61,4	61,4	61,6	61,6
Riverbed	45,1	45,5	45,3	45,0	44,6	46,1	46,4	46,4	58,5	59,8	60,1	59,7	59,6	59,9	59,9	59,9
Station2	44,8	45,4	45,2	44,9	45,3	45,6	46,1	46,1	57,9	59,4	59,7	59,4	59,4	59,5	59,6	59,6
Cactus	40,7	41,7	41,6	41,1	41,1	41,1	41,7	41,7	52,7	54,4	54,8	55,1	54,9	54,9	55,1	55,1
Média	**50,1**	**50,5**	**50,4**	**50,0**	**49,8**	**50,3**	**50,7**	**50,7**	**63,9**	**64,7**	**64,8**	**64,7**	**64,7**	**64,7**	**64,8**	**64,8**

Tabela 17 – Taxa de compressão média do DDRFVLC para blocos 16x16 e 32x32

Tamanho do Bloco	**16x16**								**32x32**							
Tabela	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**
Man in car	79,9	79,9	79,8	79,9	79,9	80,0	80,0	80,0	81,1	81,2	81,2	81,2	81,2	81,2	81,2	81,2
Rolling Tomatoes	77,0	76,9	76,8	76,8	76,8	76,8	76,8	76,8	78,2	78,2	78,2	78,1	78,1	78,1	78,1	78,1
Pedestrian area	71,6	72,0	72,0	72,0	72,0	72,0	72,1	72,1	73,2	73,6	73,6	73,7	73,7	73,7	73,7	73,7
Tennis	71,1	71,8	71,8	71,9	71,8	71,8	71,8	71,8	72,5	73,3	73,4	73,4	73,4	73,4	73,4	73,4
Basketball Drive	65,1	66,1	66,5	66,6	66,5	66,5	66,6	66,6	66,8	67,8	68,2	68,3	68,3	68,3	68,3	68,3
Riverbed	64,5	65,4	65,4	65,4	65,3	65,4	65,4	65,4	66,6	67,3	67,4	67,4	67,4	67,4	67,4	67,4
Station2	63,0	64,4	64,6	64,6	64,6	64,6	64,6	64,6	64,7	66,1	66,3	66,4	66,4	66,4	66,4	66,4
Cactus	56,2	58,3	59,3	59,5	59,6	59,5	59,6	59,6	57,8	59,9	61,0	61,3	61,3	61,3	61,3	61,3
Média	**68,5**	**69,3**	**69,5**	**69,6**	**69,6**	**69,6**	**69,6**	**69,6**	**70,1**	**70,9**	**71,2**	**71,2**	**71,2**	**71,2**	**71,2**	**71,2**

Tabela 18 – Taxa de compressão média do DDRFVLC para blocos 64x64 e 128x128

Tamanho do Bloco	**64x64**								**128x128**							
Tabela	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**full**
Man in car	81,5	81,5	81,6	81,6	81,6	81,6	81,6	81,6	81,6	81,7	81,7	81,7	81,7	81,7	81,7	81,7
Rolling Tomatoes	78,5	78,6	78,6	78,6	78,6	78,6	78,6	78,6	78,7	78,7	78,7	78,7	78,7	78,7	78,7	78,7
Pedestrian area	73,8	74,2	74,2	74,3	74,3	74,3	74,3	74,3	74,1	74,4	74,5	74,6	74,6	74,6	74,6	74,6
Tennis	73,1	73,8	73,9	74,0	74,0	74,0	74,0	74,0	73,3	74,0	74,2	74,2	74,2	74,2	74,2	74,2
Basketball Drive	67,5	68,5	68,9	69,0	69,0	69,0	69,0	69,0	67,7	68,8	69,2	69,3	69,3	69,3	69,3	69,3
Riverbed	67,3	68,0	68,0	68,1	68,1	68,1	68,1	68,1	67,7	68,3	68,4	68,4	68,4	68,4	68,4	68,4
Station2	65,5	66,7	67,0	67,1	67,1	67,1	67,1	67,1	65,8	67,0	67,4	67,4	67,4	67,4	67,4	67,4
Cactus	58,5	60,6	61,7	62,0	62,1	62,1	62,1	62,1	58,8	60,9	61,9	62,3	62,4	62,4	62,4	62,4
Média	**70,7**	**71,5**	**71,7**	**71,8**	**71,8**	**71,8**	**71,8**	**71,8**	**71,0**	**71,7**	**72,0**	**72,1**	**72,1**	**72,1**	**72,1**	**72,1**

APENDICE B – CÓDIGOS UTILIZADOS NAS TABELAS ESTÁTICAS

Este apêndice apresenta os códigos utilizados nas tabelas estáticas dos três algoritmos. A Tabela 19 apresenta os códigos utilizados pelo algoritmo DRFC para a configuração que apresentou o melhor custo benefício entre taxa de compressão e custo computacional, essa configuração utiliza blocos 64x64 e tabela com oito códigos estáticos, sendo um deles o código de exceção. O código de exceção é utiliza para indicar resíduos que estão fora do intervalo especificado, que nesse caso é [-3, 3].

Tabela 19 – Códigos de três bits utilizados pelo DRFC

Resíduo	Código	Resíduo	Código	Resíduo	Código	Resíduo	Código
0	000	-1	010	-2	100	-3	110
1	001	2	011	3	101	Exceção	111

A Tabela 20 apresenta os códigos utilizados pelo algoritmo DRFVLC, esse algoritmo utiliza códigos de *Huffman* em sua etapa de codificação de entropia. A tabela estática utilizada na melhor configuração do DRFVLC representa o intervalo [-16,16], desta forma 34 códigos são utilizados, incluindo código de exceção. Esses códigos possuem 9 tamanhos diferentes, sendo o menor com 1 bit (resíduo 0) e o maior com 11 bits (resíduo 16).

Tabela 20 – Códigos de *Huffman* utilizados no DRFVLC

Resíduo	Código	Tamanho	Resíduo	Código	Tamanho
0	0	1	-1	110	3
1	101	3	-2	11110	5
2	11111	5	-3	10000	5
3	10001	5	-4	100110	6
4	100111	6	-5	1110010	7
5	1110011	7	-6	1001001	7
6	1001011	7	-7	11101001	8
7	11101010	8	-8	11100000	8
8	11100010	8	-9	10010001	8
9	10010100	8	-10	111010001	9
10	111010111	9	-11	111000110	9
11	111000111	9	-12	111000010	9
12	111000011	9	-13	100101010	9
13	100101011	9	-14	100100000	9
14	100100001	9	-15	1110101100	10

15	1110101101	10	-16	1110100001	10
16	11101000001	11	Exceção	111011	6

A Tabela 21 apresenta os códigos utilizados pelo DDRFVLC, esse algoritmo também utiliza códigos de tamanho variável gerados pelo algoritmo de *Huffman*. A melhor configuração desse algoritmo utiliza blocos 64x64 e tabela com 34 códigos. Esses códigos possuem 9 tamanhos diferentes, sendo menor o resíduo 0 (1 bit) e maior o resíduo 16 (11 bits).

Tabela 21 – Códigos de *Huffman* utilizados no DDRFVLC

Resíduo	Código	Tamanho	Resíduo	Código	Tamanho
0	0	1	-1	111	3
1	110	3	-2	1010	4
2	1001	4	-3	101110	6
3	101101	6	-4	1011111	7
4	1011110	7	-5	1000110	7
5	1000101	7	-6	10110011	8
6	10110010	8	-7	10001111	8
7	10001110	8	-8	10000011	8
8	10000010	8	-9	101100011	9
9	101100010	9	-10	101100000	9
10	100010011	9	-11	100010001	9
11	100010000	9	-12	100000001	9
12	100000000	9	-13	1011000011	10
13	1011000010	10	-14	1000100101	10
14	1000100100	10	-15	1000000111	10
15	1000000110	10	-16	1000000101	10
16	10000001001	11	Exceção	100001	6

APENDICE C – PRÊMIOS E PUBLICAÇÕES DURANTE O MESTRADO

C.1 Prêmios

2014 – 1º Lugar na área das Engenharias no XVI Encontro de Pós-Graduação da UFPel

C.2 Publicações

C.2.1 Artigos em revista

Revista: Journal of Real-Time Image Processing (JRTIP), Springer, 2014
Título: Reference frame context-adaptive variable-length coder: a real-time hardware-friendly approach for lossless external memory bandwidth reduction in current video-coding systems.
Autores: Dieison Silveira, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: B1

C.2.2 Trabalhos completos publicados em anais de eventos

Evento: IEEE International Symposium on Circuits and Systems (ISCAS), 2015 (Aceito para publicação)
Título: A Real-Time Architecture for Reference Frame Compression for High Definition Video Coders.
Autores: Dieison Silveira, Guilherme Povala, Lívia Amaral, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: A1

Evento: IEEE International Conference on Image Processing (ICIP), 2014
Título: A New Differential and Lossless Reference Frame Variable-Length Coder: An Approach for High Definition Video Coders.
Autores: Dieison Silveira, Guilherme Povala, Lívia Amaral, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: A1

Evento: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014
Título: A low-complexity and lossless reference frame encoder algorithm for video coding.
Autores: Dieison Silveira, Guilherme Povala, Lívia Amaral, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: A1

Evento: IEEE International Symposium on Circuits and Systems (ISCAS), 2014
Título: Memory Bandwidth Reduction for H.264 and HEVC Encoders Using Lossless Reference Frame Coding.
Autores: Dieison Silveira, Guilherme Povala, Lívia Amaral, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: A1

Evento: Symposium on Integrated Circuits and Systems Design (SBCCI), 2014
Título: A Memory Energy Consumption Analysis of Motion Estimation Algorithms using Data Reuse in Video Coding Systems.
Autores: Lívia Amaral, Dieison Silveira, Guilherme Povala, Luciano Agostini, Marcelo Porto, Bruno Zatt
Qualis: B1

Evento: IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2014
Título: Memory energy consumption reduction in video coding systems.
Autores: Lívia Amaral, Dieison Silveira, Guilherme Povala, Marcelo Porto, Luciano Agostini e Bruno Zatt
Qualis: B5

Evento: IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2014
Título: An efficient reference frame compression approach for video coding systems.
Autores: Guilherme Povala, Dieison Silveira, Lívia Amaral, Bruno Zatt, Marcelo Porto e Luciano Agostini
Qualis: B5

Evento: IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2013
Título: An energy-efficient hardware design for lossless reference frame compression in video coders.
Autores: Dieison Silveira, Guilherme Povala, Lívia Amaral, Bruno Zatt, Luciano Agostini e Marcelo Porto
Qualis: B1