UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DIEISON ANTONELLO DEPRÁ

Algoritmos e Desenvolvimento de Arquitetura para a Codificação Binária Adaptativa ao Contexto para o Decodificador H.264/AVC

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Sergio Bampi
Orientador

Porto Alegre, março de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Deprá, Dieison Antonello

Algoritmos e Desenvolvimento de Arquitetura para a Codificação Binária Adaptativa ao Contexto para o Decodificador H.264/AVC / Dieison Antonello Deprá – Porto Alegre: Programa de Pós Graduação em Computação, 2009.

153 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: Sergio Bampi.

1.Compressão de Vídeo. 2.Codificação de Entropia. 3.CABAD. 4. CABAC. 5.Arquitetura de Hardware para o Decodificador de Vídeo segundo o Padrão H.264/AVC. I. Bampi, Sergio. II.Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Aldo Bolten Lucion
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador do PPGC: Prof. Álvaro Freitas Moreira
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Ao concluir mais uma importante etapa de meus estudos e, portanto, de minha vida é imprescindível parar, olhar para aquilo que se passou e agradecer a tudo e a todos que de alguma forma contribuíram para que essa vitória fosse alcançada.

Gostaria de começar meus agradecimentos pela minha família, pois a considero o principal pilar de sustentação do indivíduo, e devo a ela os valores que norteiam minha vida e fazem de mim a pessoa que sou. Agradeço aos meus pais Alcir Jose Deprá e Cleci Vitória Antonello Deprá por toda paciência, carinho e amor que dedicaram a mim ao longo da vida. Sou grato pela educação que me passaram e pelo esforço incansável para obter oportunidades de acesso ao ensino em todos os níveis. Ainda lembro muito bem das palavras do meu pai quando me disse que se só pudesse me oferecer uma coisa seria a educação, pois o conhecimento obtido a partir dela seria um patrimônio para toda a vida. Além disso, devo mencionar que o incentivo constante e o exemplo de trabalho e determinação de ambos foram e sempre serão fontes de motivação e muito orgulho. Fiquem cientes que essa distância que hoje nos separa, apesar do gosto amargo da saudade que nos impõe só me faz valorizar ainda mais todos os momentos que partilhamos. Quero agradecer também meu irmão, Draison Antonello Deprá, que além de irmão é meu grande amigo. Mesmo distante não deixamos de manter nossa relação e não faltaram os momentos de sorrisos e lágrimas onde nos apoiamos mutuamente. Agradeço por todo o carinho que me dedicaste e pelo apoio aos nossos pais. Saiba que és parte fundamental desta caminhada. AMO VOCÊS.

Gostaria também de agradecer a minha noiva, Laura T. C. da Silva, que dividiu comigo um grande período de solidão devido à distância que nos separava. Por ter tido a coragem de mudar sua vida para viver junto a mim e ter tido compreensão com os períodos em que humor e o "estresse" falavam mais alto que minha capacidade de lhe dedicar atenção. Obrigado pelo apoio, carinho e amor dedicado durante todos esses anos de relacionamento. Te amo.

Sou muito grato também ao Prof. Sergio Bampi meu orientador. Obrigado por ter me aceitado como seu aluno de mestrado, pela compreensão e apoio dispensado durante o desenvolvimento deste trabalho. Quero também estender esse agradecimento ao prof. Altamiro Susin que muitas vezes contribuiu com orientações e discussões sobre o andamento das atividades. Devo ressaltar minha admiração pela incansável luta de ambos para viabilizar os diversos projetos que, além de fomentar as pesquisas acadêmicas, possibilitam que tantos alunos sejam contemplados com as bolsas de apoio financeiro que são indispensáveis para à dedicação exclusiva aos estudos. Foi uma grande honra poder trabalhar e ser orientado por professores tão capacitados e imbuídos em suas atividades.

Agradeço também ao programa de pós-graduação em computação do Instituto de Informática (II) da UFRGS, representado por seu diretor prof. Flávio R. Wagner, pela infra-estrutura, organização e condições oferecidas, as quais propiciam aos alunos um ambiente ímpar a nível de instituições de ensino superior em nosso País. Sem dúvidas a

palavra qualidade seria a melhor definição para esse conjunto que compõe o II. Considerando os recursos computacionais disponíveis (tanto em *hardware* como em *software*), a qualidade dos professores, o nível do serviço dos setores administrativos e pela qualidade das pesquisas desenvolvidas estou certo que o II não fica devendo a nenhum instituto de qualquer Universidade mundo a fora. Foi um grande privilégio e motivo de orgulho fazer parte desta comunidade.

Aos meus amigos e colegas do grupo de pesquisa do laboratório 67/215 do II da UFRGS, os quais participaram em diferentes etapas. Aos meus "companheiros" que estiverem presentes desde minha chegada ao mestrado, Marcelo Porto, Bruno Zatt, Thaísa Leal e Roger Porto, com os quais muito conversei e debati para resolver dúvidas e realizar os trabalhos de disciplinas além de passar ótimos momentos de descontração. Aos amigos Vagner Rosa (doutorando) e Dr. Luciano Agostini, colegas de laboratório em momentos distintos aos quais muito recorri para esclarecer dúvidas e buscar sugestões para realização de trabalhos. Aos colegas Claudio Diniz e Guilherme Freitas que chegaram um pouco depois, mas se integraram perfeitamente ao nosso laboratório, sempre participando dos debates e colaborando para a resolução de problemas. Enfim a todos os amigos que fizeram e fazem o nosso "lab215" tão forte, unido e dedicado, meu muito obrigado.

Por fim, gostaria de agradecer a todos os amigos e colegas que de alguma forma me apoiaram durante este período e que não foram nominalmente citados, sejam eles do futebol, dos churrascos, das conversas descontraídas da "hora do cafezinho", das caronas e das comemorações, fica meu sincero agradecimento a todos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

1-D	Uma Dimensão
2-D	Duas Dimensões
AVC	*Advanced Video Coding*
ASO	*Arbitrary Slice Order*
BAC	*Binary Arithmetic Coder*
BAE	*Binary Arithmetic Engine*
BAD	*Binary Arithmetic Decoder*
BADE	*Binary Arithmetic Decoder Engine*
BIN	*Bit* de um *binstring*
BINSTRING	*String* de *bits* binarizados
BITSTREAM	Fluxo de *bits*
BS	*Bitstream*
CABAC	*Context-Based Adaptive Binary Arithmetic Coding*
CABAD	*Context-Based Adaptive Binary Arithmetic Decoding*
CAVLC	*Context-Based Adaptive Variable Length Coding*
CCIR	*Consultative Committee for International Radio*
CIF	*Common Intermediate Format*
CODEC	Codificador e Decodificador
D1	*Standard Definition*
DC	*Direct Current*
DCT	*Discrete Cosine Transform*
DCT 2-D	*Discrete Cosine Transform in Two Dimensions*
DDR	*Double Data Rate*
DPCM	*Differential Pulse Code Modulation*
DRAM	*Dynamic Random Access Memory*
DSP	*Digital Signal Processor*
DVD	*Digital Versatile Disk*
FDCT	*Forward Discrete Cosine Transform*
FDCT 2-D	*Forward Discrete Cosine Transform in Two Dimensions*

FIFO	*First In First Out*
FIR	*Finite Impulse Response*
FPGA	*Field Programmable Gate Array*
FRExt	*Fidelity Range Extensions*
GB	*Gigabytes*
HDTV	*High Definition Digital Television*
H422P	*High 4:2:2 Profile*
H444P	*High 4:4:4 Profile*
Hi10P	*High 10 Profile*
HP	*High Profile*
HSI	*Hue, Saturation, Intensity*
IDCT	*Inverse Discrete Cosine Transform*
IDCT 2-D	*Inverse Discrete Cosine Transform in Two Dimensions*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electric and Electronics Engineers*
INTER	*Inter Prediction*
INTRA	*Intra Prediction*
ISCAS	*IEEE International Symposium on Circuits and Systems*
ISO	*International Organization for Standardization*
ITU-T	*International Telecommunication Union - Telecommunication*
JVT	*Joint Video Team*
LCD	*Liquid Crystal Display*
LPS	*Least Probable Symbol*
MB	*Megabytes*
MCAB	Módulos de Codificação Aritmética Binária
MC	*Motion Compensation*
ME	*Motion Estimation*
MPEG	*Moving Picture Experts Group*
MSB	*Most Significant Bit*
MPS	*Most Probable Symbol*
NAL	*Network Adaptation Layer*
PC	*Personal Computer*
PCI	*Peripheral Component Interface*
PPG	Processsador de propósito geral
PSNR	*Peak Signal-to-Noise Ratio*

Q	*Quantization*
Q^{-1}	*Inverse Quantization*
QCIF	*Quarter Common Intermediate Format*
QP	*Quantization Parameter*
RAM	*Random Access Memory*
RGB	*Red, Green, Blue*
RLE	*Run Length Encoding*
RLP	Registrador de Linha de Pesquisa
RLB	Registrador de Linha de Busca
SAE	*Sum of Absolute Errors*
SAD	*Sum of Absolute Differences*
SBTVD	Sistema Brasileiro de Televisão Digital
SDTV	*Standard Definition Television*
SP	*Switching P*
SI	*Switching I*
T	*Transform*
TB	*Terabyte*
T^{-1}	*Inverse Transform*
UFRGS	Universidade Federal do Rio Grande do Sul
UP	Unidade de Processamento
USB	*Universal Serial Bus*
VCEG	*Video Coding Experts Group*
VCL	*Video Coding Layer*
VGA	*Video Graphics Array*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*
VLC	*Variable Length Coding*
XUP	*Xilinx University Program*
YUV	*Espaço de cores, sinônimo de YCbCr*
YCbCr	*Luminance, Chrominance Blue, Chrominance Red*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

RESUMO

As inovações tecnológicas têm propiciado transformações nas formas de interação e, principalmente, na comunicação entre as pessoas. Os avanços nas áreas de tecnologia da informação e comunicações abriram novos horizontes para a criação de demandas até então não existentes. Nesse contexto, a utilização de vídeo digital de alta definição para aplicações de tempo real ganha ênfase. Entretanto, os desafios envolvidos na manipulação da quantidade de informações necessárias à sua representação, fomentam pesquisas na indústria e na academia para minimizar os impactos sobre a largura de banda necessária para transmissão e/ou no espaço para o seu armazenamento.

Para enfrentar esses problemas diversos padrões de compressão de vídeo têm sido desenvolvidos sendo que, nesse aspecto, o padrão H.264/AVC é considerado o estado da arte. O padrão H.264/AVC introduz ganhos significativos na taxa de compressão, em relação a seus antecessores, porém esses ganhos vêem acompanhados pelo aumento na complexidade computacional das ferramentas aplicadas como, por exemplo, a Codificação Aritmética Binária Adaptativa ao Contexto (CABAC). A complexidade computacional relacionado ao padrão H.264/AVC é tal que torna impraticável sua execução em *software* (para operar em um processador de propósito geral, ao menos para nos disponíveis atuais) com a finalidade de realizar a codificação ou decodificação em tempo real para sequências de vídeo de alta definição.

Esta dissertação apresenta uma arquitetura de *hardware* para o processo de decodificação do CABAC, conforme especificação do padrão H.264/AVC. Tendo o objetivo de contribuir para a resolução de alguns dos problemas relacionados à tarefa de decodificação de vídeo de alta definição em tempo real. Para isso, apresenta-se uma introdução sobre conceitos fundamentais da compressão de dados e vídeo digital, além da discussão sobre as principais características do padrão H.264/AVC. O conjunto de algoritmos presentes no CABAC e o fluxo de decodificação do CABAC são descritos em detalhes. Para fundamentar as decisões de projeto um vasto conjunto de experimentos foi realizado para analisar o comportamento do *bitstream* durante o processo de decodificação do CABAC. A arquitetura de *hardware* proposta e desenvolvida é apresentada em detalhes, tendo seu desempenho comparado com outras propostas encontradas na literatura.

Os resultados obtidos mostram que a arquitetura desenvolvida é eficaz em seu objetivo, pois atinge a capacidade de processamento de vídeos em alta definição (HD1080p) em tempo real. Além disso, os experimentos realizados deram origem a observações inovadoras, que permitiram determinar os pontos chave para minimizar os gargalos inerentes ao conjunto de algoritmos que compõe o CABAC.

Palavras-Chave: Compressão de Vídeo, Codificação de Entropia, Codificação Aritmética, CABAC, CABAD, Arquitetura de Hardware para o Decodificador de Vídeo segundo o Padrão H.264/AVC.

Algorithms and Architecture Design for Context-Adaptive Binary Arithmetic Coder for the H.264/AVC Decoder

ABSTRACT

The technological innovations of recent decades have brought changes in the forms of human interaction especially in communication area. Advances in the areas of information technology and communications opened new horizons for creating demands non-existent so far. In this scenario the high-definition digital video for real-time applications has gained emphasis for this context. However, the challenges involved in handling the amount of information necessary for its representation, promoting research in industry and academia to minimize the impact on the bandwidth needed for transmission and / or the space for the storage.

To address those problems several video compression standards have been developed and the H.264/AVC standard is the state-of-the-art. The H.264/AVC standard introduces significant gains in compression rate, compared to its predecessors. These gains are obtained by an increase in computational complexity of the techniques used, such as the CABAC. The computational requirements of H.264/AVC standard is so strong that make its implementation impractical in *software* (to operate on a general purpose processor) for the purpose of performing encoding or decoding in real time for high-definition video sequences.

This dissertation presents a new CABAD architecture with the implementation in *hardware* intended to solve the problems related to the task of decoding high-definition video in real time. An introduction to fundamental concepts of data compression and digital video is presented, in addition to discussing the main features of the H.264/AVC standard. The set of algorithms the CABAC and of the CABAD decode flow are described in detail. A wide number of experiments were conducted to identify the static and dynamic behavior of the bitstream to support the design decisions. At the end the developed architecture is examined and compared with other proposals found in literature.

The results show that the architecture developed is effective in its purpose to handle high-definition video (HD1080p) in real time. Furthermore, the experiments have led to innovative observations to determine the key points to minimize the bottlenecks inherent in the set of algorithms that make the CABAD.

Keywords: Video Compression, Entropy Encoding, Arithmetic Encoding, CABAC, CABAD, Hardware Architecture for the H.264/AVC Video Standard Decoder.

1 INTRODUÇÃO

Os avanços na pesquisa científica, intensificados nas últimas quatro décadas, tem propiciado uma evolução expressiva em diversas áreas do conhecimento. A disseminação e adoção de inovações tecnológicas que, a cada dia, se fazem mais presentes em nosso cotidiano podem ser consideradas produto deste processo. Assim, gradativamente, realizações técnico-científicas fomentam transformações na sociedade à medida que oferecem alterações no modo de vida que podem culminar em desenvolvimento sócio-cultural e econômico. Acontecimentos como a expansão da comunicação móvel, a massificação dos computadores pessoais e a Internet mudaram nosso estilo de vida, trazendo alterações significativas na forma e na dinâmica dos relacionamentos entre pessoas e instituições, constituindo-se em um dos pilares de sustentação do processo de globalização (MASSI, 1998).

Atualmente, é possível observar uma forte demanda do mercado por produtos eletrônicos voltados ao entretenimento, sejam para jogos, músicas, fotos e/ou vídeos. Em especial, o crescente interesse por vídeo digital pode ser reflexo das diversas possibilidades de aplicações para este tipo de recurso. Os nichos de aplicação para produtos que manipulem vídeo digital são bastante diversificados passando por dispositivos portáteis como, por exemplo, telefones celulares, câmeras, filmadoras, CD/DVD *players,* computadores pessoais, Internet TV e por televisores (*set-top-box*) com capacidade de receber sinal de TV digital. Obviamente, a utilização de vídeo digital sobre cada um destes dispositivos apresenta desafios distintos, necessitando de soluções que são específicas para cada caso.

Manipular vídeo digital representa um desafio para os sistemas computacionais empregados atualmente, pois a representação de uma sequência de vídeo em formato digital exige uma quantidade extremamente elevada de informações, principalmente, considerando cenários onde as resoluções de alta definição (HD) são adotadas. Tarefas como, armazenamento e ou transmissão, tornam-se, praticamente, inviáveis considerando-se vídeo digital sem compressão. Por exemplo, para representar um vídeo digital com resolução de 640x480 *pixels* com amostragem de 30 quadros por segundo (comumente denominado de VGA), utilizando 24 bits por *pixel* (padrão de cores RGB), a quantidade de informações produzida seria, aproximadamente, 211 milhões de bits por segundo (211 Mbps). Para armazenar uma sequência de curta duração, com 10 minutos, seriam necessários mais de 15 bilhões de bytes (15GB). Considerando vídeo digital de alta resolução como, por exemplo, 1920x1088 *pixels* com amostragem de 30 quadros por segundo (usual em televisão digital com alta definição ou HDTV), com 24 bits por *pixel*, a quantidade de informações sobe para 1,4 bilhões de bits por segundo (1,4 Gbps) e seriam necessários 105 bilhões de bytes (105 GB) para armazenar uma sequência com a mesma duração de 10 minutos (AGOSTINI, 2007). Para armazenar um vídeo, nas condições supracitadas, com duração típica de um filme (entre 80 e 100 minutos), seriam necessários quase um trilhão de bytes (1 TB).

Neste contexto, a compressão de vídeo surge como fator decisivo para o sucesso de aplicações e dispositivos que manipulam vídeos digitais, pois permite redução dos custos relativos ao armazenamento e transmissão, tornando economicamente viável a utilização destas tecnologias.

> A compressão de vídeo parte do princípio que a grande quantidade de informação necessária para representar as sequências de vídeos no formato digital pode ser reduzida drasticamente, pois essas sequências possuem, em geral, uma importante propriedade intrínseca: apresentam elevado grau de redundância. Isto significa que boa parte da enorme quantidade de informação empregada na sua representação é supérflua (AGOSTINI, 2007).

Baseado nesta observação, a compressão de vídeo ganha destaque, uma vez que seu objetivo é, justamente, explorar técnicas que ofereçam novas formas de representar uma sequência de vídeo, minimizando a redundância de informação através da supressão dos dados supérfluos. Deste modo, torna-se possível representar uma sequência de vídeo, em formato digital, com uma quantidade de informações duas ordens de grandeza menor que a quantidade originalmente necessária. (AGOSTINI, 2007).

O desenvolvimento de novas técnicas de compressão de vídeo que ofereçam redução significativa das redundâncias presentes neste tipo de conteúdo tem sido objeto de pesquisa, tanto da indústria quanto da academia, as quais já resultaram na criação de diversas técnicas para compressão de vídeo que, apesar de enfatizarem o mesmo objetivo, adotam estratégias diferentes (SAID, 2004); possibilitando transformar a representação do vídeo para domínios diferentes a fim de empregar os métodos mais adequados para minimizar a incidência das redundâncias com base na observação de suas propriedades de ocorrência. A partir da utilização combinada de diferentes técnicas de compressão de vídeo, começaram a ser definidos os padrões de compressão de vídeo, também denominados “compressores”. Esses padrões são formados por um conjunto de algoritmos e técnicas que, quando empregados em determinada ordem (etapas), produzem uma representação comprimida da sequência de vídeo original.

Dentre os diversos compressores de vídeo digital da atualidade o padrão H.264/AVC (ITU-T, 2003) é considerado o estado da arte, pois atinge taxas de compressão mais elevadas quando as penalidades perceptivas à qualidade do vídeo são equivalentes, pois o método é mais sofisticado (RICHARDSON, 2003). Porém, o padrão H.264/AVC introduz um incremento significativo na complexidade computacional tipicamente elevada dos padrões de compressão de vídeo. A utilização do padrão H.264/AVC para comprimir uma sequência de vídeo abre a possibilidade de escolher entre duas alternativas: I - utilizar uma qualidade mais elevada, mantendo inalterada a largura de banda necessária para a transmissão ou o espaço de armazenamento; ou II - reduzir a largura de banda necessária para a transmissão ou o espaço de armazenamento, mantendo a mesma qualidade.

Outro aspecto relevante é que, dado o nível de complexidade computacional inerente aos padrões de compressão de vídeo, a tarefa de desenvolver codificadores e decodificadores através de soluções baseadas em *software* operando sobre um processador de propósito geral (PPG) tem sido um desafio potencializado pelas peculiaridades do padrão H.264/AVC. Além disso, cabe salientar que soluções baseadas em *software* podem ser extremamente limitadas para cenários onde as resoluções de alta definição e a execução em tempo real são combinadas. Não obstante, no contexto da computação móvel, as soluções baseadas em *software* podem ser economicamente

inviáveis, pois o PPG embarcado nestes dispositivos geralmente não possui grande capacidade computacional quando comparado com computadores pessoais, estações de trabalho e/ou servidores.

Como acontece com grande parte dos padrões de compressão de vídeo da atualidade o padrão H.264/AVC também é composto por um vasto conjunto de ferramentas de propósitos distintos dentre as quais podem ser citadas: estimação de movimento, compensação de movimento, transformadas, quantização, filtro redutor de efeitos de bloco e a codificação de entropia.

Para enfrentar os desafios impostos pelo padrão H.264/AVC tem-se, primeiramente, que definir qual o processo de interesse (compressão ou descompressão) e o perfil desejado, pois a norma de especificação do padrão H.264/AVC define o conjunto de ferramentas que devem ser empregadas a cada perfil (ITU-T, 2003). A seguir, os gargalos do sistema devem ser identificados. Entretanto, avaliar a complexidade computacional de cada ferramenta que compõe o padrão exigiria um esforço considerável. Felizmente, trabalhos disponíveis na literatura já tratam este tema (KALVA, 2006; OSTERMANN, 2004; PURI, 2004; SULLIVAN, 2004).

As análises de complexidade sobre ferramentas empregadas na construção de um decodificador de vídeo digital compatível com o perfil "***Main***" do padrão H.264/AVC apresentadas na literatura indicam que a etapa de estimação de movimento possui a maior complexidade computacional em termos de numero de operações, porém essa etapa permite exploração de elevado grau de paralelismo (MARPE, SCHWARZ, WIEGAND, 2003). Considerando o processo de decodificação o gargalo do sistema reside no bloco de decodificação de entropia embora este não seja o bloco com maior complexidade (KALVA, 2006).

O padrão H.264/AVC define três métodos para codificação de entropia, sendo que destes os dois principais são o CABAC - codificação binária aritmética adaptativa ao contexto (na sigla do inglês: *Context Adaptive Binary Arithmetic Coder*) e o CAVLC - codificação adaptativa por palavra de código com tamanho variável (na sigla do inglês: *Context Adaptive Vary-Length Coder*). Quando empregados em um decodificador de vídeo compatível com o padrão H.264/AVC ambos os métodos de codificação entrópica realizam o processo inverso, neste caso, o CABAC realiza a decodificação aritmética binária adaptativa ao contexto - CABAD (na sigla do inglês: *Context Adaptive Binary Arithmetic Decoder*). (ITU-T, 2003).

A utilização do CABAC como método de codificação de entropia permite obter ganhos entre 9% e 15% sobre a taxa de compressão quando comparado aos resultados obtidos com a utilização do CAVLC. (MARPE, SCHWARZ, WIEGAND, 2003). Entretanto, observa-se que a adoção do CABAC implica em um considerável acréscimo da complexidade computacional. Esse fato está diretamente relacionado com a natureza essencialmente sequencial dos algoritmos empregados pelo CABAC, fato que introduz outro grau de desafio para a construção de soluções de alto desempenho, pois dificulta a exploração de paralelismo, constituindo o gargalo desse sistema. (MARPE, SCHWARZ, WIEGAND, 2003).

Relacionar os assuntos que formam a base contextual para o tema central desta dissertação, introduzindo uma visão geral dos desafios relacionados ao processo de decodificação de entropia, foi o foco dos tópicos já relatados. Na sequência a seção 1.1 contempla o objetivo do trabalho ora proposto, enquanto a seção 1.2 é destinada à apresentação da justificativa e motivação do mesmo.

A estrutura do restante do texto está organizada da seguinte maneira: no Capítulo dois são introduzidos alguns conceitos básicos sobre compressão de vídeo digital, que poderão ser úteis para a compreensão dos capítulos subsequentes; o Capítulo três apresenta, de forma superficial, as principais características do conjunto de ferramentas empregadas pelo padrão H.264/AVC; o Capítulo quatro detalha o conjunto de algoritmos que compõem o CABAC, além de discutir as características que tornam seu comportamento essencialmente sequencial; o Capítulo cinco apresenta análises sobre características de comportamento estático e dinâmico do CABAD baseado em dados estatísticos coletados através de simulações em *software* sobre um conjunto de *benchmarcks* padronizados; no Capítulo seis as arquiteturas de *hardware* desenvolvidas são apresentadas, sendo que as escolhas são discutidas e fundamentadas; no Capítulo sete apresentam-se comparações das soluções desenvolvidas com trabalhos encontrados na literatura; por fim, no Capítulo oito, são apresentadas as conclusões atingidas no decorrer deste trabalho e as propostas para trabalhos futuros.

1.1 Objetivo

O objetivo principal desta dissertação foca o desenvolvimento de uma arquitetura de *hardware* dedicada ao processo de decodificação de entropia realizado pelo CABAC, conforme restrições impostas pelo padrão H.264/AVC, considerando o perfil "***Main***" e o nível 4.0. Entretanto, frente ao elevado grau de complexidade do conjunto de algoritmos que definem o processo de decodificação do CABAC se faz necessário definir alguns objetivos secundários para nortear o estudo realizado, sendo eles: I – a investigação detalhada do funcionamento destes algoritmos; II – a análise das restrições de desempenho impostos pela natureza destes algoritmos; e III – a análise de comportamento dinâmico do *bitstream* gerado pelo processo de codificação do padrão H.264/AVC para explorar a possibilidade de inovações arquiteturais.

1.2 Motivação e Justificativa

Além da eficiência relativa à taxa de compressão outro aspecto que, para o cenário nacional, confere especial relevância ao padrão H.264/AVC está relacionado à norma que define o sistema brasileiro de televisão digital (SBTVD). Esta norma determina que o conteúdo televisivo a ser transmitido via *broadcast*, utilizando como meio de transmissão o espectro eletromagnético, deva passar por um processo de codificação conforme especificações do padrão H.264/AVC. Esta definição leva à necessidade de embarcar, em cada receptor (televisor) para o sinal de televisão digital gerado no Brasil, um decodificador capaz de fazer o processo de descompressão para restaurar o conteúdo do vídeo digital para possibilitar a sua exibição.

Ao avaliar as restrições impostas para a utilização de um decodificador de vídeo que receba o sinal de televisão digital de alta definição e que opere em tempo real em relação a questões econômicas como custo e a escala de produção, pode-se concluir que o desenvolvimento de uma arquitetura de *hardware* dedicada para executar o processo de decodificação torna-se quase imperativo, devido a suas vantagens quando comparado a uma solução baseada em *software* operando sobre um PPG. Além disso, há o desejo de oferecer uma contribuição prática para construção de um protótipo de sistema de decodificação de vídeo que vêem sendo desenvolvido no âmbito do SBTVD.

2 CONCEITOS SOBRE VÍDEO DIGITAL E COMPRESSÃO DE DADOS

Este capítulo tem por objetivo apresentar conceitos relacionados ao processo de geração e compressão de vídeo digital. Para isso, serão explorados conceitos básicos como a forma de captura e a composição do vídeo digital através da amostragem espacial e temporal. Além disso, serão discutidas questões relativas à identificação das redundâncias presentes no vídeo digital. Os tópicos abordados neste capítulo são importantes para a compreensão dos demais capítulos presentes nesta dissertação.

2.1 Conceitos Básicos

Uma cena visual natural é contínua no tempo e no espaço, porém, representar uma cena do mundo real de forma digital envolve um processo de amostragem espacial e temporal. Para se obter uma imagem amostrada, a câmera foca uma projeção bidimensional da cena em um sensor. Para o caso de captura de imagens coloridas, cada componente de cor é separada e projetada em um sensor. Cada imagem capturada é composta por um conjunto de pontos, organizados em uma matriz bidimensional com largura e altura definidas de acordo com o número de pontos de amostragem em cada eixo. Repetindo-se o processo de aquisição de imagens de forma sucessiva, cadenciada por um intervalo de tempo entre as iterações, obtém-se uma sequência de imagens, onde cada imagem desta sequência é intitulada quadro. A exibição destes quadros de forma subseqüente, com pequenos intervalos de tempo, cria a sensação de movimento, que caracteriza o vídeo digital (RICHARDSON, 2003). A Figura 2.1 ilustra uma sequência de imagens capturadas e exibidas de forma subsequentes como ocorre em um vídeo digital, destaca-se também a matriz com os pontos de amostragem de cada imagem.

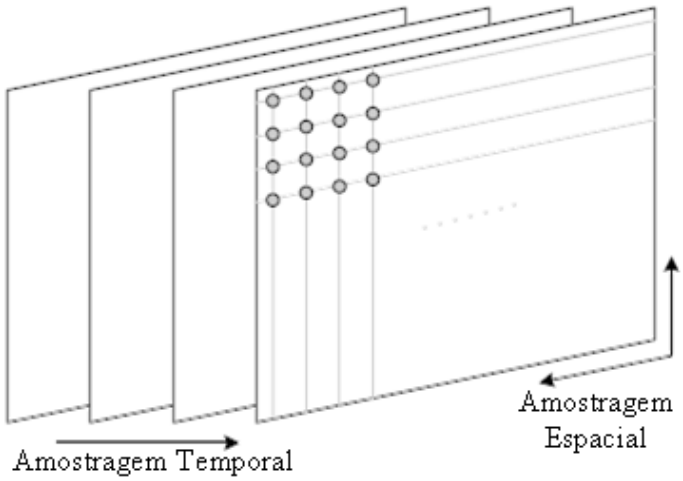


Figura 2.1: Sequência de quadros de um vídeo digital.

Cada ponto de amostragem espacial-temporal da imagem, denominado elemento de figura ou *pixel* (na sigla do inglês: *Picture Element*), é definido por um número ou

conjunto de números que representam a cor ou a combinação de brilho (luminosidade) e cor, dependendo do espaço de cores utilizado. A resolução da imagem capturada está relacionada com as dimensões da matriz de amostragem utilizada. Em geral, quanto maior a resolução melhor a qualidade da imagem, pois cada ponto representa uma região menor da cena natural projetada, oferecendo maior fidelidade entre a percepção da cena natural e do vídeo digital.

Avaliar a qualidade visual de um vídeo digital é uma tarefa difícil devido à diversidade de fatores envolvidos. Todavia, observa-se que a quantidade de pontos de amostragem e a forma como essas cores são representadas têm impacto significativo sobre a qualidade do vídeo digital percebido pelo sistema visual humano. Assim, uma das formas de melhorar a qualidade do vídeo digital é aumentar o número de pontos em cada eixo da matriz de amostragem, ou seja, modificar a resolução da imagem. Neste ponto, identificam-se dois interesses conflitantes: qualidade visual e quantidade de informação, pois aumentar a resolução da imagem tem impacto direto sobre a quantidade de informação necessária à representação digital da cena.

Outra relação interessante é que, em geral, quanto maior a quantidade de pontos de amostragem menor a distância física entre eles e, consequentemente, maior a probabilidade que os pontos próximos uns dos outros, ou seja, *pixels* vizinhos possuam amostras de mesmo valor, fato que caracteriza a redundância espacial. O mesmo efeito pode ser observado entre os quadros de uma sequência de vídeo, pois quanto maior a frequência de captura entre os quadros menor a distância temporal entre os eles e, dessa forma, maior a probabilidade de que quadros temporalmente próximos uns dos outros possuam elevado grau de correlação, fato que caracteriza a redundância temporal.

Explorar as características encontradas no sinal de vídeo digital a fim de produzir uma representação que minimize a incidência das redundâncias, tipicamente presentes neste tipo de sinal, é o objetivo da codificação de vídeo digital. O processo de codificação consiste das etapas de compressão e descompressão. Durante a etapa de compressão o sinal de vídeo digital é processado através de um conjunto de algoritmos que visam gerar uma nova representação do sinal original que minimize a incidência de redundâncias. Posteriormente, na etapa de decodificação, sob essa nova representação será processada aplicado o processo inverso para restaurar o sinal de vídeo original.

2.1.1 Amostragem Espacial

O dispositivo de carga acoplado (CCD) é um sensor para captação de imagens a partir de uma cena (WIKIPÉDIA, 2009). A saída do CCD é um sinal analógico, uma variação do sinal elétrico que representa uma imagem. Capturando imagens em pequenos intervalos de tempo, se produz uma sequência de imagens, ou quadros, que têm valores definidos em um conjunto de pontos de amostragem. A forma mais comum da imagem amostrada é um retângulo com os pontos de amostragem posicionados em uma matriz quadrada ou retangular. A amostragem ocorre em cada intersecção das linhas com as colunas da matriz de captura, sendo que a imagem amostrada pode ser reconstruída representando cada amostra como um elemento de figura quadrado. A qualidade da imagem é influenciada pela quantidade de pontos de amostragem (*pixels*).

2.1.2 Amostragem Temporal

Cada quadro de um vídeo é capturado por um sistema eletrônico de sensores, em intervalos regulares de tempo. Quando esta sequência de quadros é apresentada de

forma subsequente, com pequeno intervalo de tempo entre cada quadro, obtém-se a sensação de movimento. Quanto mais elevada a frequência em que os quadros são capturados (*frame rate*) mais suave será a sensação de movimento. Entretanto, uma quantidade maior de amostras deve ser capturada e armazenada, aumentando a complexidade para o tratamento e o armazenamento desses vídeos. Frequências de amostragem entre 25 e 30 quadros por segundo são consideradas adequadas para reproduzir sensação de movimento semelhante a percebida pelo sistema visual humano em cenas naturais e são um padrão para os televisores analógicos. Taxas entre 10 e 20 quadros por segundo são utilizados para a comunicação de vídeos em canais de transmissão de baixa capacidade, taxas de 50 ou 60 quadros por segundo são utilizados para vídeos de alta qualidade.

2.1.3 Quadros e Campos

Um sinal de vídeo pode ser amostrado como uma série de quadros progressivos (frames – *progressive sampling*) ou como uma sequência de campos entrelaçados (*interlaced sampling*). Em uma sequência de vídeo entrelaçado, metade dos dados de um quadro (um campo) é amostrada a cada intervalo temporal de amostragem. Um campo é formado apenas por linhas pares ou por linhas ímpares de um quadro de uma sequência inteira de quadros. Uma sequência de vídeo entrelaçado contém uma sequência de campos, cada um representando metade da informação de um quadro do vídeo.

2.2 Espaço e Subamostragem de Cores

A representação digital de um vídeo colorido está associada à forma de interpretação das cores do sistema visual humano. Um sistema para representação de cores é denominado “espaço de cores” e a definição do espaço de cores a ser utilizado para representar um vídeo é essencial para uma eficiente codificação do vídeo. Existem diversos espaços de cores que são utilizados para representar imagens digitais, tais como: RGB, HSI e YCbCr (SHI, SUN, 1999). O espaço de cores RGB é um dos mais comuns, tendo em vista que é este o espaço de cores utilizado nos monitores coloridos. O RGB representa, em três matrizes distintas, as três cores primárias captadas pelo sistema visual humano: vermelho, verde e azul (AGOSTINI, 2007).

No espaço de cores YCbCr, as três componentes utilizadas são luminância (Y), que define a intensidade luminosa ou o brilho, crominância azul (Cb) e crominância vermelha (Cr) (MIANO, 1999). Os componentes R, G e B possuem um elevado grau de correlação, tornando difícil o processamento de cada uma das informações de cor de forma independente. Por isso, a compressão de vídeos é aplicada para espaços de cores do tipo luminância e crominância, como o YCbCr (RICHARDSON, 2002). No espaço de cores RGB não há separação entre as componentes de cor e a luminosidade enquanto no espaço de cores YCbCr existe maior separação entre a informação de cor e a informação de brilho ou luminosidade. Deste modo, estas informações podem ser tratadas de forma diferenciada pelos compressores de imagens estáticas e vídeos.

> O sistema visual humano possui maior sensibilidade para a luminosidade ou brilho do que as informações de cores contidas nas imagens (GONZALEZ, 2003). Por esse motivo os padrões de compressão de imagens estáticas e vídeos explorão esta característica psicovisual humana para aumentar a eficiência de codificação através da redução da taxa de amostragem dos componentes de crominância em relação aos componentes de luminância

> (RICHARDSON, 2002). Esta operação é chamada de subamostragem de cores e é realizada sob o espaço de cores YCbCr nos padrões de compressão de vídeos atuais (AGOSTINI, 2007).

A representação da cor de cada posição através de um conjunto de informações de cor e luz (crominância e luminância) possibilita explorar diversas relações de subamostragem dentre essas variáveis, sendo os formatos 4:4:4, 4:2:2 e 4:2:0 os mais utilizados. No formato 4:4:4 a técnica de subamostragem não é aplicada, pois para cada amostra de luminância (Y) são armazenadas um par de amostras de crominância (uma amostra de crominância azul (Cb) e uma de crominância vermelha (Cr)). No formato 4:2:2, para cada quatro informações de Y apenas duas informações de Cb e Cr são representadas, e para o formato 4:2:0, para cada quatro informações de Y apenas uma informação de Cb e uma de Cr são representadas (AGOSTINI, 2007).

> A subamostragem de cor aumenta significativamente a eficiência da compressão, uma vez que parte da informação da imagem é simplesmente descartada, sem causar impacto visual perceptível. Considerando o formato 4:2:0 como exemplo, uma vez que cada componente de crominância possui exatamente um quarto das amostras presentes no componente de luminância. Dessa forma, um vídeo que utilize o espaço de cores YCbCr, no formato 4:2:0, irá utilizar exatamente a metade das amostras necessárias para um vídeo RGB ou YCbCr no formato 4:4:4, resultando em uma taxa de compressão de 50%, apenas com a técnica de sub-amostragem (AGOSTINI, 2007).

2.3 Redundância de Dados na Representação de Vídeos

O princípio básico do processo de compressão é baseado na identificação e minimização de dados redundantes existentes em vídeos e imagens. Um dado é considerado redundante quando a sua utilização não acrescenta informação nova ou relevante para a representação da imagem. Existem, basicamente, quatro tipos diferentes de redundâncias exploradas na compressão de vídeos: redundância espacial, redundância temporal, redundância psicovisual e redundância entrópica. Em virtude da existência de controvérsias entre os autores a respeito da definição dos tipos de redundância, é necessário destacar que este trabalho adota a definição e classificação apresentada em Shi e Sun (1999) e Gonzalez (2003).

As características do sistema visual humano fazem com que não consigamos captar alguns tipos de informações que podem estar presentes na imagem. Algumas informações da imagem, como o brilho, por exemplo, são mais importantes para o sistema visual humano que as cores. Este tipo de informação é classificada como irrelevante, pois apesar não ser sobressalente sua omissão não resulta em perda significativa da qualidade visual percebida pelo sistema visual humano (GONZALEZ, 2003). Mesmo existindo diferenças semânticas entre as palavras irrelevância e redundância por uma questão de uniformidade de tratamento nós iremos utilizar o termo “redundâncias psicovisuais” com o mesmo significado de “irrelevâncias psicovisuais”.

Para comprimir uma imagem explorando as irrelevâncias psicovisuais parte da informação original da imagem é eliminada de forma irreversível pelo codificador. Existem dois tipos principais de processos utilizados para este fim. O primeiro, intitulado subamostragem, é utilizado na entrada do vídeo e elimina parte das informações de cores. O segundo, conhecido por quantização, normalmente é aplicado no domínio das frequências e elimina ou atenua as frequências de menor percepção para

o sistema visual humano. Ambos os processos empregam a codificação com perdas. É importante destacar que a eliminação destas informações contribui para que o codificador atinja elevadas taxas de compressão com pequeno impacto na qualidade visual da imagem que, eventualmente, pode ser imperceptível (AGOSTINI, 2007).

A redundância espacial, também conhecida como "redundância *intra-frame*" (GHANBARI, 2003), é resultado da correlação existente entre os *pixels* vizinhos do mesmo quadro. Os pixels vizinhos tendem a possuir valores semelhantes e, por consequência, com elevado grau de redundância de informação, gerando pouco acréscimo de informação visual a cada ponto em relação aos vizinhos.

> A redundância temporal, também denominada "redundância *inter-frame*" ou redundância entre quadros (GHANBARI, 2003), é causada pela correlação existente entre quadros subsequentes. Na verdade, a redundância temporal poderia ser classificada como apenas mais uma dimensão da redundância espacial, como faz (GONZALEZ, 2003). Muitos blocos de *pixels* simplesmente não mudam de valor de um quadro para outro do vídeo como, por exemplo, a parte da imagem que corresponde ao fundo (*background*). Outros *pixels* apresentam uma pequena variação de valores que pode ser causada, por exemplo, por uma variação de iluminação (AGOSTINI, 2007).

Também é possível que o bloco de *pixels* simplesmente tenha se deslocado de um quadro para o outro como, por exemplo, quando do movimento de um objeto da cena ou da própria câmera. Estes fatores resultam em um pequeno acréscimo de informação visual de cada imagem com relação às imagens vizinhas (AGOSTINI, 2007).

A redundância entrópica está relacionada com a forma de representação computacional dos símbolos codificados e não se relaciona diretamente ao conteúdo da imagem. A entropia é uma medida da quantidade média de informação transmitida por símbolo do vídeo (SHI, SUN, 1999). A quantidade de informação nova transmitida por um símbolo diminui na medida em que a probabilidade de ocorrência deste símbolo aumenta. Então, os codificadores que exploram a redundância entrópica têm por objetivo transmitir o máximo de informação possível por símbolo codificado e, deste modo, representar mais informações com um número menor de símbolos. Um diferencial da codificação de entropia reside no fato que o conjunto de técnicas e algoritmos aplicados ao processo de compressão atinge seu objetivo sem inserir perdas, ou seja, é possível restaurar o conteúdo original na íntegra (AGOSTINI, 2007).

Os padrões de codificação de vídeo atuais empregam um vasto conjunto de ferramentas que visam explorar a eliminação de um determinado tipo redundância. A representação comprimida do vídeo digital é obtida pela aplicação sucessiva deste conjunto de ferramentas sobre o sinal de vídeo não comprimido. A Figura 2.2 apresenta o conjunto típico de ferramentas, aplicadas pelos padrões de codificação de vídeo da atualidade, organizadas de acordo com o tipo de redundância que cada qual atua e pela ordem que elas são comumente aplicadas durante o processo de compressão.

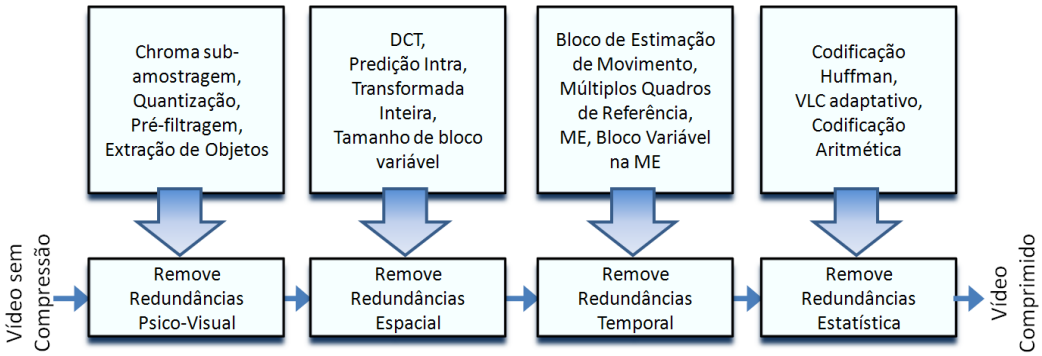


Figura 2.2: Ferramentas aplicadas para exploração de cada tipo de redundância (modificado a partir de Kalva (2006)).

2.4 Conceitos sobre Entropia

No contexto da compressão de dados as técnicas de codificação de entropia são quase ubíquas dada sua eficiência e simplicidade de adaptação a novos cenários e à clara separação entre as etapas de modelagem e de codificação. Entretanto, essa disseminação demorou muito tempo para se concretizar, visto que as primeiras "aparições" do termo entropia remontam a meados do século XVIII quando da definição da primeira lei da termodinâmica que versava sobre as relações de conversão entre energia e calor. Desde então diversos trabalhos complementaram e estenderam a definição deste conceito que, através dos esforços de Ludwig Boltzman, publicados entre 1872 e 1877, culminou na criação de um marco fundamental para o embasamento matemático-estatístico do termo. Contudo, a codificação de entropia não teria atingido a importância atual não fosse à genialidade de Claude Elwood Shannon que combinou, em seus trabalhos, um vasto conjunto de conceitos (de Boltzman à Nyquist) que resultaram na criação da teoria matemática da comunicação, publicada em 1948, a qual forneceu a base para teoria da informação como conhecemos hoje (HANKERSON, HARRIS, PETER, 2003; SAYOOD, 2003; SHANNON, 1948).

Atualmente o conceito de entropia apresenta significados diversos dependendo do contexto no qual é empregado. Considerando o contexto da teoria da informação uma das definições mais usuais é a de Sayood (2003) que define entropia como "uma métrica para avaliar a incerteza sobre uma variável aleatória" ou, conforme David Salomon o termo entropia é uma informação proporcional ao número mínimo de questões (do tipo sim ou não) que devem ser realizadas para que seja possível atingir a resposta de uma pergunta qualquer (SALOMON, 2000). Em outras palavras, podemos assumir que a entropia é uma medida que representa a quantidade de informação transmitida a partir da ocorrência de um evento qualquer, para o qual conhecemos sua probabilidade de ocorrência. No que tange à compressão de vídeo, a técnica de codificação de entropia é considerada uma forma de compressão sem perdas, pois o processo é totalmente reversível, ou seja, uma entrada codificada pode ser decodificada tendo seu valor integralmente restaurado, sem implicar na perda de informação (MARPE, SCHWARZ, WIEGAND, 2003).

Detalhar a codificação de entropia, passando por todos os conceitos e teorias relacionadas, foge ao escopo deste trabalho. Tal informação pode ser obtida na literatura através de livros e artigos especializados. Entretanto, uma visão geral do tema será apresentada para familiarizar o leitor com as notações empregadas. Na teoria da

informação a entropia significa uma métrica para avaliar a quantidade de informação transmitida a partir da ocorrência de um evento aleatório $\boldsymbol{x}$. Considerando apenas a hipótese deste evento ocorrer ou não e observando-se um número suficientemente grande de iterações deste processo, define-se a informação em termos da probabilidade, dada por $\boldsymbol{p}$, deste evento $\boldsymbol{x}$ ocorrer (MOFFAT, NEAL, WITTEN, 1998). Assim, define-se que a informação, dada por I(p), possui as seguintes propriedades:

1. A informação é quantidade não negativa, ou seja, $\mathbf{I(p) \geq 0}$;
2. Se um evento x possui probabilidade p=1 então a informação obtida a partir da ocorrência deste evento é $\mathbf{I(1) = 0}$;
3. Se dois eventos independentes ocorrem (cuja união das probabilidades é o produto de suas probabilidades individuais) então a informação obtida a partir da ocorrência destes eventos é a soma das duas informações, conforme apresentado pela Equação 2.1;

$$I(p_1 * p_2) = I(p_1) + I(p_2) \tag{2.1}$$

4. Verifica-se que a métrica de avaliação da informação é monotônica e contínua em função da probabilidade, pois pequenas alterações nas probabilidades implicam em pequenas alterações na informação.

Seguindo-se uma sequência de derivações matemáticas (que foge ao escopo deste trabalho), pautadas pelas propriedades enumeradas acima, atinge-se a definição de informação expressa pela Equação (2.2), onde b significa a base de representação dos símbolos gerados, ou seja, o tamanho do alfabeto empregado para representar a informação obtida.

$$I(p) = log_b\left(\frac{1}{p}\right) = -log_b(p) \tag{2.2}$$

Através da Equação 2.2 é possível determinar o número ótimo de informações produzidas para representar um símbolo com probabilidade p em um alfabeto α. Assim, considerando que X é uma variável discreta e aleatória que pode assumir qualquer valor dentro de um alfabeto finito α. Dado que |α| denota a cardinalidade do alfabeto α, que representaremos por n, e supondo que a probabilidade de ocorrência de cada instância desta variável é representada por:

$$p(x) := \text{Probabilidade}\{\, X = x \,\} \forall\, x \in \alpha \tag{2.3}$$

Tem-se que a entropia é definida por:

$$H(X) := \sum_{x \in \alpha}^{n} \mathrm{p}(x)\, log_b\left(\frac{1}{p(x)}\right) \tag{2.4}$$

É necessário fazer ao menos dois esclarecimentos sobre a Equação 2.4. Primeiro é que a base da função logarítmica deve ser a cardinalidade do alfabeto α. Então, para obtermos uma medida de avaliação da entropia em bits basta empregar base $_b$=2. Segundo, se $p(x) = 0$ então o termo $p(x)log_b\left(\frac{1}{p(x)}\right)$ da Equação 2.4 é indeterminado e em tal situação considera-se que $H(X) = 0$.

No próximo capítulo será apresentada uma visão geral das principais ferramentas que compõem o padrão de compressão de vídeo H.264/AVC, no qual muitos destes conceitos são aplicados.

3 O PADRÃO H.264/AVC

O padrão H.264/AVC, também conhecido como MPEG4 parte 10, foi desenvolvido de forma colaborativa pelo grupo de especialistas em codificação de vídeo denominado VCEG (na sigla do inglês: *Video Coding Experts Group*) vinculados à ITU (na sigla do inglês: *International Telecommunication Union*) e pelo grupo de especialistas em imagem em movimento denominado MPEG (na sigla do inglês: *Moving Picture Experts Group*) vinculados à ISO/IEC (na sigla do inglês: *International Organization for Standardization/ International Electrotechnical Commission*). A força conjunta resultante da união de esforços destes dois grupos recebeu o nome de time unido de vídeo - JVT (na sigla do inglês: *Joint Video Team*).

O trabalho do time JVT resultou na publicação, em meados de 2003, de um documento contendo a norma de especificação para um padrão de compressão de vídeo digital, intitulado H.264/AVC. Esta norma descreve o formato, a sintaxe e a semântica dos dados produzidos pelo processo de decodificação e *parsing* do vídeo codificado segundo o padrão H.264/AVC. A norma não trata do processo de codificação. O documento possui mais de 250 páginas, um fato que merece destaque é a afirmação de Gary J. Sullivan, coordenador do JVT, segundo o qual "[...] a norma foi escrita primeiramente para ser precisa, consistente, completa e correta e não para ser particularmente legível". (RICHARDSON, 2003, pg XVII).

Além do texto da norma, foi colocado em domínio público um *software* de referência, o qual tem como função servir como prova de conceito dos resultados obtidos através da compressão de vídeos digitais pelo padrão H.264/AVC.

Para o escopo deste trabalho se faz necessário apresentar aspectos básicos sobre a estrutura e as ferramentas empregadas pelo padrão H.264/AVC, pois a técnica de codificação de entropia trata as redundâncias relacionadas às palavras de código produzidas durante o processo de codificação a partir de informações geradas por ferramentas que atuam na redução de outros tipos de redundâncias.

3.1 Estrutura do padrão H.264/AVC

No padrão H.264/AVC o sistema de cores utilizado para representar o vídeo de entrada é baseado na distinção entre cor e luminosidade (ou brilho), conforme definido pelo espaço de cores YCbCr (RICHARDSON, 2003).

Para o padrão H.264/AVC a formação de uma imagem codificada pode acontecer a partir de um quadro, quando o vídeo é progressivo ou entrelaçado, ou de um campo, quando o vídeo é entrelaçado (RICHARDSON, 2003).

O padrão H.264/AVC adota ferramentas que tratam os quatro tipos de redundâncias e/ou irrelevâncias presente nos vídeos digitais. Para identificar redundâncias temporais

e/ou espaciais as ferramentas de codificação necessitam fazer comparação entre regiões de um quadro com outras regiões do mesmo quadro ou de quadros subsequentes. Para realizar essa comparação entre regiões o quadro é segmentado em partições. Essas partições são grandes blocos de *pixels* denominados macroblocos que possuem dimensões fixas.

No padrão H.264/AVC o processo de codificação e decodificação de um quadro é orientado em nível de macrobloco, os quais possuem dimensão de 16x16 *pixels* para luminância. Entretanto, cada macrobloco pode ser dividido em submacroblocos e estes em blocos. Cada macrobloco de 16x16 pixels contém amostras de crominância associadas às amostras de luminância que, dependendo do perfil utilizado, podem sofrer variações na quantidade de amostras.

Dentro de cada quadro, os macroblocos são organizados em fatias (*slices*). Uma fatia, ou *slice*, é composta por um grupo de macroblocos e um quadro pode ser composto por uma ou mais fatias, cada qual contendo um número inteiro de macroblocos. O número de macroblocos em uma fatia pode variar de um até o número total de macroblocos do quadro.

Os quadros possuem uma informação de tipo associada, que é definida com base no tipo de predição de movimento utilizada dentro do quadro. Existem três tipos possíveis, sendo: I – apenas predição intraquadros; P – predição inter-quadros referenciando quadro da lista 0; e B – predição inter-quadros podendo referenciar quadros da lista 0 (quadros anteriores), da lista 1 (quadros posteriores) ou de ambas. Além dos três tipos básicos o padrão H.264/AVC prevê a existência de outros dois tipos de *slices*: SI e SP. (RICHARDSON, 2003).

Os slices SP (Switching P) e SI (Switching I) são slices que são transmitidos para permitir o chaveamento entre fluxos de bits diferentes, mas sem causar um grande prejuízo na eficiência de codificação (KARCZEWICZ; KURCEREN, 2003). É comum, em codificação de vídeo, o chaveamento entre fluxos de bits diferentes, mas este chaveamento não pode acontecer logo antes de um quadro P ou B, pois a codificação destes quadros usa como referência quadros do fluxo de bits atual. É possível fazer o chaveamento usando um quadro do tipo I, mas o problema é que estes quadros consomem muito mais bits do que os quadros P ou B. Por isso o padrão H.264/AVC prevê a utilização de slices tipo SP e SI no seu perfil Extended, que será apresentado na próxima seção do texto.

Dentro de *slices* do tipo I só podem ocorrer macroblocos do tipo I, que são codificados usando a codificação intraquadro a partir das amostras do *slice* atual. A codificação pode acontecer sobre macroblocos completos ou para cada bloco. Os macroblocos do tipo I também podem ser codificados através do modo I_PCM, onde o codificador transmite os valores das amostras diretamente, sem predição ou transformação. Em alguns casos anômalos, o modo I_PCM pode ser mais eficiente do que os demais modos de codificação (RICHARDSON, 2003).

Dentro de *slices* do tipo P podem ocorrer macroblocos do tipo I e do tipo P. Macroblocos do tipo P são codificados usando a codificação inter-quadro, a partir de quadros de referência previamente codificados. Um macrobloco codificado no modo inter-quadros pode ser dividido em partições de macroblocos, isto é, em blocos de *pixels* com dimensões de 16x16, 16x8, 8x16 ou 8x8. Se o tamanho de partição escolhido for o 8x8, então cada partição 8x8, denominada de submacrobloco, pode ser dividida novamente em partições de tamanho 8x8, 8x4, 4x8 ou 4x4. Cada partição de

macrobloco é codificada utilizando como referência um quadro da lista 0. Se existir uma partição de submacrobloco, então esta partição é codificada utilizando o mesmo quadro da lista 0 utilizado para codificar a partição de macrobloco. A Figura 3.1 ilustra as possibilidades de divisões do macroblocos e submacroblocos em partições de diferentes tamanhos.

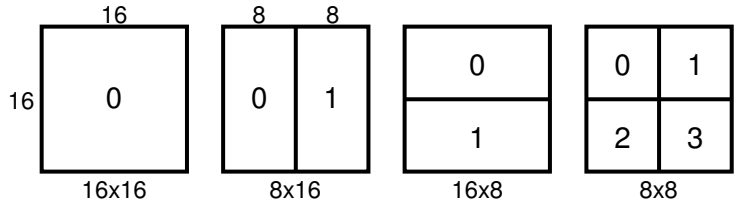


Figura 3.1: Divisão do macrobloco em partições.

Dentro de *slices* do tipo B podem aparecer macroblocos do tipo I e do tipo B. Os macroblocos do tipo B são codificados usando a codificação inter-quadros e permitem que a predição possa ocorrer a partir de um ou dois quadros de referência, que se encontram antes ou depois do quadro corrente, em ordem temporal. Essa possibilidade permite diversas opções para a escolha da predição mais adequada para cada macrobloco em um *slice* do tipo B, permitindo maior eficiência na codificação.

Além dos tipos de macroblocos supracitados o padrão H.264/AVC apresenta um tipo de macrobloco denominado *skip*. Macroblocos do tipo *skip* são macroblocos para os quais nenhuma informação adicional é codificada, nem mesmo o resíduo ou os vetores de movimento. Macroblocos do tipo *skip* são gerados quando o custo da taxa de distorção para codificar o macrobloco é mais elevado do que o custo de não enviar informação alguma sobre o macrobloco (KANNANGARA; RICHARDSON, 2005). Os macroblocos *skip* são gerados apenas pela codificação inter-quadro.

3.2 Perfis e Níveis

Um aspecto interessante do padrão H.264/AVC consiste no fato que o conjunto de ferramentas empregado para o processo de compressão de um vídeo não é único, ou seja, para cada cenário de aplicação existe um subconjunto de ferramentas mais apropriadas que devem ser empregadas em detrimento de outras. Para permitir uma utilização padronizada das diferentes combinações de ferramentas que podem ser aplicadas o padrão H.264/AVC foi organizado em diferentes perfis.

Cada perfil suporta um grupo particular de funções de codificação e especifica o que é necessário para cada codificador e decodificador que seguem este perfil. Não obstante, o padrão H.264/AVC possibilita a utilização de diversas combinações entre a resolução e a taxa de quadros por segundo para os quais os limites são classificados através de níveis.

A primeira versão do padrão H.264/AVC, de maio de 2003 (ITU-T, 2003), define um grupo de três diferentes perfis: *Baseline*, *Main* e *Extended*. O perfil *Baseline* é direcionado a aplicações como videotelefonia, videoconferência e vídeo sem fio. O perfil *Baseline* suporta codificação intra e inter (usando somente *slices* I e P) e codificação de entropia através do CAVLC. O perfil *Main* é focado na transmissão de televisão e no armazenamento de vídeo. Este perfil inclui suporte para vídeo entrelaçado, suporte à codificação intra e inter, *slices* do tipo B com predição ponderada e suporte à codificação de entropia através do CABAC. O perfil *Extended* é mais

voltado para aplicações em streaming de vídeo e não suporta vídeo entrelaçado ou o CABAC, mas agrega modos para habilitar uma troca eficiente entre *bitstream*s codificados (através de *slices* do tipo SP e SI) e melhora a resiliência a erros (através do particionamento de dados).

Para os três perfis definidos na primeira versão do padrão, a relação entre os elementos de cores é fixa. Esta relação é de 4:2:0 para os elementos Y, Cb e Cr. Isso significa que os elementos de crominância Cb e Cr possuem metade da resolução horizontal e vertical dos elementos de luminância. Isso implica em uma subamostragem dos elementos de crominância já no vídeo original. Esta subamostragem contribui na redução da redundância psicovisual, como foi explicado na seção 2.2 do texto.

Para vídeos que utilização a relação 4:2:0, entre os elementos de luminosidade e de cor, um macrobloco é formado por uma matriz de 16x16 amostras de luminância e por duas matrizes 8x8 amostras de crominância, uma para Cb e outra para Cr. Todas as amostras, tanto de luminância quanto de crominância, utilizam 8 bits de resolução.

Os três perfis inicialmente propostos pelo padrão H.264/AVC foram focados em vídeos de entretenimento ou de qualidade. Mas estes perfis não incluíram suporte para vídeos com resoluções mais elevadas, como as necessárias em ambientes profissionais. Para responder às exigências deste tipo de aplicação, uma continuação do projeto JVT foi realizada visando expandir as capacidades do padrão original. Estas extensões foram chamadas de extensões para alcance de fidelidade (*fidelity range extensions* - FRExt). O FRExt produziu um grupo de quatro novos perfis denominados coletivamente de perfis *High* (SULLIVAN, 2004).

A relação entre as ferramentas que compõe cada perfil do padrão é apresentada na Figura 3.2. Além da organização em perfis o padrão H.264/AVC também define 16 níveis baseados nas combinações entre resolução e taxa de amostragem. Com estas informações é possível deduzir o número máximo de quadros de referência e a máxima taxa de bits que podem ser utilizados (SULLIVAN, 2004).

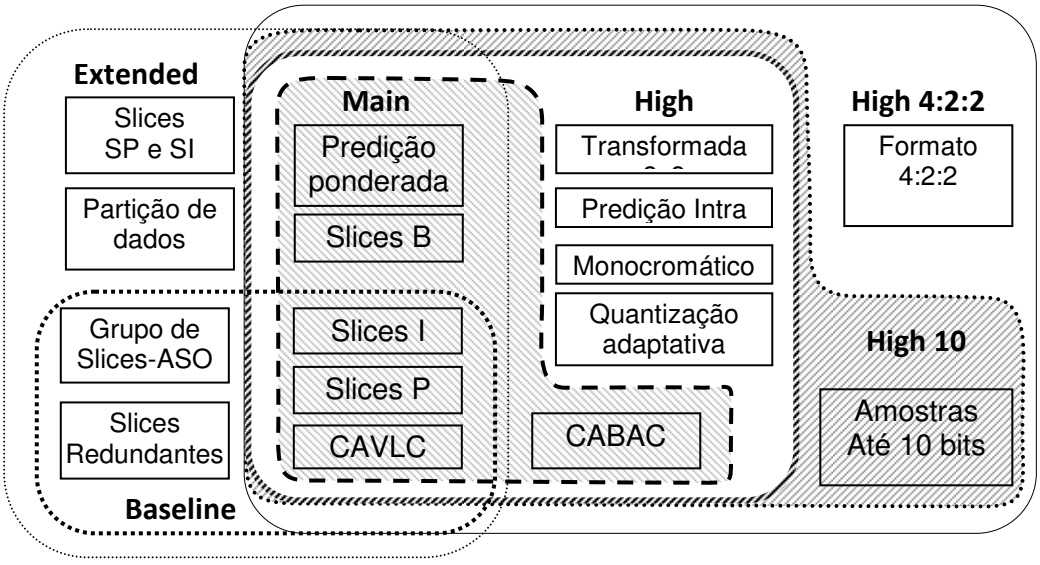


Figura 3.2: Relação entre ferramentas e os diferentes perfis definidos pelo padrão H.264/AVC.

3.3 Núcleo do padrão H.264/AVC

A Figura 3.3 apresenta um diagrama de bloco com as principais ferramentas que compõem um codificador segundo o padrão H.264/AVC. Embora possam existir algumas variações em relação ao perfil o conjunto básico de ferramentas é mantido. O fluxo de codificação e de codificação é realizado em nível de macroblocos. As setas entre os retângulos da Figura 3.3 indicam o sentido do fluxo de dados entre as ferramentas. Um decodificador para o padrão H.264/AVC emprega, basicamente, o mesmo conjunto de ferramentas, porém organizada em ordem inversa de processamento em relação ao processo de codificação.

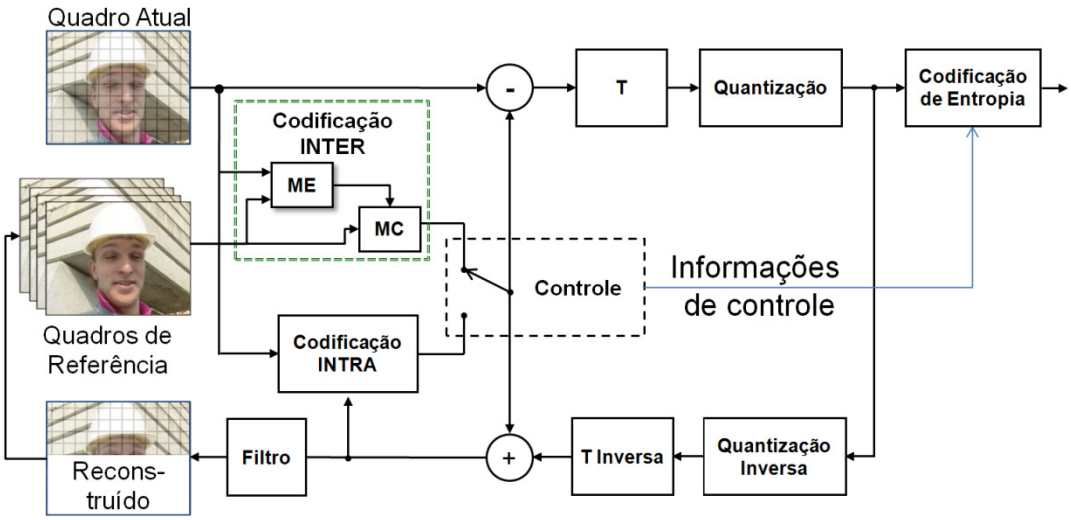


Figura 3.3: Diagrama sistêmico com as principais ferramentas presentes em um codificador para o padrão H.264/AVC.

Na Figura 3.3 é possível observar que o quadro atual (cena captura e representada digitalmente) é primeiramente analisado pelas ferramentas de predição de movimento. A padrão H.264/AVC adota dois tipos de ferramentas para predição de movimento, uma baseada na predição espacial (dentro do quadro) e outra na predição temporal (entre quadros). Na Figura 3.3 essas ferramentas são representadas pelos retângulos "Codificação INTRA" e "Codificação INTER", respectivamente. Depois, o resultado da etapa de predição é subtraído do quadro atual gerando um conjunto de valores residuais que são processados pelo bloco de transformadas (representado pelo retângulo "T"). A seguir, os resíduos transformados sofrem um processo de quantização. Na sequência, os resíduos quantizados são processados pela codificação de entropia, que além destes dados também processa as informações de controle produzidas pelas demais ferramentas empregadas no processo. Por fim, a saída do bloco de codificação de entropia resulta em um *bitstream* que representa o sinal de vídeo digital codificado.

Na Figura 3.3 existe uma seta após o bloco de quantização que "liga" a saída deste com um bloco de quantização inversa. Ocorre que, em paralelo com a execução da codificação de entropia, os resíduos quantizados são enviados para os blocos de quantização e transformada inversa com o objetivo de restaurar o valor dos resíduos gerados após a predição. Em seguida, os dados residuais restaurados são somados as dados preditos e enviados ao bloco de "Codificação INTRA" e ao filtro redutor de efeitos de bloco. Logo após processados pelo filtro redutor de efeitos de bloco os macroblocos são armazenados no *buffer* de quadro. Depois, quando o quadro atual estiver complemente codificado, a versão reconstruída do quadro atual será armazenada

na memória de quadros para ser utilizado como quadro de referência durante a codificação dos quadros futuros.

Uma característica interessante do padrão H.264/AVC é que apenas o decodificador é normatizado. Assim, vários graus de liberdade podem ser explorados no desenvolvimento do codificador. Entretanto o *bitstream* gerado deve ser compatível com as exigências do padrão, porém a forma como este *bitstream* deve ser gerado não é normatizada. As alternativas que podem ser aplicadas ao desenvolvimento do codificador vão desde a inserção de algoritmos mais eficientes ou mais fáceis de serem executados em *hardware* até a simples eliminação de algumas possibilidades de codificação previstas pelo padrão (AGOSTINI, 2007).

As ferramentas que compõem o padrão H.264/AVC, ilustradas na Figura 3.3, compreendem um extenso conjunto de algoritmos e técnicas de elevado grau de complexidade. As técnicas e algoritmos podem ser agrupados de acordo com sua finalidade e para esses grupos será utilizado o termo "módulo". A explicação detalhada de cada um destas ferramentas foge ao escopo deste trabalho, porém podem ser encontradas em vários trabalhos disponíveis na literatura (AGOSTINI, 2007; PURI, 2005; MARPE, SCHWARZ, WIEGAND, 2003; ITU-T, 2003). Contudo, um breve esclarecimento sobre as funcionalidades dos principais módulos será apresentado a fim de apresentarmos a informações produzidas por essas ferramentas e que serão tratadas pela codificação de entropia, mais especificamente pelo CABAC.

3.3.1 O Módulo de Estimação de Movimento (ME)

A predição inter-quadros trata as redundâncias temporais. Para realizar esta tarefa é necessário comparar cada macrobloco do quadro atual com todos os macroblocos de cada quadro de referência, ou seja, quadros que serão exibidos antes ou depois do quadro atual. Na saída desse processo temos, de forma simplificada, um vetor de movimento que aponta para a posição que melhor combine o quadro atual dentro de um quadro de referência.

O módulo da ME apresenta grande complexidade computacional (PURI, 2004). O custo computacional associado à ME é função das inovações adotadas pelo padrão H.264/AVC, as quais visam atingir elevadas taxas de compressão. Assim, os módulos de ME e sua contraparte, a MC, são responsáveis pelos maiores ganhos do padrão H.264/AVC em relação à taxa de compressão, quando comparados a padrão anteriores (WIEGAND, 2003; RICHARDSON, 2003).

A principal inovação do H.264/AVC no ponto de vista da ME está na possibilidade de utilização de tamanhos de blocos variáveis para realizar a estimação de movimento. Ao invés de usar um macrobloco completo na estimação de movimento, o padrão H.264/AVC permite o uso de partições de macrobloco e partições de submacroblocos, denominadas "blocos". Além disso, o padrão H.264/AVC oferece suporte a bi-predição (predição ponderada), ou seja, um macrobloco (ou partição do macrobloco) pode "apontar" para dois quadros de referência (um anterior e outro posterior) que serão utilizados para montar o valor do quadro reconstruído. Outra funcionalidade que merece destaque é a capacidade de realizar a predição em nível de *half-pixel* ou *quarter-pixel* através de um processo de interpolação entre as amostras.

As informações resultantes do processo de ME, considerando todas as funcionalidades suportadas pelo padrão H.264/AVC, podem ser organizadas em dois grupos. No primeiro temos o resíduo da predição, que é obtido a partir da subtração

pixel a pixel do bloco predito com o bloco original. No segundo temos as informações de controle que indicam quais funcionalidades da ME foram utilizadas para cada macrobloco ou partição dele. Essas informações são, basicamente, o vetor de movimento diferencial, os índices para os quadros de referências para as listas 0 e 1 e a forma de segmentação dos macroblocos e suas partições. Ambos os grupos de informações serão, posteriormente processados pela codificação de entropia. Maiores detalhes sobre a estimação de movimento não serão abordados neste texto, podendo ser encontrados em Agostini (2007).

3.3.2 O Módulo da Compensação de Movimento (MC)

A etapa de compensação de movimento deve adaptar-se às definições da estimação de movimento. A ME localiza o melhor casamento dentre os quadros de referência e gera um vetor de movimento. É função da compensação de movimento, a partir do vetor de movimento gerado na ME, localizar os blocos de melhor casamento na memória de quadros anteriormente codificados (quadros passados e/ou futuros) e montar o quadro predito. Este quadro será subtraído do quadro atual para gerar o quadro de resíduos que passará pela transformada (AGOSTINI, 2007).

A compensação de movimento é realizada tanto no codificador quanto no decodificador. A MC no codificador pode ser simplificada, tendo em vista as possíveis simplificações que podem ser realizadas na ME no codificador, conforme foi apresentado na seção anterior. Por outro lado, no decodificador, a compensação de movimento deve ser completa, isto é, deve estar apta a operar sobre todas as funcionalidades do padrão (AGOSTINI, 2007).

A compensação de movimento deve estar apta a atender as exigências da estimação de movimento. Deste modo, a MC deve:

1. Tratar múltiplos tamanhos de partições de macroblocos;
2. Utilizar múltiplos quadros de referência anteriores e posteriores;
3. Interpretar corretamente os vetores construídos com base na predição de vetores;
4. Tratar vetores que apontam para fora da borda do quadro;
5. Reconstruir os macroblocos considerando uma precisão de ¼ de *pixel* para os vetores de movimento;
6. Reconstruir os macroblocos que utilizam as predições bi-preditiva, ponderada e direta para *slices* do tipo B;
7. Reconstruir corretamente os macroblocos do tipo skip para *slices* tipos P e B.

3.3.3 O Módulo da Predição Intraquadros

O módulo da predição intraquadros adotada pelo padrão H.264/AVC é responsável por realizar a predição de movimento dentro dos quadros. O tipo I é atribuído aos macroblocos que utilização este tipo predição. Esta predição é baseada nos valores previamente codificados do quadro atual, considerando os *pixels* acima e à esquerda de um determinado bloco.

Existem vários modos diferenciados de aplicação da predição intra, sendo que o tratamento da componente de luminância (Y) pode ser diferente do tratamento para as

componentes de crominância (Cb e Cr). A predição intra para amostras de luminância pode ser utilizada sobre blocos 4x4 ou 16x16. Existem nove diferentes modos de predição intra para blocos 4x4 e quatro modos para a predição sobre blocos 16x16 (RICHARDSON, 2003). A predição da crominância é realizada diretamente sobre blocos 8x8 e utiliza quatro modos distintos de predição, mas os dois componentes de crominância utilizam sempre o mesmo modo. Os modos de predição para crominância são muito similares aos modos de predição de luminância para blocos 16x16 que foram apresentados na Figura 3.1, exceto pela numeração dos modos. (RICHARDSON, 2003).

Os diferentes modos de predição intra para luminância e crominância possibilitam a geração de uma predição para macroblocos do tipo I que conduz a uma codificação eficiente para este tipo de macrobloco. As informações geradas pela predição de movimento intraquadros podem ser organizadas em dois grupos. No primeiro temos o resíduo da predição, que é obtido a partir da subtração pixel a pixel do bloco predito com o bloco original. No segundo temos as informações de controle que indicam quais os modos de predição adotados.

3.3.4 O Módulo das Transformadas Diretas

As informações residuais, ou seja, o sinal de erro produzido pelas ferramentas de predição de movimento é tratado pelo módulo de transformadas diretas (T), representado na Figura 3.3 pelo retângulo com a letra "T". O módulo T aplica um conjunto de operações sobre o sinal de erro que transforma esse sinal para sua representação equivalente no domínio das frequências, a fim de facilitar a aplicação de um processo de quantização que descarta parte das informações de menor relevância para o sistema visual humano.

O padrão H.264/AVC adota dois tipos de transformadas: I - Transformada Discreta do Cosseno – DCT (na sigla do inglês: *Discrete Cosine Transform*); e II – Hadamard. As entradas para o módulo T são blocos 4x4 (amostras de luminância) com os resíduos gerados pela etapa de predição associados com os dois blocos 2x2 para as informações de crominância. Para aplicação das transformas a informação residual precisa ser segmentada de acordo com os níveis DC e AC. A Figura 3.4 apresenta como as informações residuais são agrupadas com base nos seus níveis e componentes de cor.

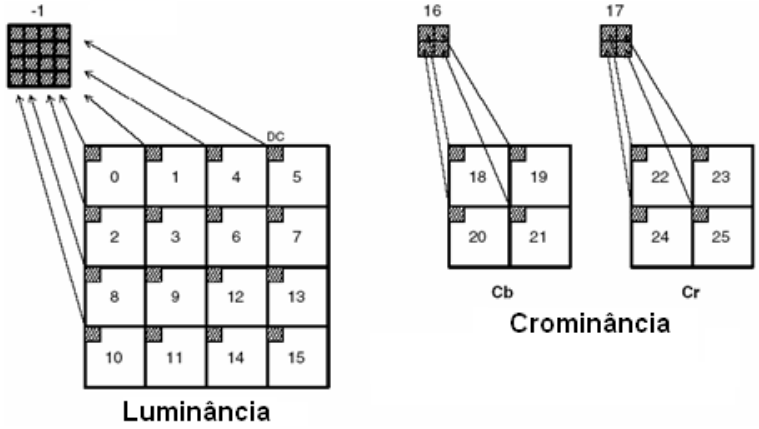


Figura 3.4: Blocos de coeficientes com tamanho 4x4 e 2x2 para aplicação das transformadas do padrão H.264/AVC para diferentes componentes de cor.

A aplicação dos diferentes tipos de transformadas prevista pode variar de acordo com o tipo de predição de movimento escolhida (AGOSTINI, 2007). Contudo, o

importante do ponto de vista da codificação de entropia é que depois da aplicação das transformadas a informação de resíduo pode será segmentada em blocos de 4x4 ou 2x2, podendo ocorrer até 26 blocos de resíduos para cada macrobloco.

3.3.5 O Módulo de Quantização

O módulo de quantização (Q) realiza a quantização direta e a correção da escala do cálculo das transformadas, sendo representado na Figura 3.3 pelo retângulo com a letra "Q". As entradas do módulo Q são o parâmetro de quantização (QP) e os coeficientes processados e reordenados pelo módulo de transformadas (T).

Dependendo do modo de predição utilizado e se o elemento é de crominância ou luminância, os cálculos realizados pelo módulo Q são modificados. As operações realizadas são, genericamente, uma multiplicação da entrada por uma constante, uma soma do resultado com outra constante e um deslocamento no resultado da soma controlado por uma terceira constante. Estas constantes são influenciadas diretamente pelo QP que informa ao módulo Q qual é o passo de quantização (Qstep) que deve ser utilizado. Os valores do QP podem variar de 0 a 51 e para cada QP existe um Qstep (AGOSTINI, 2007).

O módulo Q é utilizado para estabelecer controlar a relação qualidade e taxa de bits. Aplica-se um fator de quantização aos coeficientes e, assim, varia-se a taxa de bits na saída do codificador. O custo de diminuir essa taxa de bits é o aumento na perda da qualidade da imagem. A quantização é a única ferramenta do H.264/AVC que introduz perdas na qualidade da imagem. Depois de quantizados, os coeficientes são enviados para a codificação de entropia em ordenação Zig-Zag, conforme Figura 3.5.

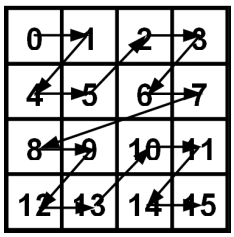


Figura 3.5: Ordem em que os coeficientes são entregues a codificação de entropia.

3.3.6 O Módulo do Filtro Redutor de Efeitos de Bloco

O módulo do filtro redutor de efeitos de bloco é representado na Figura 3.3 pelo retângulo intitulado “Filtro”. O objetivo do filtro é suavizar o efeito de blocos do quadro reconstruído antes que ele seja exibido ou utilizado para fazer a predição de um macrobloco que utilize a predição de movimento inter-quadros.

A inovação no caso do H.264/AVC em relação ao filtro redutor de efeitos de bloco é que ele é adaptativo, conseguindo “distinguir” entre uma aresta real, que não deve ser filtrada, e um artefato gerado por um elevado passo de quantização, que deve ser filtrado. Ou seja, o filtro redutor de blocagem deve ser aplicado nos casos em que há uma descontinuidade entre os blocos que são grandes o suficiente para ser percebido pelo sistema visual humano e é pequeno o suficiente para não caracterizar uma aresta real da imagem.

A operação de filtragem é realizada após a transformada inversa, e afeta até 3 amostras de cada lado na fronteira entre os blocos. A quantidade (“força”) de filtragem a ser realizada depende do valor do passo de quantização, do modo de decodificação

dos blocos vizinhos e do gradiente através da fronteira entre blocos. O filtro proporciona o aumento significativo da qualidade subjetiva do vídeo reconstruído, especialmente em baixas taxas.

3.3.7 O Módulo da Codificação de Entropia

No padrão H.264/AVC a codificação de entropia é o bloco responsável pelo tratamento das redundâncias existentes nas informações produzidas pelas outras ferramentas aplicadas no processo de codificação. A codificação de entropia explora relações características entre as palavras de código que deveriam formar o sinal de vídeo codificado.

No padrão H.264/AVC as ferramentas empregados geram um conjunto de informações de controle e dados codificados, as quais são denominadas de elementos sintáticos - SE (na sigla do inglês: *Syntax Element*). A norma de especificação do padrão H.264/AVC define três métodos básicos para codificação de entropia, sendo eles: Exp-Golomb, CAVLC e CABAC.

Dentro da codificação de entropia o método de codificação adaptativa por códigos de tamanhos variáveis - CAVLC (na sigla do inglês: *Context-Adaptive Variable Length Coding*) e o método da codificação aritmética binária adaptativa ao contexto - CABAC (na sigla do inglês: *Context-Adaptive Binary Arithmetic Coding*) são considerados os principais (RICHARDSON, 2003).

Considerando a estrutura de organização dos vídeos segundo o padrão H.264/AVC temos que nos níveis hierárquicos superiores (quadros, etc.) os SEs são codificados usando códigos binários fixos ou de comprimento variável. A partir do nível de *slices* ou abaixo (macroblocos, blocos, etc.), os SEs podem ser codificados com o CABAC ou com CAVLC.

A seleção do método de codificação de entropia a ser empregado depende do perfil de compatibilidade que se visa atingir. No perfil “baseline” e “extended” não há opção de escolha pelo CABAC, pois apenas o CAVLC está disponível. Entretanto, no perfil “main” os dois métodos estão disponíveis e a escolha por um deles fica a cargo do codificador (RICHARDSON, 2003). Para os casos onde o CAVLC é utilizado como método de codificação de entropia, cabe salientar que apenas os SEs de informação residual (coeficientes de transformadas) quantizados são codificados por esse método, enquanto os demais SEs e parâmetros de compressão são codificados através de Exp-Golomb (SALOMON, 2000).

A principal inovação introduzida na codificação de entropia pelo padrão H.264/AVC reside na técnica de codificação adaptativa baseada em contextos, a qual é aplicada tanto no CAVLC quanto no CABAC. Com esta técnica a maneira pelo qual os diferentes SEs são codificados depende dos SEs codificados em passos anteriores e da fase em que o algoritmo de codificação se encontra (RICHARDSON, 2003).

A possibilidade de adaptação dinâmica das probabilidades de ocorrência de um SE à fonte de entrada confere um grande diferencial às técnicas de codificação de entropia adotadas pelo padrão H.264/AVC em relação a outras abordagens discutidas na literatura. A Figura 3.6 ilustra a organização interna do bloco de codificação de entropia com cada um dos seus sub-blocos.

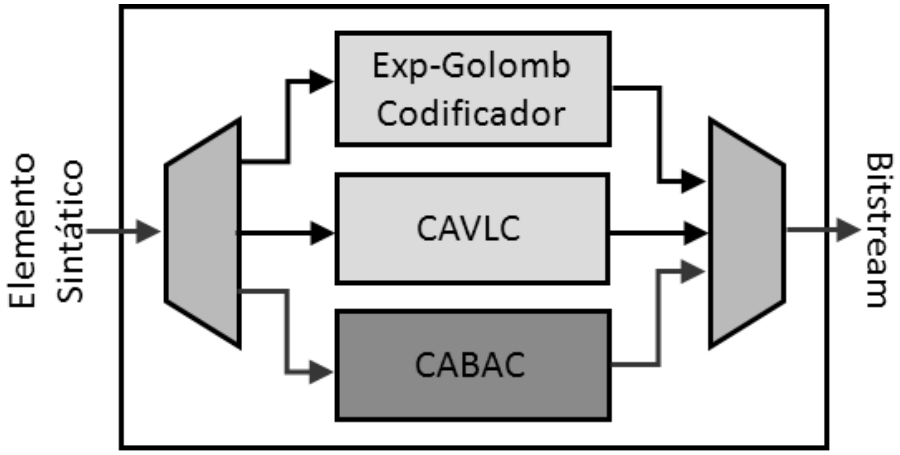


Figura 3.6: Organização interna do bloco de codificação de entropia conforme padrão H.264/AVC.

No próximo capítulo a codificação aritmética binária adaptativa ao contexto será apresentada e discutida em nível de detalhe adequado, conforme norma de especificação do padrão H.264/AVC, para que as diversas questões relacionadas às suas restrições e vantagens possam ser vislumbradas.

4 CODIFICAÇÃO ARITMÉTICA BINÁRIA ADAPTATIVA AO CONTEXTO

As aplicações para compressão de dados empregam técnicas distintas, com vários graus de complexidade, porém compartilham algumas etapas fundamentais. Os processos utilizados dependem do tipo de dados considerados e podem sofrer alterações na ordem de aplicação. Contudo, as etapas de processamento numérico, processamento lógico e processamento de entropia formam um conjunto clássico de etapas encontradas nestas aplicações.

No processamento numérico estão inseridas as técnicas de predição de movimento, transformações lineares e quantização. O processamento lógico visa transformar a forma como os dados estão organizados a fim de permitir a compressão e, para isso, aplicam-se técnicas como os códigos "*run-length*", as árvores de zeros, as rajadas de uns, o particionamento de informações e a construção de dicionário de dados. O processo de codificação de entropia visa explorar características estatísticas do comportamento do conjunto de informações produzidas pelas outras etapas do processo de compressão através da modelagem das probabilidades de ocorrência de cada conjunto de símbolos, seja de forma estática ou dinâmica, para possibilitar a representação destas informações de forma mais compacta (SAID, 2004).

O sinal de vídeo capturado por uma câmera a partir de uma cena natural apresenta um comportamento estatístico não estacionário. Além disso, as estatísticas geradas por esse sinal são fortemente dependentes do conteúdo do vídeo em questão. A codificação de vídeo, tipicamente, realiza o mapeamento do sinal de vídeo original em uma sequência de bits que representam códigos de comprimento variável para os diversos elementos sintáticos (SE) produzidos durante o processo de compressão. Esses códigos de comprimento variável exploram algumas das características não estacionárias, porém, certamente, não cobrem todas as possibilidades. Ademais, as relações estatísticas existentes entre as ocorrências dos diferentes SEs são, na sua maioria, ignoradas pelos esquemas de codificação de vídeo. Dessa forma, projetar um esquema de codificação de entropia - para um codificador de vídeo - que considere estas propriedades estatísticas oferece espaço para melhorias significativas na eficiência da codificação (MARPE, WIEGAND, 2003).

Este capítulo tem por objetivo apresentar em detalhes a técnica de codificação aritmética binária adaptativa ao contexto que é um dos métodos de codificação de entropia definidos pelo padrão H.264/AVC, pois compreender esta técnica é essencial para compreensão do trabalho desenvolvido. Na próxima seção vamos examinar os princípios da codificação aritmética. A seção 4.2 traz uma visão geral sobre o CABAC, discutindo sua estrutura de blocos e o funcionamento do algoritmo. Nas seções 4.3, 4.4 e 4.5 cada um dos principais blocos são detalhados.

4.1 Codificação Aritmética

Segundo Amir Said (2004) quando consideramos todos os métodos de codificação de entropia com suas possíveis aplicações para compressão de dados, o método de codificação aritmética merece destaque em termos de elegância, efetividade e versatilidade, pois apresenta maior eficiência em um grande número de circunstâncias e propósitos (SAID, 2004). Para ilustrar a afirmação anterior basta compararmos as taxas de compressão obtidas pelo padrão H.264/AVC para codificar uma mesma sequência de vídeo duas vezes, apenas trocando o método de codificação de entropia – de CAVLC para CABAC – onde pode ser observada uma redução média entre 9% a 15% do tamanho do *bitstream* gerado (MARPE, WIEGAND, 2003).

Dentre as características mais importantes do método de codificação aritmética podemos destacar:

- A compressão de cada símbolo é, provavelmente, ótima para fontes independentes e igualmente distribuídas (iid), com o comprimento da palavra de código próximo ao limite teórico conforme definido pela Equação 2.4;
- Efetiva para um grande conjunto de situações e taxas de compressão, pois a mesma implementação de codificação aritmética pode codificar de forma eficiente fontes de dados de origem diversas;
- Simplifica a modelagem automática de fontes complexas, com ganho próximo do ótimo ou com significantes melhoras na taxa de compressão para as fontes que não são independentes e igualmente distribuídas;
- As palavras de código geradas pela codificação aritmética podem ser mapeadas para qualquer sistema de representação.

Apesar destas características favoráveis a codificação aritmética enfrenta alguns problemas de ordem prática, dentre os quais se destacam:

- A elevada complexidade das operações aritméticas que dificultam a implementação, pois exigem muito processamento;
- As fontes de dados normalmente não são independentes e igualmente distribuídas;
- A sequência dos símbolos a serem codificados não é independente entre si;
- A forte dependência de dados entre as etapas do processo de codificação aritmética transformam esse método em um grande gargalo para os sistemas de compressão.

Para compreender as implicações das características supracitadas será necessário apresentar os princípios da codificação aritmética, a notação empregada pela literatura, a forma de representação das palavras de código e um exemplo do processo codificação e decodificação de uma mensagem através da codificação aritmética. Esses tópicos são abordados nas próximas seções deste texto.

4.1.1 Princípios da Codificação Aritmética

Na codificação aritmética a mensagem é representada por um intervalo de números reais [0 e 1), o qual é formado a partir de uma subdivisão recursiva do intervalo de

valores prováveis a partir da sequência de símbolos codificados, baseando-se nas probabilidades de ocorrência de cada símbolo dentro do alfabeto empregado. Essa probabilidade de ocorrência dos símbolos deve ser conhecida a priori ou obtida através da relação entre a frequência de cada símbolo e o número total de símbolos produzidos pela fonte. Conforme a mensagem torna-se longa, o intervalo necessário para representá-la torna-se menor e o número de bits necessário para especificar o intervalo cresce. Símbolos sucessivos na mensagem reduzem o tamanho do intervalo de acordo com o modelo de probabilidades de cada símbolo. O símbolo mais provável acarreta uma redução do intervalo e, portanto, acrescenta menor quantidade de bits para a mensagem em relação ao símbolo menos provável (PENNEBAKER, et al., 1988).

Antes que qualquer símbolo seja transmitido o intervalo da mensagem é definido entre [0, 1), denotando um intervalo meio aberto com $0 \leq x < 1$. Conforme cada símbolo é processado o intervalo é reduzido para a porção do intervalo alocada pelo símbolo em questão, de acordo com seu modelo de ocorrência probabilística. Por exemplo, supondo que um alfabeto α, que é composto pelos símbolos {a, e, i, o, u, !}, possua um modelo de probabilidades fixo, conforme apresentado na Tabela 4.1.

Tabela 4.1: Exemplo de um modelo fixo para alfabeto {a, e, i, o, u, !}

Símbolo	Probabilidade	Intervalo
A	0.2	[0.0, 0.2)
E	0.3	[0.2, 0.5)
I	0.1	[0.5, 0.6)
O	0.2	[0.6, 0.8)
U	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

Para exemplificar o processo de codificação aritmética vamos definir que a mensagem a ser transmitida deva ser "eaii!". Inicialmente ambos, codificador e decodificador, possuem o mesmo intervalo, definido para [0, 1). Ao receber o primeiro símbolo, “e”, o codificador reduz o intervalo para [0.2, 0.5) e realoca os símbolos do alfabeto para esse novo intervalo com base na tabela de probabilidades. O segundo símbolo, “a”, ao ser codificado vai reduzir esse novo intervalo para 1/5 do seu tamanho uma vez que o intervalo destinado a esse símbolo é de [0.0, 0.2), ou seja, 20% do intervalo total. Assim, o novo intervalo produzido após a codificação do segundo símbolo será [0.2, 0.26) visto que o intervalo anterior possuía 0.3 unidades de comprimento e 1/5 deste tamanho corresponde a 0.06 unidades. O próximo símbolo, “i”, é alocado para o intervalo [0.5, 0.6), ou seja, fica situado a partir da metade do intervalo e ocupa 1/10 deste. No caso em questão ele será aplicado sobre o e quando aplicado sobre o intervalo [0.2, 0.26) resultando em um intervalo ainda menor que é [0.23, 0.236). Seguindo esse processo para todos os símbolos a mensagem será codificada conforme ilustrado pela Tabela 4.2 (WITTEN, NEAL, CLEARY, 1987).

A Figura 4.1 é uma representação gráfica para o processo de compressão apresentado na Tabela 4.2. As barras verticais representam o comprimento total do intervalo disponível para cada etapa do processo de codificação. Cada barra vertical é subdividida em retângulos, os quais contêm em seu interior uma determinada letra, que representam a probabilidade daquele símbolo ocorrer conforme estipulado pelo modelo apresentado na Tabela 4.1. Para cada símbolo da mensagem uma etapa do processo de

codificação é realizada fazendo com que o intervalo inicial seja escalado para a nova faixa de valor definida pelo modelo de probabilidades de ocorrência de cada símbolo.

Tabela 4.2: Exemplo de subdivisão recursiva do intervalo durante o processo de codificação de uma mensagem “eaii!”.

Símbolo	Intervalo
Inicialmente	[0.0, 1.0)
Depois do “e”	[0.2, 0.5)
“a”	[0.2, 0.26)
“i”	[0.23, 0.236)
“i”	[0.233, 0.2336)
“!”	[0.23354, 0.2336)

Na Figura 4.1 a primeira barra vertical, considerando as barras da esquerda para direita, ilustra a divisão do intervalo inicial [0.0, 1.0) em seis faixas de valores de acordo com as probabilidades de ocorrência de cada símbolo. Após a codificação do primeiro símbolo, no caso “e”, o modelo de distribuição de probabilidade é escalado dentro do intervalo correspondente ao símbolo “e” que é [0.2, 0.5), ou seja, com comprimento total de 0.3 unidades. Esse processo de subdivisão recursiva do intervalo se repete até que o último símbolo da mensagem seja codificado.

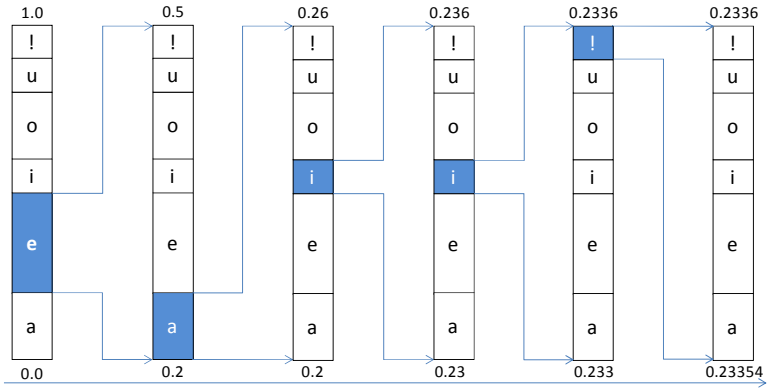


Figura 4.1: Representação do processo de codificação aritmética para a mensagem “eaii!”.

Quando o decodificador receber o intervalo final da mensagem codificada, que é [0.23354, 0.2336), ele poderá recuperar o primeiro símbolo codificado através de uma consulta ao modelo de probabilidade. Na realidade não é necessário que o decodificador conheça o intervalo final inteiro produzido pelo codificador. Basta um simples número dentro da faixa de valores do intervalo final para que o decodificador consiga executar a operação reversa, por exemplo, qualquer dos números {0.23355, 0.23357, 0.23358 e 0.23359} seria suficiente. Entretanto, o decodificador deve enfrentar problemas para detectar o final da mensagem, não sabendo quando parar o processo de decodificação, a final um único número como 0.0 poderia representar a codificação de qualquer uma das mensagens {"a", "aa", "aaa", "aaaa"}. Para resolver esse problema de ambiguidade deve-se assegurar que cada mensagem seja encerrada com um símbolo especial EOF conhecido por ambos, codificador e decodificador. No caso do alfabeto α esse símbolo é o “!”, que quando encontrado pelo decodificador leva ao término do processo (WITTEN, NEAL, CLEARY, 1987).

4.1.2 Notação

Dado que Ω é uma fonte de dados que gera símbolos "S_k", codificados como números inteiros, a partir de um alfabeto α com símbolos definidos em um conjunto como {0, 1, ..., M-1}, onde M denota o número de símbolos definidos pelo alfabeto α, e dado S = {s_1, s_2, ..., S_N} como sendo uma sequência de N valores aleatórios produzidos a partir da fonte Ω [1, 4, 5, 21, 55, 56]. Além disso, assume-se que os símbolos da fonte de dados são independentes e igualmente distribuídos com as probabilidades dadas de acordo com a Equação 4.1 (WITTEN, NEAL, CLEARY, 1987).

$$p(m) = Prob(sk = m), \ \ onde: m = 0, 1, 2, \dots, M-1 \ e \ k = 1, 2, \dots, N \qquad (4.1)$$

Assume-se também que todos os símbolos do alfabeto α possuam $p(m) \neq 0$, onde p(m) significa a probabilidade de cada um dos símbolos e que a distribuição acumulativa, dada por $c(m)$, é definida pela Equação 4.2.

$$c(m) = \sum_{s=0}^{m-1} p(s), m = 1, 2, \dots, M \qquad (4.2)$$

Deve considerar-se também que $c(0) \equiv 0$ e $c(M) \equiv 1$, e

$$p(m) = c(m+1) - \ c(m) \qquad (4.3)$$

As probabilidades de ocorrência de cada símbolo e a distribuição acumulativa destas probabilidades são armazenadas em dois vetores, contendo todos os símbolos produzidos por p(m) e c(m) sendo, respectivamente, representados por:

$$\boldsymbol{p} = [p(0)p(1) \dots p(M-1)]$$

$$\boldsymbol{d} = [\mathrm{c}(0)c(1) \dots c(M-1)c(M)]$$

Considerando todos esses aspectos um método de codificação ótimo para cada símbolo *s* produzido pela fonte Ω deve gerar um código com um número médio de bits definido de acordo com a Equação 4.4.

$$B(s) = \ -\log_2 p(s) \ \ bits. \qquad (4.4)$$

4.1.3 Palavras de Código

Uma grande diferença entre a codificação aritmética e outros métodos de codificação de entropia é que neste não é possível conhecer o relacionamento exato entre o símbolo codificado e os bits realmente produzidos pela saída do codificador. Cada símbolo, codificado em um determinado momento, recebe um número real de bits de acordo com a Equação 4.4. Para compreender esse processo é necessário ter em mente que cada símbolo codificado gera uma palavra de código que representa o valor do seu intervalo, ou seja, as mensagens codificadas são mapeadas para um número real dentro do intervalo [0, 1) (WITTEN, NEAL, CLEARY, 1987).

Uma palavra de código υ representa uma sequência de dados codificada com um número de dígitos fracionários igual a sequência dos símbolos codificados. É possível converter uma sequência de dados codificados em palavras de código apenas adicionando "0." à frente da sequência codificada e interpretar o resultado como um número em um base-D, onde D é o número de símbolos no alfabeto da sequência codificada. Por exemplo, se um método de codificação gera a sequência definida pelo

conjunto $\boldsymbol{d} = [0011000101100]$ então a palavra de código (o subscrito na palavra de código υ significa notação na base 2) pode ser obtida conforme:

$$\text{sequência codificada d} = [0011000101100]$$

$$palavra\ de\ código\ \upsilon = [0.0011000101100_2] = 0.19287109375$$

Através desta construção cria-se uma conveniente forma de mapeamento entre infinitas sequências de símbolos definidos em um alfabeto com tamanho D e um número real dentro do intervalo [0, 1), onde qualquer sequência de dados pode ser representada por um número real e vice-versa. A representação da palavra de código pode ser usada por qualquer sistema de codificação, pois provê um caminho universal para representar grande quantidade de informação independentemente do conjunto de símbolos usados na codificação (binário, ternário, octal, decimal, hexadecimal, etc).

4.1.4 Fluxo de Codificação e Decodificação

A codificação aritmética consiste, fundamentalmente, da criação de uma sequência aninhada de intervalos na forma $\Phi_{\mathrm{k}}(\mathrm{S}) = [\alpha_k, \beta_k), k = 0, 1, \dots, N$ onde S é a fonte de dados e $\alpha_k\ e\ \beta_k$ são números reais tal que $0 \leq \alpha_k \leq \alpha_{k+1}$ e $\beta_{k+1} \leq \beta_k \leq 1$. Em uma descrição simplificada da codificação aritmética os intervalos são representados na forma $[b, l)$, onde b representa a base ou ponto inicial do intervalo e l representa o comprimento deste intervalo. O relacionamento entre a forma tradicional e a forma simplificada de representação dos intervalos pode ser observada através da Equação 4.5 (MOFFAT, NEAL, WITTEN, 1998).

$$[b, l) = [\alpha, \beta)\ se\ b = \alpha\ e\ l = \beta - \alpha \tag{4.5}$$

Os intervalos empregados durante o processo de codificação, de acordo com a notação simplificada, são definidos pelas equações recursivas apresentadas a seguir:

$$\Phi_0(\mathrm{S}) = [b_0, l_0) = [0, 1) \tag{4.6}$$

$$\Phi_{\mathrm{k}}(\mathrm{S}) = [b_k, l_k) = [b_{k-1} + c(s_k).\, l_{k-1},\ \ p(s_k).\, l_{k-1}),\ \ k = 1, 2, \dots, N \tag{4.7}$$

Para o processo de decodificação a ordem em que os símbolos são decodificados é determinada, unicamente, pela palavra de código produzida a partir da sequência comprimida. A sequência de dados decodificada é representada por:

$$\hat{S}(\upsilon) = \{\widehat{s_1}(\upsilon), \widehat{s_2}(\upsilon), \dots, \hat{\mathrm{S}}_{\mathrm{N}}(\upsilon)\} \tag{4.8}$$

Para recuperar a sequência de dados originalmente codificados o decodificador aplica um conjunto de equações recursivas. Formalmente, para encontrar a solução numérica, é necessário definir uma sequência de palavras de código normalizadas $\{\tilde{v}_1, \tilde{v}_2,\ \dots, \tilde{v}_N\}$. Iniciando com $\tilde{v}_1 = \upsilon$ busca-se, sequencialmente, $\widehat{\mathrm{s_k}}$ a partir de $\tilde{v}_k$ para calcular $\tilde{v}_{k+1}$As equações recursivas são:

$$\tilde{\upsilon}_1 = \upsilon, \tag{4.9}$$

$$\hat{\mathrm{s}}_{\mathrm{k}}(\upsilon) = \{s : c(s)\ \leq \tilde{\upsilon}_{\mathrm{k}} < c(s+1)\}\ \ k = 1, 2, \dots, N \tag{4.10}$$

$$\tilde{v}_{k+1} = \frac{\tilde{v}_k - \mathrm{c}(\hat{\mathrm{s}}_{\mathrm{k}}(\upsilon))}{\mathrm{p}(\hat{\mathrm{s}}_{\mathrm{k}}(\upsilon)}\ \ k = 1, 2, \dots, N-1 \tag{4.11}$$

Contudo, em aplicações práticas a codificação aritmética normalmente utiliza um recurso de precisão aritmética fixa, que possui equivalência matemática ao método tradicional, porém emprega outro conjunto de equações que são definidas a seguir:

$$\Phi_0(\hat{S}) = [b_0, l_0) = [0, 1) \tag{4.12}$$

$$\hat{s}_k(\upsilon) = \{s: c(s) \leq \frac{\upsilon - b_{k-1}}{l_{k-1}} < c(s+1)\} \quad k = 1, 2, \ldots, N \tag{4.13}$$

$$\Phi_k(\hat{S}) = [b_k, l_k) = [b_{k-1} + c(\hat{s}_k(\upsilon)).\, l_{k-1},\ \ p(\hat{s}_k(\upsilon)).\, l_{k-1}), \qquad k = 1, 2, \ldots, N \tag{4.14}$$

4.2 Visão Geral do CABAC

Na seção anterior os detalhes e embasamento matemático da codificação aritmética foram expostos. Nesta seção o foco será a apresentação da estrutura, organização e etapas do processo de codificação empregado pelo CABAC.

A codificação aritmética binária adaptativa ao contexto conforme definida pelo padrão H.264/AVC é um framework para codificação e decodificação de entropia que transforma o valor de um símbolo em uma palavra de código com tamanho variável e comprimento próximo ao limite teórico da entropia (MARPE, SCHWARZ, WIEGAND, 2003). Em essência ele é um codificador aritmético clássico, baseado no princípio da subdivisão recursiva do intervalo, combinado a um sofisticado esquema de mapeamento de valores decimais para cadeias de bits (denominado binarização) e um mecanismo de adaptação dinâmica do modelo probabilístico às tomadas de decisões realizadas durante o processo de codificação aritmética (denominado adaptação ao contexto), sendo estas duas grandes inovações introduzidas pelo padrão H.264/AVC. Além disso, cabe ressaltar que o CABAC só está disponível a partir do perfil *Main* do padrão H.264/AVC.

Cada subintervalo representa um único símbolo e o tamanho deste intervalo é proporcional à probabilidade de ocorrência deste símbolo. Entretanto, modelar as probabilidades de ocorrência de cada símbolo do alfabeto implica no acréscimo da complexidade computacional. Uma das formas de diminuir essa complexidade é através da utilização de um alfabeto de símbolos reduzido.

O padrão H.264/AVC oferece suporte ao CABAC apenas a partir do perfil *Main*. Neste perfil, todos os SEs sintáticos produzidos a partir do nível de macrobloco são codificados através do CABAC, enquanto os SEs acima deste nível são codificados através do Exp-Golomb. Os SEs são compostos por duas informações básicas: I - tipo; e II - valor. O tipo do SE é utilizado pelo CABAC para determinar a forma de binarização e para calcular os endereços dos contextos associados a partir do modelo probabilístico. O valor representa o subintervalo associado a informação original codificada.

A similaridade estrutural entre o CABAC (codificador) e o CABAD (decodificador) fica evidente ao observar o conjunto de blocos que os compõe. Exceto pelo bloco de mapeamento binário, que apresenta comportamento diferenciado, os demais são praticamente iguais nos dois processos, sofrendo apenas alteração em sua ordem de execução. O diagrama de blocos apresentado pela Figura 4.2 ilustra o relacionamento entre os módulos do codificador e do decodificador, bem como a organização interna de cada um deles. Na parte superior da Figura 4.2 encontram-se, delimitados por um retângulo, os blocos que compõem o codificador; para o qual se observa que, inicialmente, o valor do SE é passado ao módulo de binarização que fará o processo de

mapeamento do valor inteiro em uma cadeia de bits baseado em um alfabeto binário de acordo com um padrão preestabelecido e selecionado de acordo com o tipo do SE. Caso o SE em questão já esteja em um alfabeto binário (por exemplo, os *flags*) esse processo não é aplicado. Depois, o bloco modelador de contexto seleciona o contexto apropriado e envia-o para o bloco de codificação aritmética. Cada contexto recuperado traz informações sobre o elemento mais provável a ser produzido pela etapa de codificação aritmética binária e um índice que aponta para o valor de intervalo associado ao elemento menos provável. No bloco de codificação aritmética um bit de saída é produzido e os novos limites do intervalo são calculados. Finalmente, no bloco de estimativa de probabilidades o novo valor de contexto é gerado para manter atualizada a precisão da estimativa de ocorrência de cada símbolo.

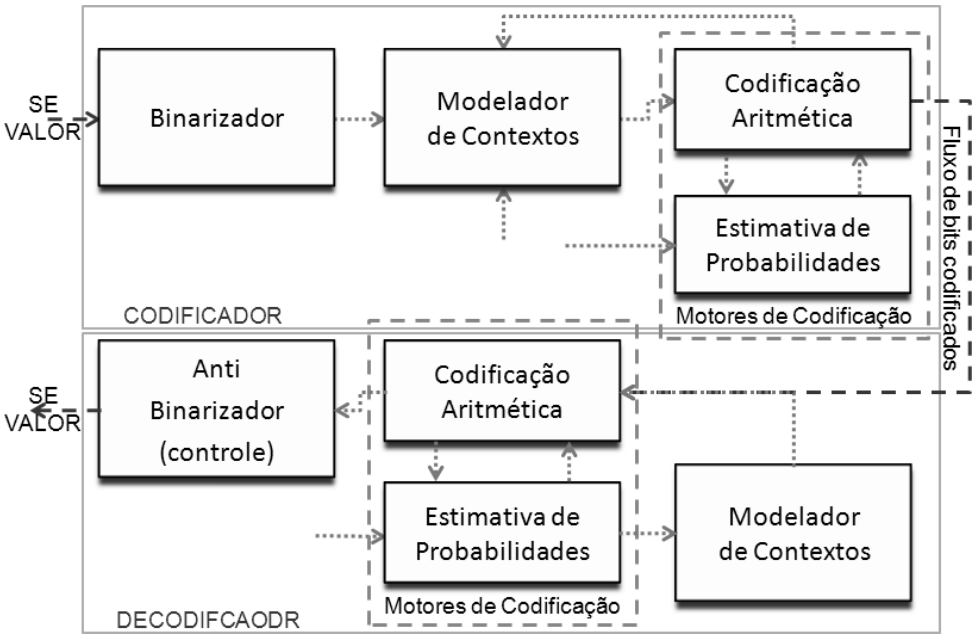


Figura 4.2: Diagrama de blocos com os principais módulos do CABAC e CABAD.

A saída do processo de codificação apresentado na Figura 4.2 é um fluxo de bits codificados que formam os dados de entrada para o decodificador, que pode ser observado na parte inferior da Figura 4.2. No decodificador o fluxo de execução inicia-se pelo bloco de codificação aritmética que visa encontrar qual o símbolo de entrada que satisfaz a Equação 4.13.

O diagrama de blocos apresentado pela Figura 4.2 pode passar a falsa impressão de simplicidade do processo de codificação e de decodificação, entretanto a norma de definição do padrão H.264/AVC dedica uma subseção inteira para cada um destes módulos, tal o nível de detalhes e implicações inerentes a cada um destes processos. Assim, as próximas subseções são destinadas a fornecer o nível de detalhes adequado para cada uma destas etapas.

4.2.1 Binarização

Como mencionado anteriormente, a etapa de binarização realiza um processo de mapeamento de valores não binários para uma cadeia de bits que visa minimizar a redundância no código. Além disso, o processo permite fácil acesso ao símbolo mais provável uma vez que esse fica localizado mais próximo da raiz da árvore de decisões binárias, facilitando a etapa de modelagem subsequente. Uma árvore de decisão de

códigos binários, na média, minimiza o número de símbolos binários a serem codificados e com isso reduz a sobrecarga computacional na etapa de codificação aritmética binária. Para o sucesso de aplicações de modelagem de contexto e codificação aritmética adaptativa em codificação de vídeo os seguintes requisitos devem ser satisfeitos:

a) Uma rápida e precisa estimativa das probabilidades condicionais deve ser obtida em um intervalo de tempo relativamente curto;
b) A complexidade computacional envolvida para executar cada operação elementar para recuperar a estimativa de probabilidade e o posterior cálculo aritmético deve ser minimizada para possibilitar uma vazão suficientemente alta entre os processos organizados de forma sequencial.

Para atender aos dois requisitos supracitados o CABAC introduz uma etapa de "pré-processamento" que reduz o tamanho do alfabeto de SE a serem codificados. A redução deste alfabeto é executada através de um esquema de binarização para cada SE que possuir um valor não binário, resultando em uma única palavra de código binário intermediária para um determinado SE, denominada de "*binstring"*. Esta técnica apresenta vantagem tanto do ponto de vista da modelagem quanto da implementação propriamente dita. Além disso, é importante notar que não há perdas em termos de precisão de modelagem, uma vez que as probabilidades individuais de cada símbolo (não binário) podem ser convertidas em probabilidades individuais de cada elemento que compõe o *binstring*, denominados *bins*. Para ilustrar essa afirmação vamos considerar um exemplo de binarização para o SE mb_type em um *slice* do tipo P/SP.

Na Figura 4.3 o nodo terminal da árvore binária corresponde aos símbolos de valor para um SE tal que a concatenação das decisões binárias que atravessam a árvore do nodo raiz até o nodo terminal representa o *binstring* para o valor daquele SE. Neste exemplo, o símbolo "3" corresponderia ao valor do SE do tipo mb_type, o qual identifica um macrobloco do tipo "P_8X8". Neste caso, o *binstring* corresponderia a sequência "001". Como consequência óbvia a probabilidade do símbolo $p("3")$ é igual produto das probabilidades $p^{(c0)}("0"), p^{(c1)}("0"), p^{(c2)}("1")$, onde $c0, c1\ e\ c2$ denotam o modelo binário com as probabilidades correspondente aos nós internos da árvore apresentada na Figura 4.3. Essa relação é verdadeira para qualquer símbolo representado por uma arvore binária, a qual pode ser deduzida pela aplicação iterativa do teorema das probabilidades totais (MARPE, SCHWARZ, WIEGAND, 2003).

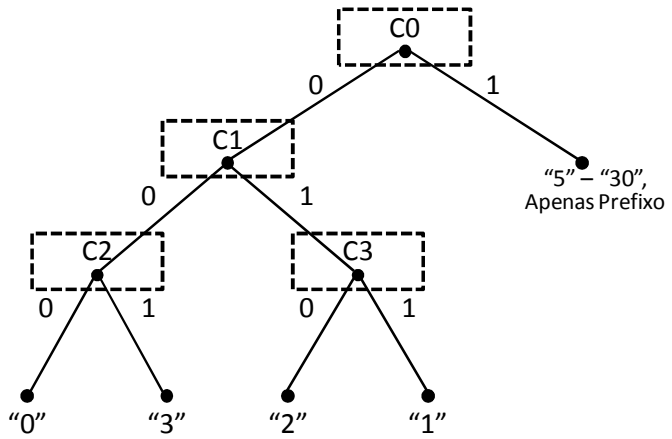


Figura 4.3: Exemplo de árvore de decisões binárias para representação do valor "3" para o SE mb_type.

Embora nesta fase não se perceba nenhum benefício da introdução da técnica de mapeamento do conjunto de símbolos originais para um alfabeto binário existe uma grande vantagem que é o fato de se utilizar um modo de codificação aritmética que opera sobre um alfabeto binário frente ao alfabeto original, dado que o processo de adaptação da codificação aritmética trata-se de uma operação computacionalmente complexa e que exige, ao menos, duas multiplicações para cada símbolo a ser codificado. Além disso, uma série de outras operações, de custo computacional elevado, precisa ser executada para manter as estimativas de probabilidade atualizadas. Em contrapartida, quando temos um alfabeto binário, é possível empregar outra estratégia de codificação aritmética, a qual é mais rápida e livre de multiplicações, sendo que esta foi desenvolvida especialmente para o CABAC. Considerando que a probabilidade dos símbolos com grande largura de *binstring* é tipicamente pequena, então o esforço computacional extra para codificar os N *bins* do *binstring* (em N passos), ao invés de codificar o símbolo do alfabeto original em um único passo em um codificador aritmético tradicional, é pouco significante e pode ser facilmente compensado por meio de um modo de codificação aritmética mais eficiente.

Finalmente, o processo de binarização permite aplicar a modelagem de contexto ao nível de subsímbolo e essa é considerada a vantagem mais significativa. Para *bins* específicos que representam as maiores frequências de observação, a modelagem condicional das probabilidades pode ser empregada, enquanto para outros que representam as menores frequências de observação, as probabilidades podem ser modeladas através de conjuntos, geralmente de ordem zero. O processo de modelagem de contexto tradicional no domínio da fonte leva a utilização de alfabetos grandes como, por exemplo, para modelar os vetores de movimento diferenciais ou o valor dos coeficientes de transformada. Nestas situações, a técnica de binarização provê mais flexibilidade ao projeto, possibilitando a utilização de modelos de probabilidades condicionais de ordem superior sem sofrer os efeitos da "diluição" de contexto. Estes efeitos são frequentemente observados nos casos em que um grande número de probabilidades condicionais tem de ser adaptativamente estimados em um pequeno intervalo de tempo, de tal forma que não existam amostras suficientes para atingir uma estimativa confiável para cada modelo.

Maiores detalhes sobre as implicações da utilização de um alfabeto binário ou alfabeto original sobre os processos de modelagem de contextos e ou a codificação aritmética fogem ao escopo deste trabalho e podem ser obtidas em Marpe, Schwarz e Wiegand (2003) e na norma de especificação do padrão H.264/AVC (ITU-T, 2003).

Quanto ao tamanho do *binstring* gerado para cada símbolo cabe salientar que ele é variável e depende do tipo de SE que está sendo processado e do estado atual do processo de codificação. O objetivo do mapeamento binário com tamanho de *bins* variável é propiciar que os símbolos com maior frequência de ocorrência sejam representados com o menor número de bits possível, possibilitando maior eficiência no processo de compressão do *bitstream*.

Para determinar a sequência de formação da cadeia de bits emprega-se, normalmente, uma árvore de Huffman. No entanto, o esquema de binarização aplicado ao CABAC opta pela utilização de poucas árvores de código básico, cuja estrutura permite um cálculo simples e direto de todas as palavras de código, sem a necessidade de armazenamento em nenhuma tabela. O esquema de binarização do CABAC é composto por métodos básicos: I - código unário; II - código unário truncado; III - código Exp-Golomb de ordem K; e IV - código de comprimento fixo. Além disso,

existem outros métodos de binarização que são baseados na concatenação de alguns desses quatro métodos básicos. Como uma exceção a esses métodos estruturados de binarização, existe outro método, o qual não é estruturado, baseado na escolha manual da árvore binária empregada e é especificamente destinado aos SEs de tipos de macroblocos e submacroblocos.

A seguir, os quatros métodos básicos de binarização serão apresentados em maiores detalhes.

4.2.1.1 Binarização para Códigos Unários (U) e Unários Truncados (TU)

O método unário converte o valor inteiro sem sinal de cada símbolo em uma cadeia de uns (“1”) com comprimento igual ao valor do símbolo, acrescido de um zero (“0”) no final. Por exemplo, se o valor do símbolo a ser binarizado for igual a 7 então a *string* de *bins* resultante do processo de binarização será “11111110”.

Para o método unário truncado o processo é exatamente igual ao método unário, desde que o valor do símbolo a ser codificado seja inferior a um limite definido por uma variável “$cMax$”. Para os casos do valor do símbolo ser maior que o limite definido, então o comprimento da cadeia de uns (“1”) deve ser limitado ao comprimento da variável “$cMax$” e o bit zero (“0”) não deve ser acrescido ao final da cadeia de bits. Por exemplo, se o valor do símbolo a ser binarizado for igual a 7 e a variável “$cMax$” for igual a 5 então a *string* de *bins* resultante do processo de binarização será “11111”.

4.2.1.2 Binarização para Código de Tamanho Fixo (FL)

A aplicação deste método considera a existência de um alfabeto finito de valores possíveis para o valor do SE a ser codificado. Dado "x", que denota o valor do SE a ser binarizado, tem-se que $0 \leq x < cMax$. Então, a palavra de código de tamanho fixo produzida para "x" será, simplesmente, a representação binária deste com um número fixo (mínimo) de bits. Tipicamente, o método de binarização FL é aplicado para SE com distribuição uniforme aproximada ou para SE onde cada bit do FL representa uma decisão de codificação específica como é o caso de parte do valor do SE relacionado ao padrão de codificação dos blocos de luminância. A definição do tamanho exato da sequência binária que deve ser gerada pode ser obtida pela fórmula apresentada na Equação 4.15.

$$\mathrm{FL} = ceil(\log_2(cMax) + 1) \tag{4.15}$$

Utilizando “cMax” como parâmetro, a expressão retorna o número de bits necessários para representar o valor do SE de forma binária. Os índices do *binstring* são ordenados do bit menos significativo para o mais significativo. Dessa forma, caso o valor retornado pela expressão não seja suficiente para representar o valor do SE, então os bits menos significativos serão mantidos.

4.2.1.3 Binarização Exp-Golomb Parametrizável (EGk)

Esta família de códigos parametrizados é derivada de códigos Golomb e tem sido comprovado como ótima para códigos livres de prefixos aplicados a fontes geometricamente distribuídas (MARPE, WIEGAND, 2003). Os códigos Exp-Golomb são construídos por uma concatenação de um prefixo e um sufixo que formam a palavra de código. O termo parametrizável é associado à variável k. O prefixo da palavra de

código gerado pelo EGk consiste em um código unário, conforme apresentado na seção 4.2.1.1, que corresponde ao valor da expressão apresentada pela Equação 4.16.

$$l(x) = [\log_2 \left(\frac{x}{2^k + 1}\right)] \quad (4.16)$$

O sufixo da palavra de código gerado pelo EGk é calculado como a representação binária da expressão $x + 2^k(1 - 2^{l(x)})$ usando apenas os $k + l(x)$ bits mais significantes. Consequentemente, o método de binarização EGk gera um *binstring* com comprimento definido por $k + \left(2\, l(x)\right) + 1$, que cresce geometricamente. A Tabela 4.3 apresenta um conjunto de valores sobre os quais é aplicado o método de binarização EGk, produzindo diferentes formatos de *binstring* e evidenciando o padrão de geração produzido para o prefixo e o sufixo da palavra de código.

Tabela 4.3: Exemplo de aplicação do método de binarização EGk sobre diferentes valores para ilustrar a forma de composição do *binstring*.

Intervalo de valores	Formato	Código	Binstring
0	1	0	1
1-2	0 1 X^0	1	0 1 0
...	…	2	0 1 1
3-6	00 1 X^1 X^0	3	00 1 00
...	...	4	00 1 01
...	...	5	00 1 10
...	...	6	00 1 11
7-14	000 1 X^2 X^1 X^0	7	000 1 000
...	...	…	…
15-30	0000 1 X^3 X^2 X^1 X^0	16	0000 1 0001
...	...	…	…
31-62	00000 1 X^4 X^3 X^2 X^1 X^0	32	00000 1 00001

4.2.1.4 Concatenação dos esquemas de binarização básicos

A partir dos quatro métodos básicos de binarização descritos acima, mais três métodos de binarização são derivados. O primeiro destes métodos é composto pela concatenação de prefixo, com 4 bits gerados pelo método FL e representando o padrão de codificação relacionado aos blocos de luminância, e o sufixo de 2 bits gerados pelo método TU e representando o padrão de codificação relacionados aos bloco de crominância.

O segundo e o terceiro esquemas de codificação restantes são derivados da combinação do método de binarização TU com o método EGk. Esses esquemas são referenciados como binarização unária com código Exp-Golomb de ordem Kth (UEGk). Eles são aplicados para vetores de movimento diferenciais (MVD) e para o valor absoluto dos coeficientes de transformada quantizados. O projeto da concatenação destes esquemas é motivado pela seguinte observação:

1. Os códigos unários são livres de prefixo, com baixo custo de implementação;
2. Permitem rápida adaptação das probabilidades de cada símbolo individual para a etapa subsequente, a modelagem de contexto, visto que a organização dos nodos na árvore correspondente é de tal forma que a distância entre os

nodos internos e a raiz aumenta conforme a probabilidade dos símbolos diminui.

Essas observações são precisas somente para valores absolutos e pequenos dos SEs MVD e valor absoluto dos coeficientes de transformada quantizados. Para grandes valores, a modelagem adaptativa não oferece ganho significante. Neste cenário, montar o *binstring* através da composição de um prefixo formado por uma árvore unária truncada e um sufixo formado por uma árvore de códigos Exp-Golomb oferece bom equilíbrio entre eficiência de codificação e simplicidade computacional, visto que as árvores de código Exp-Golomb são estáticas, não necessitando modelagem de contexto. Assim, para grandes valores a utilização de um sufixo EGk oferece ganhos de performance significativos ao CABAC, pois permite a utilização de um método especial para a codificação aritmética, denominado *bypass*, o qual não faz acesso aos modelos probabilísticos e é apropriado para ser empregado na codificação de símbolos que possuam uma distribuição probabilística uniforme.

Para o valor dos símbolos MVD, o esquema de binarização UEGk é construído da seguinte forma: assume-se que o valor de um componente do vetor de movimento é conhecido; para o prefixo do *binstring* deste símbolo, o método de binarização TU é empregado, com $S = 9$, para o qual considera-se o valor do MVD obtido através da função de mínimo min $(|mvd_{value}|, S)$; caso o valor do MVD seja igual a zero, a palavra de código gerada será o *binstring* com conteúdo igual a "0"; caso contrário, se a condição $|mvd_{value}| \geq 9$ for satisfeita, então o sufixo do *binstring* será gerado a partir do método de binarização EGk, com $k = 3$, aplicado sobre o valor resultante da operação $|mvd_{value}| - 9$ sendo que o sinal do valor original do MVD será adicionado ao final deste sufixo, seguindo a convenção de um bit com valor igual a "0" quando o sinal for positivo e um bit com valor igual "1" quando o sinal for negativo; para valores de MVD que satisfaçam a condição $0 < |mvd_{value}| < 9$ o sufixo do *binstring* consistirá apenas do bit de sinal.

Para o valor absoluto dos coeficientes de transformada, o esquema de binarização UEGk é construído da seguinte forma: para o prefixo do *binstring* deste símbolo, o método de binarização TU é empregado, com $S = 14$, enquanto para o sufixo do *binstring* EGk é utilizado, com $k = 3$; o processo de formação do prefixo e sufixo é similar ao processo aplicado aos componentes de vetor de movimento, com a diferença que nenhum bit de sinal é adicionado ao final do *binstring*. A Tabela 4.4 apresenta um conjunto de possíveis valores para os coeficientes de transformada, onde o prefixo e o sufixo do *binstring* gerado podem ser observados, exemplificando o processo.

Tabela 4.4: Exemplo de geração de *binstring* para o valor absoluto dos coeficientes de transformada quantizados – com valores definidos entre 1 e 20.

Nível absoluto	Binstring	
	Prefixo (TU)	Sufixo (EGk)
0	0	
1	10	
2	110	
3	1110	
4	11110	
5	111110	
...	...	
13	11111111111110	
14	111111111111110	
15	111111111111111	0
16	111111111111111	1 0 0
17	111111111111111	1 0 1
18	111111111111111	11 0 00
19	111111111111111	11 0 01
20	111111111111111	11 0 10
...	...	…

4.2.2 Modelagem de Contexto

Um modelo de contexto é um modelo probabilístico que representa as estatísticas de distribuição e/ou ocorrência de um determinado símbolo, com base na análise da frequência de ocorrência daquele símbolo para a fonte em questão, e da observação dos símbolos processados anteriormente durante um determinado processo de codificação. Uma das mais importantes propriedades da codificação aritmética é a possibilidade de uma clara interface de separação entre a modelagem das probabilidades e a codificação propriamente dita. Na etapa de modelagem, uma determinada distribuição probabilística é associada a cada símbolo para que a etapa subsequente (etapa de codificação) possa gerar a sequência de bits, que seja a representação codificada daquele símbolo, de acordo com o seu modelo de distribuição probabilístico para uma determinada fonte. Uma vez que o modelo determine como o código será gerado, então é de importância primordial que a etapa de modelagem seja projetada para explorar adequadamente e em profundidade as dependências estatísticas, mantendo-as atualizadas durante todo o processo de codificação. Entretanto, há um custo significativo envolvido para adaptar as estimativas de probabilidades condicionais de grande ordem.

Supondo um conjunto de símbolos passados definido como T, também denominado modelo de contexto, e dado um conjunto de contextos correlatos $\mathrm{C} = \{0, \ldots, \mathrm{C}-1\}$, onde os contextos são especificados através de uma função de modelagem $F\colon \mathrm{T} \to \mathrm{C}$ operando sobre o modelo T. Para cada símbolo x codificado, uma probabilidade condicional, dada por $p(x|F(z))$, é estimada a partir do chaveamento entre diferentes modelos de probabilidade, de acordo com os símbolos vizinhos já codificados e definidos como $z \in \mathrm{T}$. Depois de codificar x, utilizando a estimativa de probabilidade condicional dada por $p(X|F(z))$, então o modelo de probabilidades é atualizado com o valor do símbolo codificado. Dessa forma, as estimativas probabilísticas refletem, a cada símbolo codificado, as observações estatísticas da fonte. Dado que o número τ de diferentes probabilidades condicionais a serem estimadas para um alfabeto com o

tamanho m é expresso por $\tau = \mathrm{C}(m-1)$, torna-se intuitivamente claro que o custo do modelo (que representa o custo de "aprender" o modelo de distribuição) é proporcional a τ. Isso implica que, ao aumentar o número C de diferentes modelos de contexto, um refinamento excessivo das estimativas geradas por $p(x|F(z))$ pode levar a um resultado impreciso.

No CABAC, esse problema é resolvido pela imposição de duas severas restrições sobre a escolha dos modelos de contexto. Primeiro, um número limitado de modelos de contexto, consistindo apenas de alguns vizinhos do símbolo corrente a ser codificado, são empregados, reduzindo a quantidade de diferentes modelos de contexto efetivamente utilizados. Segundo, a modelagem de contexto é realizada para alguns *bins* de cada símbolo, conforme o modelo de binarização aplicado. Como resultado, o custo da etapa de modelagem de contextos é drasticamente reduzido, o que pode não significar, em termos de eficiência da codificação, a escolha ótima. Contudo, esse compromisso representa um balanceamento entre custo computacional e eficácia de codificação. No CABAC, a modelagem de contexto emprega quatro tipos distintos:

- O primeiro tipo envolve SEs que necessitam de apenas dois vizinhos para determinar suas condições de contexto. Nesse caso, a especificação do tipo de vizinho depende do SE atual. Geralmente, a seleção deste tipo de modelo de contexto para um determinado *bin* é baseada em uma função de modelagem sobre o valor dos *bins* dos SEs vizinhos, da esquerda e superior ao elemento atual.
- O segundo tipo de modelo de contexto é definido apenas para os SEs dos tipos mb_type e sub_mb_type. Neste tipo de contexto, o valor dos *bins* codificados anteriormente é utilizado para escolher um modelo para um determinado bin, indexado por binidx.

O terceiro e o quarto tipo de modelo de contexto são aplicados somente para os dados de resíduos, conforme apresentado na Tabela 4.5. Em contraste com os dois primeiros tipos, esses últimos dependem de categorias de contexto de diferentes tipos de blocos, conforme apresentado na Tabela 4.6. Além disso, o terceiro tipo não se baseia nos dados processados anteriormente e sim na posição do coeficiente que está sendo tratado no momento. Enquanto para o quarto tipo, as funções de modelagem envolvem a avaliação do número de níveis já codificados ou decodificados antes do nível atual.

Tabela 4.5: Associação do tipo inicial de modelo de contexto para SE de dados residuais de acordo com o tipo de categoria de contexto.

Tipo de Elemento Sintático	**Categorias de Contexto (ctx_cat)**				
	0	**1**	**2**	**3**	**4**
coded_block_flag	0	4	8	12	16
sig_coef_flag	0	15	29	44	47
map_coef_flag	0	15	29	44	47
coef_abs_level_minus1	0	10	20	30	39

Tabela 4.6: Tipos de blocos básicos com número de coeficientes associados e a categoria de contexto correspondente.

Tipos de Blocos	Número máximo de coeficientes	Categoria de Contexto (ctx_cat)
Luma DC Intra 16x16	16	Luma-Intra-16-DC: 0
Luma AC Intra 16x16	15	Luma-Intra-16-aC: 1
Luma Intra 4x4	16	Luma-4x4: 2
Luma Inter	16	
U-Chorma DC Intra	4	Chorma-DC: 3
V-Chorma DC Intra	4	
U-Chorma DC Inter	4	
V-Chorma DC Inter	4	
U-Chorma AC Intra	15	Chorma-AC: 4
V-Chorma AC Intra	15	
U-Chorma AC Inter	15	
V-Chorma AC Inter	15	

Além destes modelos de contexto baseados em probabilidades condicionais, há outras atribuições fixas de modelos de probabilidade para índices de *bin* para todos aqueles *bins* que têm que ser codificados na modalidade regular, aos quais nenhum modelo de contexto das categorias precedentes pode ser aplicado.

A modelagem de contexto utilizada no CABAC pode ser organizada de forma que cada modelo possa ser identificado por um único índice, denominado índice de contexto γ. A Tabela 4.7 contém uma visão geral dos principais SEs definidos pelo padrão H.264/AVC e o relacionamento entre esses SEs e os índices de contextos correspondentes. A numeração dos índices foi organizada de forma que os modelos relacionados aos SEs dos tipos *mb_type*, *sub_mb_type* e *mb_skip_flag* para os diferentes tipos de *slices* sejam distinguidos pelo próprio índice associado. A disposição e organização de contextos conforme apresentado na Tabela 4.6 não é o mais eficiente, considerando a sequência de acessos durante o processo de codificação e decodificação, porém, serve para fins ilustrativos da idéia básica.

Tabela 4.7: Associação do tipo de SE com o intervalo de contextos correspondente para cada tipo de *Slice*.

Tipo de Elemento Sintático	Tipo de *Slice*		
	I/SI	**P/SP**	**B**
mb_type	0/3-10	14-20	27-35
mb_skip_flag		11-13	24-26
sub_mb_type		21-23	36-39
mvd(horizontal)		30	40-46
mvd(vertical)			47-53
ref_idx			54-59
mb_qp_delta	60-63	60-63	60-63
intra_chroma_pred_mode	64-67	64-67	64-67
prev_intra4x4_pred_mode	68	68	68
rem_intra4x4_pred_mode	69	69	69
mb_field_decoding_flag	70-72	70-72	70-72
coded_block_pattern	73-84	73-84	73-84
coded_block_flag	85-104	85-104	85-104
significant_coeff_flag	105-165/277-337	105-165/277-337	105-165/277-337
last_significant_coeff_flag	166-226/338-398	166-226/338-398	166-226/338-398
coeff_abs_level_minus1	227-275	227-275	227-275
end_of_*slice*	276	276	276

O modelo de probabilidade relacionado a cada índice de contexto γ é determinado por um par de valores, sendo eles o estado do índice de probabilidade σ_γ (com 6 bits) e um valor binário, definido como $\overline{\omega}_\gamma$, para o símbolo mais provável (MPS). Assim, o par de valores $(\sigma_\gamma, \overline{\omega}_\gamma)$, definidos para $0 \leq \gamma \leq 459$, que compõem o modelo de contexto, pode ser representado por 7 bits.

Os índices de contexto no intervalo entre 0 e 72 são relacionados aos SEs para tipos de macroblocos, submacroblocos e modos de predição espacial e temporal para cada tipo de *slice* e informações de controle do macrobloco. Para esses SEs, o índice de contextos pode ser calculado como:

$$\gamma = \Gamma_S + \chi_S \tag{4.17}$$

onde Γ_S significa o deslocamento inicial do índice de contextos, os quais são a base do intervalo apresentado na Tabela 4.7, e o χ_S denota o incremento do índice de contexto para um determinado SE "S". Deve-se destacar que a variável χ_S depende apenas do índice do *bin* nos casos onde o assinalamento fixo de um modelo de contexto é selecionado. No entanto, ela pode especificar o primeiro ou segundo modelo de contexto quando for um dos outros casos de assinalamento, conforme descrito anteriormente.

Os índices de contextos para o intervalo de 73 até 459 são relacionados à codificação de dados residuais. Dois conjuntos de modelos de contexto são especificados para os SEs *significant_coeff_flag* e *last_significant_coeff_flag*, sendo que a codificação de ambos os SEs é condicionado à posição de processamento dentro do bloco de resíduos. Cabe salientar que nem todos os 460 contextos definidos são utilizados para a codificação de cada tipo de macrobloco, uma vez que a utilização destes depende do tipo de codificação: quadro ou campo.

O índice de contexto para o SE do tipo *coded_block_pattern* é especificado de acordo com a Equação 4.17, enquanto para todos os demais SEs referentes a dados residuais, o índice de contexto γ é dado por:

$$\gamma = \Gamma_S + \Delta_S(ctx_{cat}) + \chi_S \tag{4.18}$$

onde em adição ao deslocamento inicial do índice de contexto, é acrescido o deslocamento da categoria do contexto, $\Delta_S(ctx_{cat})$, para cada SE, conforme especificado na Tabela 4.5. A relação de todas as categorias de contexto pode ser observada na Tabela 4.6.

É importante observar que, independente do tipo de modelo de contexto empregado, as condições de seleção de contexto estão sempre associadas ao valor do SE de tal forma que o processo de codificação ou decodificação de entropia pode ser desassociado das demais operações de codificação ou decodificação do padrão H.264/AVC.

4.2.3 Codificação Aritmética Binária

Conforme mencionado anteriormente o processo de codificação aritmética é baseado no princípio da subdivisão recursiva do intervalo. Suponha que a estimativa probabilística $pLPS \in (0, 0.5]$ da ocorrência do símbolo menos provável (LPS) seja dada e que o intervalo associado é representado por seu limite inferior L e pela sua largura R; o intervalo dado é subdividido em dois subintervalos: um deles dado por:

$$R_{LPS} = \mathrm{R} * \rho_{\mathrm{LPS}} \tag{4.19}$$

o qual é associado com o LPS. O intervalo complementar associado ao símbolo mais provável (MPS), que tem probabilidade estimada em $1 - \rho_{\mathrm{LPS}}$, e é definido por:

$$R_{MPS} = \mathrm{R} - R_{LPS} \tag{4.20}$$

Dependendo da decisão binária, observada pela etapa de codificação, um dos dois subintervalos (MPS ou LPS) é selecionado para ser o novo intervalo. Um valor binário contido dentro deste novo intervalo representa a sequência de decisões binárias tomadas até o momento, considerando que o comprimento deste intervalo corresponde ao produto das probabilidades de cada um destes símbolos binários gerados. Assim, para garantir que o intervalo final possa ser representado sem ambiguidade é que a teoria de Shannon define que o limite inferior para a entropia da sequência é assintoticamente aproximada através de um limite inferior do intervalo final, considerando a mínima precisão de bits possível (SHANNON, 1948).

Em uma implementação prática para um codificador aritmético binário o principal gargalo em termos de vazão são as operações de multiplicação requeridas para executar a subdivisão do intervalo. Nesse sentido, encontram-se, na literatura, diversos trabalhos desenvolvidos visando acelerar a velocidade de cálculo através da introdução de algumas técnicas de aproximação do intervalo R ou das probabilidades $pLPS$ de forma a evitar a necessidade de multiplicações. Entre as propostas de codificadores de baixa complexidade merecem destaque os codificadores da família Q (PENNEBAKER, et al., 1988) e seus derivados QM e MQ (SLATTERY, MITCHELL, 1998), especialmente para o contexto da codificação de imagens. Apesar de o codificador MQ ser considerado o estado da arte em termos de velocidade de execução, observa-se que, para o cenário de compressão de vídeo, ele apresenta degradação da eficiência de codificação, fato que motivou a equipe de desenvolvimento do padrão H.264/AVC a desenvolver um novo

esquema de codificação aritmética binária livre de multiplicações, denominado codificador de módulo (ou codificador M), o qual atinge altas taxas de vazão sem causar degradação da eficiência.

A idéia básica deste codificador M, que não emprega multiplicações, é projetar o intervalo R adequado dentro dos limites corretos $[R_{min}, R_{max})$, bem como o intervalo de probabilidades associados ao LPS, através de um pequeno conjunto de valores representativos $Q = \{Q_0, \dots, Q_{K-1}\}$ e $P = \{p_0, \dots, p_{N-1}\}$, respectivamente. Dessa forma, a multiplicação do lado direito da Equação 4.19 pode ser aproximada através da utilização de uma tabela pré-calculada de $K\,x\,N$ produtos de valores $Q_\rho . p_\sigma$ para $\{0 \leq \rho \leq K-1\}$ e $\{0 \leq \sigma \leq N-1\}$. Para o módulo de codificação regular, empregado no padrão H.264/AVC, um bom compromisso entre o tamanho da tabela e a precisão dos intervalos pré-calculados foi obtido através de um conjunto $Q\ de\ K = 4$ de valores de intervalos quantizados com um conjunto $P\ de\ N = 64\ LPS$ valores de probabilidades relacionadas a esses intervalos.

4.3 Descrição Detalhada do CABAC

Esta seção apresenta informações detalhadas sobre as etapas de processamento para cada tipo de SE, desde a seleção do tipo de esquema de binarização a ser aplicado, passando pela seleção de modelo de contexto apropriado e pela geração do *binstring* correspondente na etapa de codificação aritmética binária. Para esse propósito os SEs foram divididos em duas categorias. A primeira categoria, a qual é descrita na subseção 4.3.1, contém os SEs para os tipos de macrobloco e submacroblocos além dos SEs associados às informações do tipo de predição, para ambos os modos (espacial e temporal), bem como informações de controle para os macroblocos e *slices*. A segunda categoria, a qual é descrita na subseção 4.3.2, apresenta os SEs relacionados com os dados residuais produzidos pelas ferramentas de transformadas e quantização bem como informações de controle dos tipos de blocos com coeficientes válidos e o mapa de significância.

Além disso, maiores explicações sobre os processos de geração das estimativas de probabilidades e de codificação aritmética binária baseada em tabelas são apresentadas nas subseções 4.3.3 e 4.3.4, respectivamente.

4.3.1 Codificação de Tipo de Macrobloco, Modo de predição e Informações de Controle

Considerando o *bitstring* final produzido pelo padrão H.264/AVC, empacotado dentro da camada VCL da NAL, encontra-se uma hierarquia de camada para as informações de cada quadro. Inicialmente, temos a camada de informações de um *Slice* e dentro desta temos uma camada de informações de macrobloco para cada macrobloco pertencente ao *Slice* em questão. No topo da camada de macrobloco são colocados os SEs mb_skip_flag e mb_type. O valor binário do SE mb_skip_flag indica se o macrobloco atual é de um dos tipos básico (SI/I, P/SP, B) ou é do tipo *skip*. Caso o SE mb_skip_flag não esteja presente ou o valor binário dele seja igual a “0”, então o SE mb_type está presente e contém a informação do tipo de macrobloco codificado neste ponto. Depois, para cada submacrobloco 8x8 de um macrobloco codificado como tipo “P_8X8” ou “B_8X8” um SE adicional do tipo sub_mb_type estará presente, representando o tipo do submacrobloco.

A. Macrobloco Skip Flag:
Para a codificação do SE do tipo mb_skip_flag as dependências estatísticas entre os SEs vizinhos são exploradas por meio de um projeto de modelagem simples, mas eficaz. Para um determinado macrobloco C, os modelos de contexto relacionados envolvem o valor do SE mb_skip_flag nos macroblocos vizinhos, da esquerda (A) e acima (B) do macrobloco C. Mais especificamente, o incremento $\chi_{MB\_SKIP\_FLAG}(C)$ ao índice contexto correspondente é definido por:

$$\chi_{MB\_SKIP\_FLAG}(C) = (\text{mb\_skip\_flag}(A) != 0)?\ 0:1 + (\text{mb\_skip\_flag}(B) != 0)?\ 0:1 \quad (4.21)$$

Se algum dos macroblocos vizinhos (A ou B), dado por X, não estiver disponível (pois está fora dos limites do *slice*), então o valor da expressão mb_skip_flag(X) será igual a zero.

B. Macrobloco Type:
Para a codificação do SE do tipo mb_type, um esquema de binarização especial é adotado. Parte deste esquema é ilustrado pela Figura 4.3. O conjunto completo de atribuições de *binstring* para os valores dos SEs podem ser obtidos nas Tabelas 9.27 e 9.28 da norma de especificação do padrão H.264/AVC (ITU-T, 2003).

C. SubMacrobloco Type:
Como mencionado anteriormente, a codificação de SE do tipo sub_mb_type adota um esquema especial de binarização. O conjunto completo de atribuições de *binstring* para o valor destes SEs é descritos na Tabela 9.29 da norma de especificação do padrão H.264/AVC (ITU-T, 2003).

4.3.1.1 Codificação dos Modos de Predição

Considerando que todas as amostras de um macrobloco são preditas, então o modo de predição correspondente tem que ser transmitido. Para um macrobloco codificado através do modo de predição intraquadros, os modos são definidos tanto para os blocos de luminância quanto para os blocos de crominância. Por outro lado, para um macrobloco codificado através da predição inter-quadros, é necessário codificar o índice para o quadro de referência selecionado, bem como as componentes que formam os vetores de movimento diferencial.

A. Modos de Predição Intra para Luma 4x4:
Os blocos de luminância de dimensões 4x4 *pixels* quando codificados através da predição intraquadros utilizam predição em relação a eles próprios e aos modos de predição intraquadros disponíveis. Para codificar o modo de predição do bloco atual em relação ao bloco anterior, utiliza-se um SE definido como prev_intra4x4_pred_mode_flag que é composto por um bit. Quando o SE prev_intra4x4_pred_mode_flag possuir valor diferente de zero, será necessário transmitir o modo de predição atualizado por esse bloco. Essa informação é codificada através do SE rem_intra4x4_pred_mode, que deverá conter um dos outros oito modos de predição intraquadros disponíveis para blocos 4x4 de luminância. Para codificador esses SEs são empregados dois modelos de contextos distintos. O primeiro modelo é específico para codificação do *flag*

enquanto o segundo é utilizado para codificar cada um dos 3 bits (FL) do *binstring* do SE rem_intra4x4_pred_mode.

B. Modos de Predição para Chroma:
Os blocos espacialmente vizinhos tipicamente exibem algumas correlações que são exploradas por um mecanismo simplificado de seleção de modelos de contexto baseado na informação dos modos de predição dos blocos vizinhos, localizados a esquerda (A) e acima (B) do bloco atual. Entretanto, para especificação dos modelos de contexto os modos de predição dos blocos vizinhos não são utilizados. Ao invés disso, uma informação de valor binário, dada por ChPredInDCMode, que indica se o modo correspondente considera o modo mais provável dado pelo valor "0" (predição DC). Assim, o incremento $\chi_{CHPRED}(\mathrm{C})$ para o índice de contexto de um dado macrobloco C é definido por:

$$\chi_{CHPRED}(\mathrm{C}) = (\mathrm{ChPredInDCMode(A)} != 0)?\ 0:1 + (\mathrm{ChPredInDCMode(B)} != 0)?\ 0:1 \tag{4.22}$$

Existem três modelos de contextos possíveis para esse SE, porém eles são aplicados apenas para o primeiro *bin* do *binstring* gerado a partir do valor do SE intra_chroma_pred_mode que foi binarizado através do método TU.

C. Índice de Imagem de Referência:
Primeiramente, a informação relevante para determinar o valor do primeiro *bin* do SE *ref_idx* que representa a informação do índice para o quadro de referência, é extraída dos índices de referência dos macroblocos ou partições de macroblocos vizinhos, tipicamente vizinhos da esquerda (A) e superior (B) da partição atual C. Esta informação é apropriadamente condensada em um *flag* binário assinalado para o SE RefIdxZeroFlag, que indica se o SE ref_idx é selecionado para uma determinada partição. Com uma ligeira variação o incremento $\chi_{RefIdx}(\mathrm{C})$ para o índice de contexto foi modelado para representar a escolha de quatro (invés de três) modelos de contextos como havia sido discutido anteriormente. A seleção do incremento para o modelo de contexto é definido por:

$$\chi_{RefIdx}(\mathrm{C}) = (\mathrm{RefIdxZeroFlag(A)} != 0)?\ 0:1 + 2.((\mathrm{RefIdxZeroFlag(B)} != 0)?\ 0:1) \tag{4.23}$$

A aplicação destes modelos de contexto para o primeiro *bin* do SE ref_idx, binarizado através do método U, é complementado pelo uso de dois modelos de probabilidade adicionais para a codificação dos valores relacionados ao segundo e a todos os outros *bins* do *binstring*.

D. Componentes do Vetor de Movimento Diferencial:
Vetores de movimento diferencial são predições para as quais o modelo de contexto é baseado no erro da predição local. Dado $mvd(X, cmp)$ que denota o valor de um componente de vetor de movimento diferencial na direção $\{horizontal, vertical\}$ relacionado ao macrobloco ou partição de macrobloco identificada por X. Então o incremento $\chi_{mvd}(\mathrm{C}, \mathrm{cmp})$ para um dado macrobloco ou partição de macrobloco C e componente cmp é definido por:

$$\chi_{mvd}(\mathrm{C}) = \begin{cases} 0, \text{se } \mathrm{e}(\mathrm{A},\mathrm{B},\mathrm{cmp}) > 3 \\ 1, \text{se } 3 \leq \mathrm{e}(\mathrm{A},\mathrm{B},\mathrm{cmp}) \leq 32 \\ 2, \text{se } \mathrm{e}(\mathrm{A},\mathrm{B},\mathrm{cmp}) > 32 \end{cases} \tag{4.24}$$

$$\text{com } \mathrm{e}(\mathrm{A},\mathrm{B},\mathrm{cmp}) = |\mathrm{mvd}(\mathrm{A},\mathrm{cmp})| + |\mathrm{mvd}(\mathrm{B},\mathrm{cmp})|$$

onde A e B denotam os macroblocos ou partições de macroblocos vizinhos, da esquerda (A) e superior (B), ao macrobloco ou partição de macrobloco C. O incremento $\chi_{mvd}(\mathrm{C},\mathrm{cmp})$ do índice de contexto é utilizado para calcular o contexto apenas do primeiro *bin* do *binstring* que representa o valor da componente de MVD. Como mencionando anteriormente, o *binstring* para as componentes de MVD é obtido através da aplicação do método de binarização UEG3, com parâmetro de limite $S = 9$. Isso implica que apenas o prefixo do *binstring* (gerado pelo método de binarização unária) é codificado pelo módulo *regular*, para o qual são necessários mais quatro modelos de contexto a serem empregados para o segundo, terceiro, quarto e demais *bins* do prefixo, respectivamente. O sufixo e o sinal do MVD empregam o método de binarização Exp-Golomb não necessitando de modelagem de contexto e, por isso, utilizam o módulo *"bypass"*.

4.3.1.2 Codificação de Informações de Controle

Três SEs adicionais são associados ao nível de macrobloco e estão relacionados a informações de controle, sendo eles: mb_qp_delta, end_of_*slice*_flag e mb_field_decoding. Os quais serão discutidos a seguir.

A. Variação de Parâmetro de Quantização no Macrobloco:
Para manter atualizado o parâmetro de quantização no nível de macrobloco utiliza-se o SE mb_qp_delta em cada macrobloco que não for do tipo *skip* e, para o qual, o valor do SE coded_block_pattern seja diferente de zero. Para codificar o valor inteiro $\delta(\mathrm{C})$ deste SE para um determinado macrobloco C, então o valor $\delta(\mathrm{C})$ é mapeado dentro de um valor positivo $\delta^{+}(\mathrm{C})$ usando a relação:

$$\delta^{+}(\mathrm{C}) = 2|\delta(\mathrm{C})| - ((\delta(\mathrm{C}) > 0 ? 1 : 0) \tag{4.25}$$

então $\delta^{+}(\mathrm{C})$ é binarizado através do método U. Para codificar o valor do primeiro *bin*, um valor de contexto é selecionado baseado na decisão binária $(\delta(P)! = 0)$ para o macrobloco P decodificado, imediatamente, antes do macrobloco ou partição de submacrobloco C. Isso resulta em dois modelos de contexto distintos para serem aplicados ao primeiro *bin* do *binstring*, enquanto para o segundo e demais *bins* outros dois modelos de contextos serão aplicados.

B. *Flag* de Final de *Slice:*
Para assinalamento do último macrobloco de cada *slice* o SE end_of_*slice*_flag deve ser incluído no *bitstring*. Esse SE é codificado utilizando um modelo de probabilidade não adaptativo, especialmente projetado, que é relacionado com as altas probabilidades do símbolo MPS ocorrer.

C. *Flag* para Par de Campo em Macrobloco:
Em quadros codificados com macrobloco adaptativo quadro/campo o SE mb_field_decoding_flag é associado a cada par de macroblocos indicando se

eles são codificados no modo de codificação de quadros ou campos. Para codificar esse *flag* as correlações espaciais entre a decisão do modo de codificação dos macroblocos vizinhos são exploradas para definir a escolha entre três modelos de probabilidades possíveis. Para um dado par de macroblocos C a seleção do modelo correspondente é baseada no incremento $\chi_{mbfield}(\mathrm{C})$ do índice de contexto definido por:

$$\chi_{mbfield}(\mathrm{C}) = mb\_field\_decoding\_flag(A) + mb\_field\_decoding\_flag(B) \tag{4.26}$$

onde A e B representam os macroblocos vizinhos, da esquerda (A) e superior (B), correspondentes ao macrobloco C.

4.3.2 Codificação de Dados Residuais

4.3.2.1 Inovações características

Para a codificação de dados residuais o padrão H.264/AVC define alguns SEs especialmente desenvolvidos para serem utilizados quando o CABAC é o modo de codificação de entropia empregado. Esses SEs e seus esquemas de codificação são caracterizados pelas seguintes diferenciações:

- Um símbolo de 1-bit denominado coded_block_flag e um mapa de significância com valores binários são utilizados para indicar quais os blocos possuem coeficientes de transformadas com valor diferente de zero e qual a localização destes coeficientes dentro do bloco;
- Coeficientes com valores diferentes de zero são codificados na ordem inversa de processamento;
- Os modelos de contexto utilizados para codificar os coeficientes com valor diferente de zero são escolhidos com base no número de coeficientes já codificados com valor diferente de zero para o bloco atual, observando a ordem inversa ao processamento.

4.3.2.2 Processo de codificação para dados residuais

O fluxo do processo de codificação para um simples bloco de coeficientes de transformada é composto por três etapas básicas, conforme ilustrado na Figura 4.4. Inicialmente o SE coded_block_flag é codificado, indicando se existe algum coeficiente de transformada com valor absoluto diferente de zero para o bloco atual. Caso exista, então um laço de repetições percorre todo o bloco, do primeiro ao último coeficiente, gerando o mapa de significância ou dois SEs são produzidos para cada coeficiente. O primeiro SE é o significant_coef_flag que indica se o coeficiente em questão possui valor diferente de zero. O segundo SE é o last_significant_coef_flag que indica se o coeficiente que está sendo processado é o último do bloco com valor diferente de zero ou não. Finalmente, o valor absoluto dos coeficientes de transformada são codificados e transmitidos em ordem de processamento inversa.

A seguir serão apresentadas informações mais detalhadas sobre cada um dos tipos de SE mencionados, bem como uma breve descrição do processo de codificação implementado pelo CABAC.

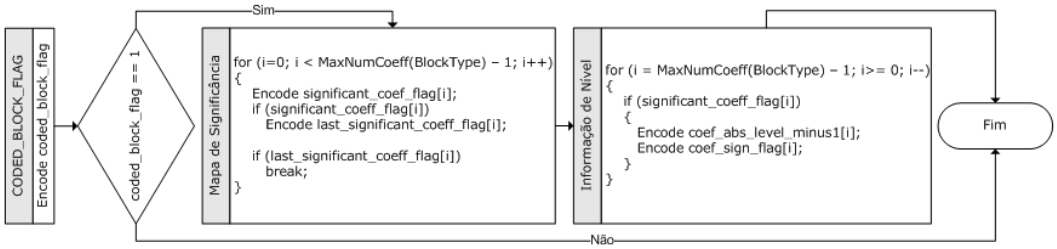


Figura 4.4: Fluxo de codificação para dados residuais dentro da camada de macrobloco.

A. Padrão de Codificação dos Blocos
Para cada macrobloco não *skip* que possua modo de predição diferente de intra_16x16, o SE do tipo coded_block_pattern indica se um dos seis blocos 8x8 - sendo quatro para luminância e dois para crominância - contém algum coeficiente de transformada como valor absoluto diferente de zero. O valor deste SE é binarizado através da aplicação do método de binarização FL, com 4 bits, concatenado ao método TU, com limite S = 2, conforme já foi descrito anteriormente.

B. Indicação de Bloco com Coeficientes de Valor Diferente de Zero
O SE coded_block_flag é um símbolo de um bit que indica se há coeficientes significantes dentro de um bloco de coeficientes de transformadas, para o qual o SE coded_block_pattern indica que há pelo menos uma entrada válida. Se o SE coded_block_flag for igual a zero então nenhuma informação adicional sobre o bloco em questão será transmitida.

C. Processamento dos Coeficientes de Transformadas
Um *array* bidimensional com os coeficientes de transformada consiste daqueles blocos 4x4 para os quais o SE coded_block_flag indica que existem coeficientes com valor absoluto diferente de zero e são, primeiramente, mapeados dentro de uma lista unidimensional usando um determinado padrão de varredura.

D. Mapa de Significância
Se o SE coded_block_flag indica que um bloco possui coeficientes de transformada significantes, então um mapa de significância de valor binário deve ser codificado. Para cada coeficiente, na ordem de varredura, um SE do tipo significant_coef_flag deve ser transmitido. Se o SE significant_coef_flag possuir valor um (“1”) – caso para o qual existe um coeficiente válido na posição atual – então um SE do tipo last_significant_coef_flag deve ser codificado. Caso o valor do SE last_significant_coef_flag seja um (“1”) este coeficiente é o último válido para o bloco atual e não haverá mais nenhum símbolo neste mapa de significância.

E. Informações de Nível:
O mapa de significância determina a localização de todos os coeficientes significantes dentro do bloco coeficientes de transformada quantizados. Os valores dos coeficientes significantes, denominados níveis, utilizam dois SEs para serem codificados: coeff_abs_level_minus1 que representa o valor absoluto dos coeficientes de transformada subtraído por uma unidade; coeff_sign_flag que representa o sinal do coeficiente, onde o valor “1” significa que o valor do coeficiente é negativo. Para codificar o SE coef_abs_level_minus1 utiliza-se o

método de binarização UEG0, conforme pode ser observado na Tabela 4.8. Os níveis são transmitidos na ordem inversa de varredura (iniciando pelo último coeficiente significante no bloco), permitindo a utilização de um ajuste razoável na seleção dos modelos de contexto.

Tabela 4.8: Exemplo de codificação do mapa de significância.

Índice de Varredura	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Coeficientes de transformada	9	0	-5	3	0	0	-1	0	1
significant_coeff_flag	1	0	1	1	0	0	1	0	1
last_significant_coeff_flag	0		0	0			0		1

4.3.2.3 Modelos de contexto para dados residuais

Na codificação de dados residuais definida pelo padrão H.264/AVC existem 12 diferentes tipos de blocos com coeficientes de transformadas, denominados de BlockType, que aparecem na coluna da esquerda Tabela 4.6. Esses tipos de blocos possuem, tipicamente, estatísticas diferentes. Entretanto, para a maioria das sequências e condições de codificação algumas das estatísticas são muito similares. Para manter o número de diferentes modelos de contexto, utilizados na codificação dos coeficientes, razoavelmente pequeno, os tipos de blocos são organizados em cinco categorias, conforme pode ser observado na Tabela 4.6. Para cada uma destas categorias um conjunto especial de modelos de contexto é utilizado para codificar todos os tipos de SE relacionados aos dados residuais, com exceção do SE coded_block_pattern, conforme descrito a seguir.

A. Padrão de Codificação dos Blocos

Considerando que cada bit do *binstring* do SE coded_block_pattern representa um decisão de codificação para um bloco de coeficientes de transformada correspondente, então a seleção do modelo de probabilidade para o SE depende do índice do *bin*. Os índices de *bin* entre 0 e 3 correspondem aos blocos 8x8 de luminância para os quais o incremento χ_{CBP} do índice de contexto, para um dado bloco C 8x8 é relacionado ao índice do *bin*, dado pela variável bin_idx, que é definido por:

$$\chi_{CBP}(\mathsf{C}, bin_{idx}) = (CBP\_bit(A)! = 0)?\ 0:1) + 2\,.\,(CBP\_bit(B)! = 0)?\ 0:1) \tag{4.26}$$

onde $CBP\_bit(A)$ e $CBP\_bit(B)$ representam um bit do padrão de codificação do bloco C relacionado aos blocos 8x8 vizinhos, da esquerda (A) e superior (B). Os *bins* associados aos índices quatro e cinco, os quais são relacionados aos blocos de crominância, possuem uma regra de assinalamento de contexto similar as regras definidas para os blocos de luminância, de forma que ambos os *bins* utilizam juntos oito modelos probabilísticos adicionais, conforme especificado em (ITU-T, 2003).

B. Indicação de Blocos com Coeficientes com Valor Diferente de Zero

A codificação do SE coded_block_flag utiliza quatro modelos probabilísticos distintos para cada um das cinco categorias de blocos especificadas na Tabela 4.7. O incremento $\chi_{CBFlag}(\mathsf{C})$ do índice de contexto para um dado bloco C é determinado por:

$$\chi_{CBFlag}(\mathrm{C}) = coded\_block\_flag(A) + 2.coded\_block\_flag(B) \quad (4.27)$$

onde A e B representam blocos do mesmo tipo do bloco C correspondentes aos vizinhos da esquerda (A) e superior (B), respectivamente. Somente blocos do mesmo tipo são utilizados para determinação de contexto. Os seguintes tipos de blocos são diferenciados: Luma-DC, Luma-AC, Chroma-U-DC, Chroma-U-AC, Chroma-V-DC e Chroma-V-AC. Se nenhum dos blocos vizinhos ($A\ ou\ B$) for do mesmo tipo do bloco atual X, então o valor do SE $coded\_block\_flag(X)$ será definido como zero ("0"). Se um bloco X vizinho (A ou B) estiver fora da área da imagem ou posicionado fora do *slice* corrente então o valor do SE $coded\_block\_flag(X)$ correspondendo será substituido pelo valor padrão ("0"). Assim, enquanto os seis tipos de blocos são distinguidos para determinar o incremento do índice de contexto cinco diferentes conjuntos de modelos (um para cada categoria especificada na coluna da direita da Tabela 4.6) são utilizadas para codificar o SE *coded_block_flag*. Isso resulta em um número total de 20 diferentes modelos de probabilidades para o SE *coded_block_flag*.

C. Mapa de Significância

Para codificar o mapa de significância mais de 15 diferentes modelos de contextos são utilizados em ambos os SE: significant_coeff_flag e o last_significant_coeff_flag. A escolha do modelo de contexto e, consequentemente, o incremento para o índice de contexto χ_{sig} e χ_{last} são dependentes da posição de varredura. Dessa forma, o incremento para o índice de contexto na i^{th} posição de varredura é determinado conforme:

$$\chi_{SIG}(coeff[i])) = X_{LAST}(coeff[i]) = i \quad (4.28)$$

Dependendo do máximo número de coeficientes (MaxNumCoeff) para cada categoria de contexto conforme apresentado na Tabela 4.6. No total 61 contextos diferentes são reservados para os SEs dos tipos significant_coeff_flag e last_significant_coeff_flag.

D. Informações de Nível

A ordem reversa de varredura das informações de nível permite uma estimativa mais confiável sobre as estatísticas, pois ao final do caminho de varredura é provável que se observe a ocorrência de corridas de uns ("1´s"). Consequentemente, para codificar o SE coeff_abs_level_minus1, dois conjuntos de modelos de contexto foram projetados. O primeiro é dedicado para o primeiro *bin*, com índice igual a zero (0). O outro é destinado aos demais *bins* do prefixo do *binstring* gerado através do método de binarização UEG0, que compreende os *bins* com índices entre um (1) e treze (13).

Dado $NumT1(i)$ que denota o número acumulado de uns ("1´s") ocorridos em sequência e $NumLgt1(i)$ que denota o número acumulado de coeficientes de transformada com valor absoluto maior que um ("1") deve-se considerar que ambos os contadores são relacionados à posição de varredura dentro do bloco de coeficientes de transforma atual. Ambos os contadores são inicializados com o valor zero ("0") no inicio da varredura, em ordem reserva, dos níveis e que o número de ambos os contadores são mono tonicamente incrementados com o decremento do índice de varredura através da trilha de coeficientes. Então o

incremento para o índice de contexto do primeiro *bin* do SE coeff_abs_level_minus1 é determinado pelo valor atual de $NumT1$, considerando a seguinte regra adicional: Se mais que três coeficientes anteriores possuírem valor absoluto igual a um ("1") então o incremento 3 para o índice de contexto deve ser escolhido. Quando o nível com um valor absoluto maior que um ("1") for codificado, então o incremento quatro ("4") para o índice de contexto deve ser utilizado para todos os níveis restantes dentro do bloco. Assim, para codificar o primeiro *bin* ($bin\_index = 0$), na posição de varredura i, o incremento $\chi_{AbsCoeff}(i, bin\_index = 0)$ para o índice de contexto associado ao *bin* deve ser definido como:

$$\chi_{AbsCoeff}(i, bin\_index = 0) = \begin{cases} 4, & Se\ NumLgt1(i) > 0 \\ \max(3, NumT1(i)), & caso\ contrário \end{cases} \quad (4.29)$$

Para codificar *bins* com índice entre um e treze ($1 \leq bin\_index \leq 13$) o incremento $\chi_{AbsCoeff}(i, bin\_index)$ para o índice de contexto é determinado pelo valor de NumLgt1, com máximo incremento de quatro ("4"), conforme:

$$\chi_{AbsCoeff}(i, bin\_index) = 5 + \max(4, NumLgt1(i)) \quad (4.30)$$

Para codificar todos os *bins* da parte de sufixo do SE coeff_abs_level_minus1, gerados através do método de binarização UEG0 (com *bin_index* > 13), bem como o sinal deste representado pelo SE *coeff_sign_flag*, emprega-se o método de codificação *bypass*, pois estes SEs não utilizam modelos probabilísticos. Assim, são necessários apenas 49 contextos diferentes para codificação das informações de nível.

A Tabela 4.9 apresenta um exemplo de como determinar o incremento $\chi_{AbsCoeff}(i, bin\_index)$ do índice de contexto relacionado ao *bin_index* utilizado para codificador o valor absoluto dos coeficientes de transformada significantes; onde significantes são todos os coeficientes com valor diferente de zero. Cabe ressaltar que os coeficientes de transformada são processados na ordem inversa a ordem de varredura.

Tabela 4.9: Exemplo de determinação do incremento para o índice de contexto para codificar o valor absoluto dos coeficientes de transformada.

Posição de Varredura *i*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Coeficientes de transformada	9	0	-5	3	0	0	-1	0	1
$\chi_{AbsCoeff}(i, bin\_index = 0)$	4		4	2			1		0
$\chi_{AbsCoeff}(i, bin\_index > 0)$	7		6	5					

4.3.3 Estimativa de Probabilidades

Como mencionado anteriormente, a idéia básica sobre a qual está assentado o novo método de codificação aritmética livre de multiplicações, empregado pelo padrão H.264/AVC, é baseada no pressuposto que as estimativas de probabilidade para cada modelo de contexto podem ser modeladas através de um conjunto limitado e suficientemente representativo de valores. No CABAC sessenta e quatro ("64") valores $\rho_\sigma \in [0.01875, 0.5]$ representativos para as probabilidades foram obtidos a partir do LPS, através da aplicação recursiva da equação:

$$\rho_\sigma = \alpha.\rho_{\sigma-1} \ \forall\sigma = \{1, \dots, 63\}$$
$$com\ \alpha = \left(\frac{0.01875}{0.5}\right)^{\frac{1}{63}}\ e\ \rho_0 = 0.5 \tag{4.31}$$

Onde a escolha do fator de escala $\alpha \approx 0.95$ e a cardinalidade N = 64 do conjunto de probabilidades representa um bom compromisso entre velocidade de adaptação ($\alpha \to 0;\ N\ pequeno$) e a necessidade de uma estimativa suficientemente precisa ($\alpha \to 1;\ N\ grande$). Note que, ao contrário do que ocorre com o codificador MQ, no CABAC não há necessidade de tabular os valores de probabilidades representativas para o LPS $\{\rho_\sigma | 0 \leq \sigma \leq 63\}$. A seguir serão descritos cada valor de probabilidade ρ_σ, os quais são implicitamente acessados pelo modo de codificação aritmética através do índice de contexto σ correspondente.

Como resultado desta organização, no CABAC cada modelo de contexto pode ser completamente determinado por dois parâmetros: a estimativa de probabilidade de LPS atual, o qual é caracterizado por um índice σ entre zero ("0") e sessenta e três ("63"); o valor do MPS ϖ, que fica entre zero ("0") e um ("1"). Assim, no CABAC a estimativa de probabilidade é executada através do uso de 128 estados de probabilidade diferentes, cada um destes eficientemente representados por um valor inteiro de 7-bits. De fato, um destes índices de estado ($\sigma = 63$) é relacionado a um estado autônomo e não adaptativo com valor de MPS fixo, qual é usado apenas para codificar a decisão binária antes da terminação da palavra de código, como será apresentado na sequência. Portanto, apenas 126 estados de probabilidades são efetivamente utilizados para a representação e adaptação de todos os modelos de contexto.

4.3.3.1 Atualização dos Estados de Probabilidades

Conforme descrito recentemente, todos os modelos de probabilidade utilizados pelo CABAC, com exceção de um, são modelos adaptativos, para os quais uma etapa de atualização de estatísticas precisa ser realizada após a codificação de cada símbolo. Na prática, a atualização de um determinado estado de probabilidade depende do índice de estado e do valor do símbolo codificado, seja LPS ou MPS. Como resultado deste processo de atualização um novo estado de probabilidade deve ser obtido, o qual é possivelmente uma modificação da estimativa de probabilidade para o símbolo LPS, ou, se necessário, do símbolo MPS.

A Figura 4.5 ilustra os valores de LPS para as estimativas de probabilidades juntamente com sua regra de transição para atualizar os índices de estados. Quando ocorre um evento MPS o índice de estado é simplesmente incrementado por uma unidade, a menos que o MPS ocorra no índice de estado 62, onde a probabilidade do LPS já atingiu o mínimo valor possível ou, equivalentemente, a probabilidade do MPS já atingiu seu máximo valor possível. No último caso, o índice de estado com valor igual à 62 permanece inalterado até que um evento LPS ocorra, momento no qual o índice de estado será alterado através de um decremento, conforme ilustrado pela linha pontilhada da Figura 4.5. Esta regra é aplicada para todas as ocorrências de um LPS, apenas com a exceção descrita a seguir. Assumindo que um evento LPS tenha sido observado durante a codificação de um estado com índice sigma = 0, o qual corresponde ao caso em que ambos os símbolos (LPS e MPS) são equiprováveis, então o índice de estado é mantido, mas os valores de MPS e do LPS serão trocados um pelo outro. Em todos os outros casos, não importa que símbolo tenha sido codificado, o valor do MPS não deve ser alterado. A derivação da regra de transição para atualizar a probabilidade

do LPS é baseada na seguinte relação entre a probabilidade LPS, dada por p_{old}, e sua contrapartida, dada por p_{new}, conforme definida a seguir:

$$p_{new} = \begin{cases} \max(\alpha . p_{old}, 62), se\ um\ evento\ MPS\ ocorrer \\ \alpha . p_{old} + (1 - \alpha), se\ um\ evento\ LPS\ ocorrer \end{cases} \quad (4.32)$$

No que se refere a uma implementação do processo de estimativa de probabilidades adotadas pelo CABAC é importante observar que essas regras de transição de estados podem ser facilmente realizadas através da utilização de duas tabelas fixas, com 63 entradas de 6 bits cada. Na prática, é suficiente empregar uma única tabela $TransIdxLPS[\sigma]$, que determina, para um determinado estado o novo índice de estado caso um LPS ocorra. Quando um MPS é observado o novo índice de estado pode ser obtido a partir de uma operação de incremento (saturado) do índice de estado σ. Essa operação pode ser representada através da função $min(\sigma + 1{,}62)$.

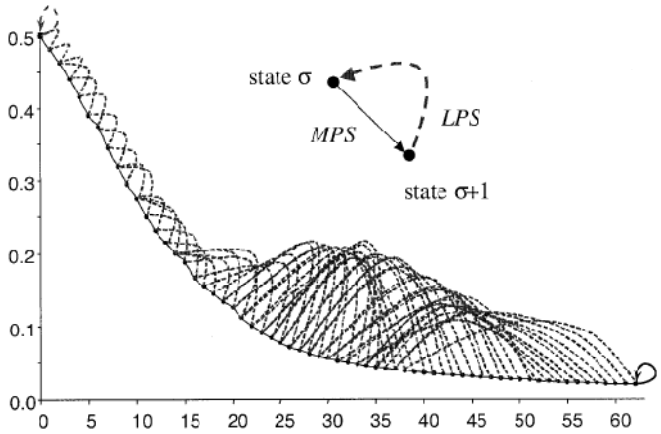


Figura 4.5: Valores de probabilidades de LPS para cada possível estado indexado por σ – modificado a partir de (WIEGAND, 2003).

4.3.3.2 Inicialização e re-inicialização dos Estados de Probabilidades

A unidade básica e auto-contida de codificação de vídeo para o padrão H.264/AVC é o *slice*. Esse fato implica em certas restrições sobre a capacidade de adaptação dos modelos probabilísticos as decisões passadas. A principal restrição relacionada é que o processo de adaptação das estimativas de probabilidades não pode exceder o limite de um *slice*. Assim, ao final de cada unidade de codificação elementar (um *slice*) todas as estimativas de probabilidades devem ser re-inicializadas para limites probabilísticos pré-definidos. Na ausência de qualquer conhecimento sobre a fonte de geração dos dados uma escolha possível para os valores de inicialização das probabilidades seria considerar que todos os estados fossem equiprováveis. Entretanto, o CABAC provê um mecanismo embutido que incorpora a priori algum conhecimento sobre as estatísticas adequadas a fonte. Esse mecanismo fornece valores probabilísticos para inicialização de cada um dos modelos de contexto, denominado processo de inicialização de contextos, que permite um ajuste dos estados de probabilidade inicial em dois níveis, os quais serão descritos a seguir.

A. Inicialização Dependente do Parâmetro de Quantização
   No nível de ajuste mais baixo existe um conjunto padrão de valores de inicialização que são obtidos a partir do parâmetro de quantização do *slice* atual, definido pelo valor do SE SliceQP, promovendo um tipo de pré-adaptação para os estados de probabilidades iniciais, para diferentes condições de codificação.

O detalhamento completo derivado da fórmula que gera os valores de inicialização foge ao escopo deste trabalho, porém o processo de geração dos valores pode ser observado na Equação 4.32.

$$
\begin{aligned}
&1)\ \sigma_{pre} = \max\left(1, min\left(126, \left(\left(\mu_y * SliceQP\right) \gg 4\right) + \upsilon_y\right)\right)\\
&2)\ se\ (\sigma_{pre} \leq 63)\{\\
&\quad \sigma = 63 - \sigma_{pre}\\
&\quad \varpi = 0\\
&\}\ else\ \{\\
&\quad \sigma = \sigma_{pre} - 64\\
&\quad \varpi = 1\\
&\}
\end{aligned}
\tag{4.32}
$$

Onde $\mu_y\ e\ \upsilon_y$ são os parâmetros de segundo nível obtidos a partir de tabelas fixas, definidas de acordo com o tipo de *slice*. A norma de especificação do padrão H.264/AVC associa o valor dos parâmetros $\mu_y\ e\ \upsilon_y$ a cada tipo de SE para cada tipo de macrobloco através da tabela 9.12 que, por sua vez, para cada entrada faz uma referência ao conjunto de tabelas entre 9.13 e 9.24 onde cada um dos 460 contextos possíveis é associado a um dos três possíveis valores do parâmetro cabac_init_idc (ITU-T, 2003)

B. Inicialização Dependente do Tipo de *Slice*
O conceito de pré-adaptação de baixo nível para os modelos de probabilidade foi generalizado pela definição de dois conjuntos adicionais de parâmetros de inicialização de contextos ($\mu_y\ e\ \upsilon_y$) para aqueles modelos de probabilidades especificamente utilizados em *slices* dos tipos P e B. Assim, o codificador é capaz de escolher para esses tipos de *slices*, entre três tabelas de inicialização, qual delas melhor adapta-se para os diferentes cenários de codificação ou tipos de conteúdos dos vídeos. Para possibilitar essa adaptação ao conteúdo do vídeo que irá ser futuramente codificado é necessário escolher a tabela de inicialização com base no tipo de *slice*. Para tal processo é necessário aumentar a quantidade de memória utilizada para armazenar cada uma das tabelas de inicialização de contextos correspondentes. Entretanto, o acesso a essa memória de inicialização de contextos só precisa ser realizado um vez a cada *slice*, gerando pouco impacto ao processo de codificação ou decodificação.

4.3.4 Módulos de Codificação Aritmética Binária

Esta seção apresenta detalhes sobre os diferentes módulos de codificação aritmética binária (MCAB) que compõe o CABAC, conforme especificado pelo padrão H.264/AVC. Na verdade, o bloco de codificação do CABAC consiste de três MCABs, sendo eles: módulo normal, denominado "regular", o qual utiliza modelos de probabilidades adaptativos para definir os limites dos valores de intervalo e deslocamento para o cálculo aritmético; um módulo dedicado para os SEs com distribuição de probabilidade uniforme, denominado*"bypass"*, o qual provê velocidade de codificação; módulo de codificação especial para geração do bit de encerramento da palavra de código, denominado "*terminate"*. A Figura 4.6 ilustra a estrutura lógica do bloco de codificação aritmética, destacando os três módulos que podem ser utilizadas

para codificação de cada *bin* que compõe o *binstring* do valor de um determinado SE. As próximas subseções demonstram aspectos internos de cada um destes módulos.

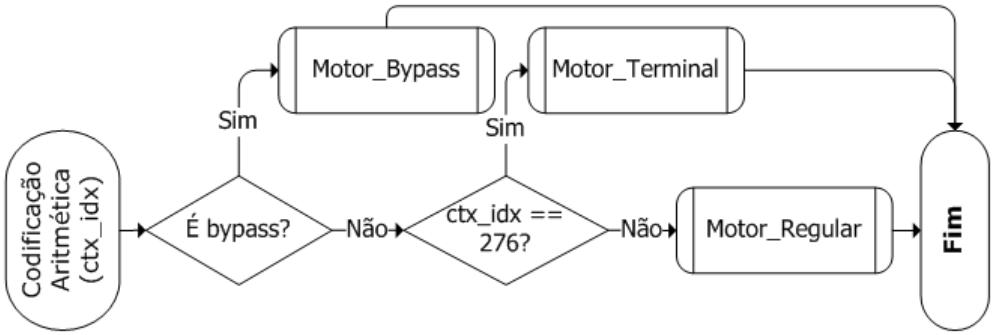


Figura 4.6: Estrutura lógica do bloco de codificação aritmética, destacando os três módulos de codificação aritmética definidos pelo CABAC.

4.3.4.1 Módulo Regular e a Subdivisão do Intervalo

O estado interno de cada MCAB é usualmente caracterizado por duas quantidades: a faixa do intervalo atual, denominado "Range"; a base (ou limite inferior) do intervalo, denominado "Offset". Cabe salientar que a precisão necessária para armazenar esses registradores em cada um dos módulos (*regular*, *bypass* e *terminate*) pode ser reduzida para 9 e 10 bits, respectivamente. Para codificar o valor binário, dado por binVal, em um contexto identificado pelo índice σ relacionado ao estado de probabilidade e pelo valor do MPS, dado por ϖ, é necessário realizar quatro passos fundamentais descritos a seguir:

No primeiro e mais importante passo, o intervalo é subdividido de acordo com as estimativas de probabilidades. Este processo de subdivisão do intervalo envolve três operações básicas, como apresentado no primeiro retângulo do lado esquerdo da Figura 4.7. Na primeira o intervalo atual, dado por R, é aproximado por um valor quantizado, dado por $Q(R)$, usando particionamento igualitário do intervalo $2^8 \leq R \leq 2^9$ em quatro partes. Porém, ao invés de utilizar os valores quantizados correspondentes $Q_0, Q_1, Q_2 \, e \, Q_3$ explicitamente, o CABAC utiliza os valores $Q(R)$ para definição do índice ρ, o qual pode ser eficientemente calculado através de uma combinação de operações de deslocamento e mascaramento de bit, conforme: $\rho = (R \gg 6)$ & 3. Então, este índice ρ e o índice do estado de probabilidade σ são utilizados para selecionar a célula correta de uma tabela bidimensional, denominada "TabRangeLPS", que contém o subintervalo relacionado ao símbolo LPS, dado por R_{LPS} como apresentado na Figura 4.7. A tabela "TabRangeLPS" contém 64x4 entradas com os valores pré-calculados de $\rho_\sigma \, . \, Q_\rho$ para $0 \leq \sigma \leq 63$ e $0 \leq \rho \leq 3$ com 8 bits de precisão.

Dada a contraparte do intervalo, definida por $R - R_{MPS}$, correspondente ao subintervalo relacionado ao símbolo MPS, então o valor de *bin*, dado por *binVal*, é escolhido em um segundo passo de codificação. Se o valor de *binVal* é igual ao valor do MPS ϖ esperado, então o subintervalo inferior é selecionado de tal forma que o valor de L não é alterado. Caso contrário, o subintervalo superior é selecionado tornando o valor do intervalo igual ao valor de R_{LPS}.

No terceiro passo do processo a codificação aritmética binária é executada, através do módulo *regular*, atualizando os estados de probabilidades. Esse processo é ilustrado

pela área achurada da Figura 4.7. Finalmente, no quarto passo, a etapa de renormalização dos registradores L e R (que será descrita na sequência) é realizada.

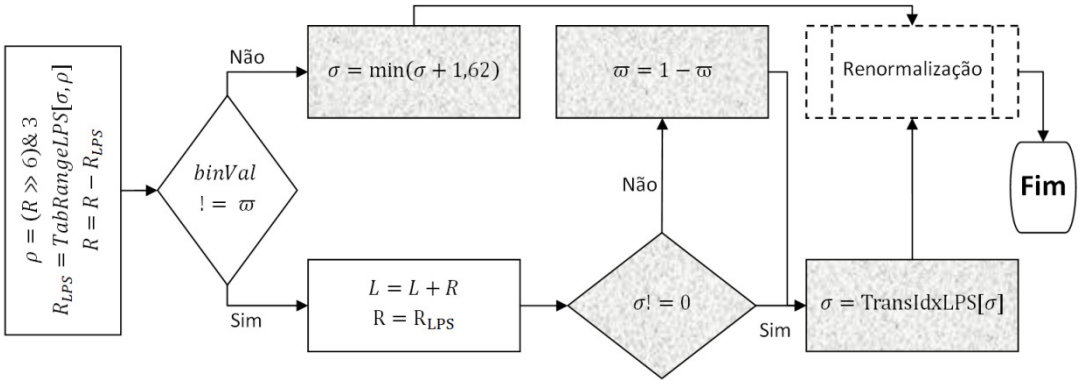


Figura 4.7: Fluxo de execução do módulo de codificação *regular*, ilustrando o processo para decodificar cada *bin*.

4.3.4.2 Módulo Bypass

Para incrementar a velocidade de codificação (e decodificação) dos símbolos o CABAC prevê um módulo alternativo para codificação aritmética binária, denominado *bypass*, o qual apresenta simplificações significativas quando comparado ao módulo *regular*. Primeiro, uma estimativa de probabilidade e o processo de atualização é estabelecido. Segundo, o intervalo é subdividido de tal forma que dois subintervalos equivalentes sejam produzidos dentro do estágio de subdivisão. Mas ao invés de reduzir explicitamente o intervalo R pela metade, o valor da variável L, que corresponde à base do intervalo, é dobrado antes que a escolha entre o limite superior ou inferior seja realizada baseada no valor do símbolo codificado (0 ou 1). Desta forma, a duplicação dos valores de R e L elimina a etapa de renormalização subsequente. A Figura 4.8 apresenta o diagrama de blocos para o módulo *bypass*.

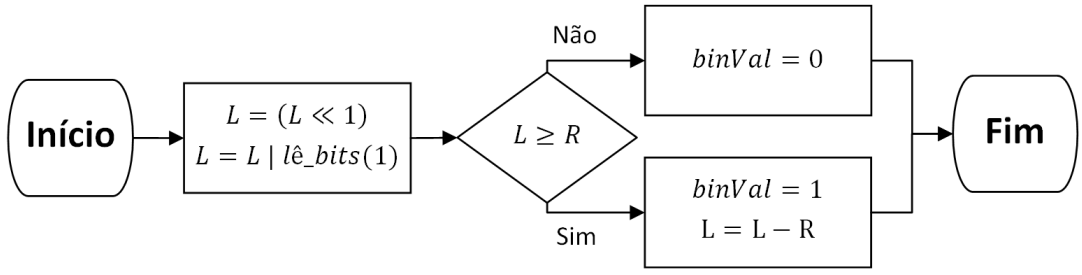


Figura 4.8: Fluxo de execução para o módulo *bypass*.

4.3.4.3 Renormalização e Controle de Carry

Uma operação de renormalização é necessária depois da subdivisão do intervalo se o valor do novo intervalo ficar fora da faixa de valores entre $[2^8, 2^9)$. Quando o processo de renormalização é executado no decodificador um ou mais bits podem ser consumidos do *bitstream* para "alimentar" o valor do registrador de base L, enquanto o valor do intervalo R é dobrado. Este processo é repetido até que o valor do intervalo R atinja a faixa de valores esperada. A Figura 4.9 ilustra este processo de renormalização.

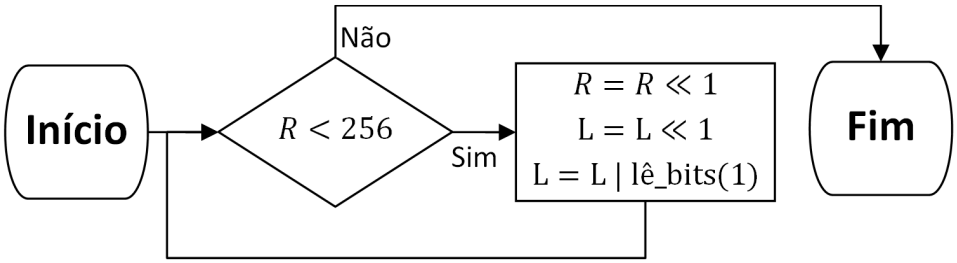


Figura 4.9: Fluxo de execução para o processo de renormalização.

4.3.4.4 Módulo Terminate

Um módulo de codificação especial, denominado "*terminate*", é definido para codificar a palavra de código que marca o final da sequência de codificação de um *slice*. Esse módulo de codificação é associado a um estado de probabilidade fixo, identificado pelo índice $\sigma = 63$ que endereça a tabela transição de estados, "$TabRangeLPS[63, \rho]$", a qual também é indexada pelo valor quantizado de R_{LPS} representado pelo índice ρ. Esse método de codificação é utilizado para codificar o SE *end_of_slice* que normalmente é codificado observando-se uma decisão por LPS, a qual produz 7 bits na etapa de renormalização subsequente. Além disso, para evitar a incidência de ambiguidades é necessário codificar dois bits adicionais, garantindo que o decodificador irá conseguir identificar onde o processo deve ser encerrado. A Figura 4.10 ilustra o processo de codificação para o método *terminate*.

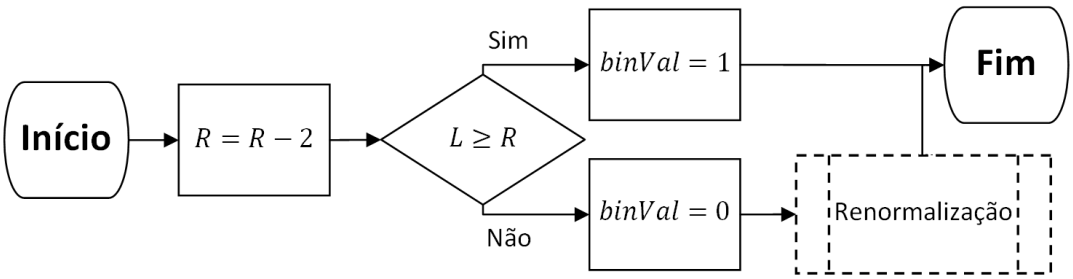


Figura 4.10: Fluxo de codificação para o módulo *terminate*.

5 ANÁLISES DE COMPORTAMENTO ESTÁTICO E DINÂMICO

Dentre as observações oriundas da fase de estudo dos algoritmos que compõem o CABAC duas, em especial, merecem destaque:

- Existe uma dependência de dados muito forte que transforma os processos CABAC/CABAD em um grande gargalo para os sistemas de compressão de vídeo;
- Não é possível determinar o número de bits produzidos durante o processo de decodificação para cada tipo de SE codificado, fato que dificulta determinar a vazão necessária para atender o pior caso.

Além disso, deve-se considerar que o fluxo de execução dos processos (codificação e decodificação) é influenciado pelos parâmetros de codificação e pelos tipos de *slice* e macrobloco que estiverem sendo processados. Dessa forma, foi necessário desenvolver uma nova etapa de pesquisa dedicada a análise sobre o comportamento do fluxo de execução do CABAD, visando identificar características estáticas e dinâmicas.

A análise comportamental foi segmentada em duas linhas, sendo a primeira destinada à exploração de características denominadas estáticas enquanto a segunda observa características dinâmicas. Neste contexto, a utilização dos termos "estática" e "dinâmica" possui definições próprias. A análise de comportamento estático compreende as variações observadas no fluxo de decodificação, decorrentes de parâmetros utilizados durante o processo de codificação ou dos tipos de *slice* e/ou macrobloco a serem decodificados, conforme definições contidas no padrão H.264/AVC. A análise de comportamento dinâmico explora diferenças no fluxo de decodificação relacionadas ao conteúdo processado, ou seja, variações de comportamento que independem de parâmetros de decodificação e/ou tipo de macrobloco. Esse comportamento dinâmico é interessante, pois permite traçar um perfil de decisões típicas com base na observação de um grande conjunto de amostras.

A identificação de variações no fluxo de execução do CABAC/CABAD, sejam elas de origem estática ou dinâmica, não constitui uma inovação, pois essas constatações podem ser observadas em outros trabalhos encontrados na literatura como, por exemplo, Chen, Chang e Lin (2005) e Yu, He (2005). Contudo, a extensão e detalhamento das analises descritas são, normalmente, restritas a um subconjunto de interesse dos autores e, possivelmente por questões de limitação de espaço, são apresentadas de forma superficial. Em linhas gerais, é possível destacar quatro grupos de informações comumente relatadas, sendo elas: I - ganho na taxa de compressão obtida pelo CABAC

frente ao CAVLC; II - incremento na complexidade computacional de compressão quando utilizado o CABAC ao invés do CAVLC; III - número médio de ciclos para codificar e/ou decodificar cada tipo de macrobloco; IV - a taxa de utilização de cada um dos MCABs. Em relação aos objetivos deste trabalho, o primeiro e segundo grupos não fornecem subsídios suficientes para elaboração de propostas arquiteturais enquanto os dois últimos são de extrema relevância.

Nas próximas seções as análises mencionadas serão apresentadas, sendo que a seção 5.1 é destinada as análises de comportamento estático enquanto a seção 5.2 apresenta dados relacionados às analises de comportamento dinâmico baseadas em simulações sobre sequências de vídeo padronizadas.

5.1 Características do Comportamento Estático

O conjunto de algoritmos utilizados pelo CABAC foi descrito ao longo do Capítulo 4. Entretanto, o detalhamento das variações no fluxo de decodificação relacionadas aos parâmetros de codificação ou tipos de *slices* e macroblocos não foram abordadas. Inicialmente é necessário observar que o próprio fluxo do CABAD, ilustrado pela Figura 5.1, prevê caminhos alternativos de acordo com o estágio de decodificação. Esse processo de decodificação é aplicado para todos os SE, dentro do nível de macroblocos, até que o SE que marca o final do *slice* seja encontrado. As etapas apresentadas pelo fluxo podem ser agrupadas em três estágios: inicialização, decodificação e encerramento. A inicialização engloba as etapas 1, 2, 3 e 4. O estágio de decodificação contém o laço de repetição que processa todos os *bins* produzidos por um SE. Finalmente, no estágio de encerramento, os MCABs são preparados para que a decodificação do próximo SE possa ser executada.

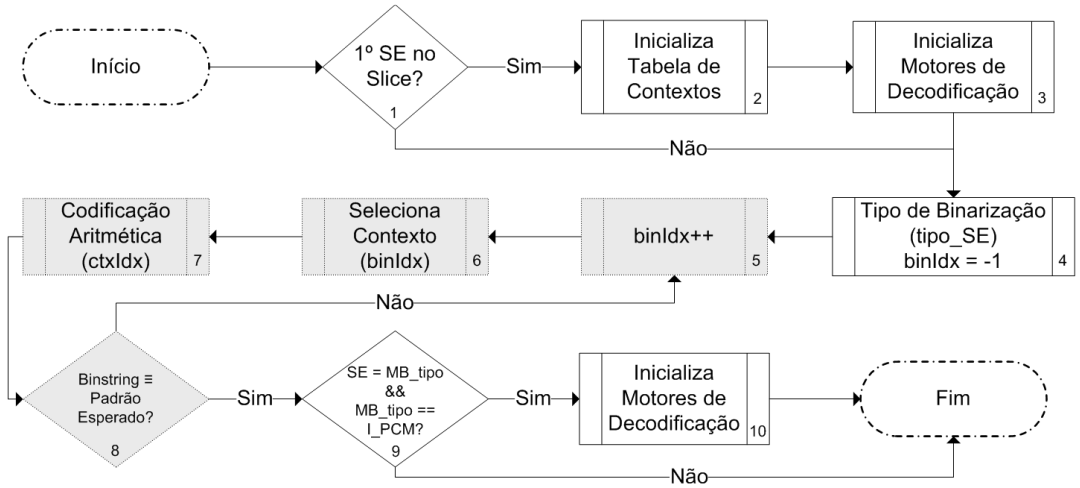


Figura 5.1: Diagrama de fluxo de execução do CABAD.

O primeiro passo do fluxo é identificar se o SE atual é o primeiro a ser decodificado dentro do *slice*, pois quando isso ocorrer a etapa de inicialização da tabela de contexto deve ser acionada. Essa inicialização é dependente do tipo *slice*, pois com base nesta informação, modelos probabilísticos diferenciados podem ser selecionados. Após concluir a inicialização da tabela de contextos, é necessário realizar a inicialização dos MCABs para garantir que o intervalo de probabilidades esteja dentro da faixa de valores.

A quarta etapa é responsável por identificar o tipo de binarização esperada para o SE a ser decodificado, além de inicializar o contador de *bins* decodificados do SE atual. O processo de identificar o tipo de binarização é possível graças ao controle do decodificador que mantém informações sobre o SE decodificado anteriormente e qual o próximo SE esperado com base nos tipos de *slice* e macrobloco atuais (processo de *parsing*).

As etapas 5, 6, 7 e 8, que possuem suas formas acuradas na Figura 5.1, representam o laço de repetição que decodifica um *bin* por iteração até que todos os *bins* que compõem o *binstring* que representa o valor do SE sejam decodificados. Essas etapas envolvem o incremento do contador de *bins* decodificados, a seleção do índice de contexto adequado para a leitura da memória de contextos, a subdivisão recursiva do intervalo de probabilidades através do cálculo aritmético binário realizado por um dos módulos existentes e a comparação entre a sequência de *bins* produzida e o padrão de *bins* esperados. O laço de iterações será repetido até que uma sequência de *binstring* compatível com o padrão de *binstring* esperado seja encontrada.

Após concluir o laço de decodificação, é necessário avaliar o teste condicional indicado pela etapa 9 e de acordo com o resultado, selecionar a nova etapa. Se a condição da etapa 9 for satisfeita então a etapa 10 será a próxima. Na etapa 10 o processo de inicialização dos MCABs é executado. Depois da etapa 10 ou quando a condição da etapa 9 for falsa, em ambos os casos, o próximo passo será o encerramento do fluxo de decodificação.

O diagrama de blocos com o fluxo de decodificação (ilustrado pela Figura 5.1) apresenta dois caminhos alternativos, sendo o primeiro no estágio de inicialização e o segundo no estágio de encerramento, porém, as variações significativas para o processo de decodificação residem dentro das etapas. No processo de inicialização das tabelas de contexto (representado pela etapa 2 da Figura 5.1) o fluxo é influenciado por parâmetros empregados durante o processo de codificação do vídeo – parâmetro de quantização e tipo de *slice* – conforme apresentado na seção 4.3.3.2. Os processos de identificação do tipo de binarização e a seleção de contexto (representados pelas etapas 4 e 6, respectivamente, da Figura 5.1) possuem diversos caminhos de execução alternativos sendo, basicamente, um para cada tipo de SE; conforme apresentado ao longo das seções 4.2.1, 4.2.2, 4.3.1 e 4.3.2. A etapa de codificação aritmética (representada pela etapa 7 da Figura 5.1) também possui comportamentos diferenciados que variam de acordo com o tipo de SE que estiver sendo decodificado assim como pelo estágio de decodificação.

As peculiaridades do fluxo de decodificação descritas até o momento são importantes, porém, a principal característica que gera variações no fluxo – que ainda não foi abordada – está relacionada com o conjunto de SEs que podem estar presentes em cada tipo de *slice*. Cada tipo de *slice* pode conter um ou dois tipos de macroblocos e, cada macrobloco, pode ser composto por um conjunto de SEs, o qual pode sofrer variações de acordo com as ferramentas empregadas durante o processo de codificação e/ou por singularidades oriundas do conteúdo que estiver sendo tratado. Baseado neste fato Yu e He (2005) e, posteriormente, Bingbo et al. (2007) analisaram o padrão H.264/AVC tendo observado que os SEs podem ser organizados em dois grandes grupos. O primeiro grupo contém apenas informações relacionadas ao tipo de macrobloco, submacroblocos e ferramentas de compressão utilizadas; essas informações são denominadas informações de cabeçalho. O segundo grupo contém apenas

informações relacionadas aos dados residuais gerados pelas ferramentas de predição de movimento, as quais são denominadas informações de resíduos.

A distinção dos SEs por grupos é importante, pois o conjunto de informações de cabeçalho muda significativamente de acordo com o tipo de macrobloco que estiver sendo processado, enquanto as informações residuais são, basicamente, as mesmas para todos os tipos de macroblocos. Outra observação relatada por Yu e He (2005) indica que os SEs do primeiro grupo, em geral, ocorrem poucas vezes dentro de um macrobloco, gerando poucos *bins* em cada ocorrência enquanto que os SEs do segundo grupo ocorrem várias vezes e produzem muitos *bins* em cada ocorrência. A Tabela 5.1 apresenta todos os SEs processados pelo CABAC organizados em grupos e níveis. Os níveis classificam os SEs de acordo com o tipo de informação que esses elementos representam. As ocorrências de cada SE em cada tipo de macrobloco e o número mínimo e máximo de *bins* (separados em prefixo e sufixo) também são apresentados na Tabela 5.1.

Tabela 5.1: Relação dos tipos de SE nos macroblocos onde eles ocorrem e o número de *bins* produzidos por cada tipo de SE, organizados entre sufixo e prefixo.

Grupo	**Nível**	**Tipo de SE**	**Ocorrências por Tipo Macrobloco**			**BINS**			
						Prefixo		Sufixo	
			I	**P**	**B**	**Mín.**	**Máx.**	**Mín.**	**Máx.**
Informações de Cabeçalho	**Macrobloco**	mb_skip_flag *	1	1	1	1	1		
		mb_field_decoding_flag *	1	1	1	1	1		
		mb_type	1	1	1	3	8		
		sub_mb_type		0-4	0-4	0	6		
		mb_qp_delta	1	1	1	6	6		
	Predição Inter	ref_idx_l0		1-4	1-4	0	5		
		ref_idx_l1		1-4	1-4	0	5		
		mvd_x (horizontal)		4-16	4-16	0	8	0	N
		mvd_y (vertical)		4-16	4-16	0	8	0	N
	Predição Intra	intra_chroma_pred_mode	0-8			0	3		
		prev_intra4x4_pred_mode	0-24			1	1		
		rem_intra4x4_pred_mode	0-24			0	3		
Informações de Resíduos	**Bloco**	coded_block_pattern	1	1	1	0	4	0	2
		coded_block_flag	1-24	1-24	1-24	0	1		
	Coefi-cientes	significant_coeff_flag	1-384	1-384	1-384	0	1		
		last_significant_coeff_flag	1-384	1-384	1-384	0	1		
		coeff_abs_level_minus1	1-384	1-384	1-384	0	13	0	K
Slice		end_of_*slice*	1	1	1	1	1		

* SE agrupado nas informações relacionadas ao macrobloco, porém ocorre antes do macrobloco propriamente dito.

Na Tabela 5.1, algumas células do grupo de colunas relacionadas às ocorrências de cada tipo de SE por tipo de macrobloco possuem valor indefinido, o que significa que o SE em questão não ocorre para esse tipo de macrobloco. O mesmo evento ocorre no grupo de colunas relacionado aos *bins* produzidos para o prefixo e sufixo de cada tipo de SE, porém, neste caso, significa que a formação do *binstring* para um determinado SE não possui duas partes ou que o número de *bins* em cada parte pode sofrer variações conforme detalhado na seção 4.3.

Além da análise das ocorrências dos SEs em cada tipo de macrobloco, o número de *bins* produzidos em cada ocorrência e sua organização em grupos também é necessário avaliar a taxa de utilização dos diferentes MCABs. Conforme definições do padrão H.264/AVC, a modelagem de contexto para os *bins* de todos os SEs é especializada e a codificação aritmética binária precisa ser realizada através do módulo *regular*. Entretanto, algumas situações especiais, como os *bins* que compõe o sufixo dos MVD e dos coeficientes de transformada, podem ser codificados através do módulo *bypass*, pois utilizam uma distribuição de probabilidades uniforme. Além disso, para o SE *end_of_slice* apenas o módulo *terminate* é utilizado. A Tabela 5.2 apresenta a relação completa de todos os SEs e os módulos de codificação aritmética utilizados por cada um deles. A barra de separação no interior de algumas células do grupo de colunas com os *bins* produzidos em cada módulo representa o número de *bins* produzidos para cada módulo nos diferentes tipos de macroblocos, considerando a ordem I, P, B; enquanto o hífen separando os números representa o intervalo entre o número mínimo e máximo de *bins* que pode ser produzido para todos os SEs em cada módulo de codificação.

Tabela 5.2: Relação entre SE e MCAB utilizado.

Tipo de SE	**BINS por Módulo**		
	Regular	**Terminate**	**Bypass**
mb_skip_flag	1		
mb_field_decoding_flag	1		
mb_type	7/3/8		
sub_mb_type	2/6		
mb_qp_delta	6		
ref_idx_l0	5		
ref_idx_l1	5		
mvd_x (horizontal)	0-8		0-N
mvd_y (vertical)	0-8		0-N
intra_chroma_pred_mode	3		
prev_intra4x4_pred_mode	1		
rem_intra4x4_pred_mode	3		
coded_block_pattern	5-6		
coded_block_flag	1		
significant_coeff_flag	1		
last_significant_coeff_flag	1		
coeff_abs_level_minus1	0-13		0-K
end_of_*slice*		1	

O exame da Tabela 5.1 conduz a conclusão que os SEs relacionados aos valores dos coeficientes de transformada e os MVDs causam impacto mais significativo ao processo de decodificação, pois podem ocorrer maior número de vezes e gerar mais *bins* a cada ocorrência. Efetuando-se uma análise similar sobre a Tabela 5.2 é possível concluir que o módulo de codificação *regular* possui taxa de utilização mais elevada, já que é utilizado por quase todos os SEs enquanto os módulos *terminate* e *bypass* são empregados apenas em situações pontuais. Contudo, essas conclusões merecem a ressalva que o processo de compressão de vídeo visa gerar o menor conjunto de informações capaz de representar a sequência de vídeo original e, sendo assim, o pior caso – apresentado pelas Tabelas 5.1 e 5.2 – pode ocorrer poucas vezes, o que dificulta qualquer afirmação sobre a taxa de utilização dos módulos de codificação. Essa observação é baseada no fato que o número de ocorrência dos SEs pode variar em

função do conteúdo que estiver sendo processado, bem como pelas decisões tomadas durante o processo de codificação. Assim, torna-se necessário avaliar o caso médio através de experimentos de codificação sobre um grande número de amostras com a utilização de parâmetros variados a fim de obter dados representativos sobre a relevância de cada SE e dos MCABs.

Outro aspecto relevante é a questão da gerência de contextos que é um tema frequentemente discutido na literatura. O acesso a memória de contextos torna-se um problema, pois, normalmente, é necessária uma operação de leitura a esta memória para cada *bin* a ser decodificado. Via de regra um único modelo de contexto é aplicado para cada *bin* de um determinado SE ou para vários *bins* consecutivos deste SE. Entretanto, para alguns casos especiais, vários modelos de contexto podem ser associados a um único *bin,* caracterizando uma modelagem de contexto condicional que pode oferecer ganhos na eficiência da codificação. Nestes casos especiais, o modelo de contexto adequado para cada situação deve ser selecionado com base na informação destes mesmos SEs decodificados anteriormente nos macroblocos vizinhos; essas observações são baseadas no estudo do padrão H.264/AVC (ITU-T, 2005).

Avaliando a intensidade de acessos à tabela de contextos, a tarefa de gerenciar os acessos a esses contextos torna-se crítica, pois pode minimizar o problema ocasionado pela latência decorrente dos acessos a memória. Propostas para solucionar essa questão podem ser encontradas em Yu e He (2005); Bingbo, et al. (2007); Mei-hua, et al., (2007). Os primeiros, Yu e He (2005), relacionaram à incidência dos acessos a memória de contexto com os seus respectivos SEs e a ordem de decodificação destes SEs no fluxo. Baseado neste estudo, os autores organizaram os contextos em 18 grupos de acordo com os SE que os utilizavam e avaliaram quantos contextos distintos poderiam ser empregados durante o processo de decodificação de cada um dos diferentes tipos de SEs. Essa informação pode ser deduzida através do exame das tabelas 9.25 e 9.30 presentes na norma de especificação do padrão H.264/AVC (ITU-T, 2005). Destes 18 grupos de contextos, apenas um possui mais de 14 contextos diferentes. Essa informação serviu de base para as propostas arquiteturais de registradores de *cache* apresentadas por Yu e He (2005); Bingbo, et al. (2007); Mei-hua, et al., (2007) com a ressalva que este último optou por um particionamento em 25 grupos de acordo com a ordem de decodificação dos SEs.

5.2 Características do Comportamento Dinâmico

Em face da dificuldade de quantificar, de forma estática, os dados produzidos pelo CABAC, muitos autores optaram por realizar estudos de caso a fim de traçar perfis de comportamentos dinâmicos que pudessem ser empregados para a avaliação do caso médio. Esses estudos de caso possuem variações significativas entre si, sendo cada qual voltado para a identificação de alguns indicadores específicos. Além disso, as análises apresentadas utilizam um pequeno conjunto de sequências de vídeo e não fornecem o detalhamento completo dos parâmetros utilizados durante o processo de codificação. Contudo, é possível elencar um subconjunto de questões mais comumente apresentadas, dentre as quais podem ser citadas:

- A taxa de utilização dos MCABs;
- A relação entre as decisões por símbolos MPS frente aos LPS;
- A relação entre bits consumidos do bitstream e os *bins* produzidos;

- A taxa de compressão do CABAC frente ao CAVLC; e
- A taxa de bits processados pelo CABAD em relação ao total de bits no bitstream.

Kim e Park (2006) propõem uma arquitetura com ênfase na exploração do processamento especulativo e, para isso, apresentam uma avaliação sobre as decisões entre símbolos LPS e MPS durante o processo de decodificação e qual o impacto destas decisões sobre os registradores internos dos módulos de codificação aritmética que justificaria o emprego da técnica proposta. Entretanto, os dados apresentados consideram apenas duas sequências de vídeo e não trazem um detalhamento sobre a resolução destas sequências ou dos parâmetros utilizados para sua codificação.

Determinar a taxa de *bins* produzidos pelo CABAD a partir de um bitstream codificado é um grande desafio, pois essa informação depende dos parâmetros de codificação empregados e, principalmente, de características da fonte de dados originais. Essa informação é fundamental para definir a vazão necessária para que uma arquitetura de *hardware* dedicada ao CABAD possa obter capacidade de processamento em tempo real. Segundo Bingbo et al. (2007) a taxa de *bins* produzidos poderia ser definida em relação a taxa de bits consumidos pelo CABAD, sendo que essa relação ficaria entre 1.2 e 1.4 vezes. Contudo, a análise realizada considera apenas quadros do tipo I e os dados coletados foram obtidos a partir de apenas três sequências de vídeo, para as quais nenhuma informação adicional, como por exemplo, resolução e parâmetros de codificação são apresentados.

Para validar a proposta arquitetural apresentada em Yang et al. (2006), os autores utilizaram um vasto conjunto de sequências de vídeo padronizadas, para as quais forneceram dados sobre os parâmetros de codificação empregados. Com os dados obtidos durante o processo de validação, os autores conseguiram definir o número médio de ciclos necessários para processar cada tipo de quadro. Entretanto, o parâmetro de quantização utilizado para codificar todas as sequências foi o mesmo, fato que limita a avaliação dos resultados obtidos. Esse tipo de avaliação de desempenho baseado em número de ciclos necessários para decodificar cada tipo de quadro já havia sido utilizado em Chen, Chang e Lin (2005), porém neste caso, exceto pela informação sobre o número de quadros avaliados, nenhuma caracterização sobre as sequências de vídeo ou parâmetros de codificação utilizados era fornecida. Posteriormente, em Zheng et al. (2007), o mesmo método é aplicado, porém apenas uma sequência de vídeo é empregada.

Outro aspecto que desperta curiosidade é o ganho real sobre a taxa de compressão do *bitstream* oferecido pelo CABAC frente ao CAVLC, visto que os relatos encontrados na literatura apresentam variação significativa. Conforme (YU, HE, 2005), o ganho na taxa de compressão pode atingir até 16% enquanto (BIN, et al., 2007) cita que o ganho pode oscilar entre 9% e 14%, mesma faixa mencionada anteriormente por Marpe e Wiegand (2003). Contudo, essa variação foi redefinida por Marpe, Wiegand, e Sullivan (2006) para uma faixa mais abrangente, entre 10% e 20%. No caminho contrário Chen e Lin (2007) apontam para uma redução mais tímida de apenas 7% sobre o tamanho do *bitstream*, enquanto outros autores como, por exemplo, Eeckhaut, et al. (2006) optam por não determinar deixando indicação que os ganhos são significativos sem expressar o percentual.

5.2.1 Caracterização do Cenário de Avaliação

Revisando o estado da arte sobre avaliações de características de comportamento do CABAD foi possível concluir que a quantidade e diversidade dos dados apresentados não ofereciam bases claras para tomada de decisão. Dessa forma, optou-se por fazer um conjunto de experimentos para coletar os dados desejados de forma padronizada, ou seja, com sequências de vídeo e parâmetros de codificação conhecidos, além de utilizar variações de resolução e parâmetros de quantização que permitissem obter uma ampla cobertura para diferentes cenários de compressão. Para isso, tomou-se como base o *software* JM10.2 (JM na versão 10.2) que é implementação de referência e prova de conceito para o padrão H.264/AVC (SÜHRING, 2008).

A versão oficial do decodificador presente no JM10.2 foi modificada – tarefa que resultou no desenvolvimento de aproximadamente 2,5k linhas de código adicionais, as quais foram embutidas em pontos específicos do *software* de referência, o que significa um acréscimo, aproximadamente, 10% sobre o tamanho do código fonte (desconsiderando comentários) do decodificador presente no JM10.2 – com a finalidade de incluir as rotinas necessárias para coletar dados do processo de decodificação. Não obstante, todas as informações que chegam até a entrada do CABAD, bem como as informações produzidas na saída, foram capturadas e armazenadas em arquivos específicos para que, posteriormente, fossem utilizadas durante o processo de validação da arquitetura desenvolvida.

Para obter um conjunto de dados representativos, optou-se por realizar avaliações sobre as sequências de vídeo frequentemente encontradas na literatura. Todas as sequências de vídeo estavam no formato YUV 4:2:0, ou seja, com subamostragem das componentes de cor na razão de 4:1 em relação a luminosidade. A Tabela 5.3 apresenta a relação completa de todas as sequências de vídeo utilizadas durante o processo de coleta de dados. No total foram utilizadas 60 sequências de vídeo que estão dividas em 18, 17, 18 e 7 entre as resoluções QCIF, CIF, D1 e HD1080, respectivamente.

Tabela 5.3: Relação das sequências de vídeo utilizadas em cada uma das resoluções.

Sequências de Vídeo			
QCIF (176×144)	**CIF (352×288)**	**D1 (720×480)**	**HD1080 (1920×1080)**
Akiyo	Bridge-close	Abstract	Bluesky
Bridge-close	Bridge-far	Artant	Pedestrian
Bridge-far	Bus	Chips	Riverbed
Carphone	Coastguard	Concert	Rush-hour
Claire	Container	F1	Station2
Coastguard	Flower	Football	Sunflower
Container	Foreman	Ice	Tractor
Foreman	Hall	Leaves	
Grand-mother	High Way	Letters	
Hall	Mobile	Mobile	
Highway	Mother-daughter	Parkrun	
Miss-america	News	Rafting	
Mobile	Paris	Rugby	
Mother-daughter	Silent	Seawall	
News	Stefan	Suzie	
Salesman	Tempete	Tempete	
Silent	Waterfall	Toweres	
Suzie		Waterfall	

A partir das sequências de vídeo indicadas na Tabela 5.3, foram realizados dois processos de compressão sendo que, para cada um deles, foram utilizadas todas as sessenta sequências. A diferença entre esses processos reside no método de codificação de entropia utilizado, sendo um o CAVLC e no outro o CABAC. Além disso, para cada uma das sequências, em cada um dos processos, foram adotados sete conjuntos de parâmetros de quantização diferentes para os *slices* do tipo I e P. O conjunto de pares {0:0, 6:0, 12:6, 18:12, 24:18 e 36:26} representa os valores utilizados para os parâmetros QPISlice e QPPSlice. Dessa forma, o processo final resultou em um total de 880 sequências de vídeo codificadas – 60 (sequências) x 7 (pares de parâmetros QP) x 2 (métodos de entropia: CAVLC/CABAC). No geral o processo de codificação das amostras utilizou 200 quadros, os quais obedeciam à organização dos tipos de *slices* conforme o parâmetro: GOP = IPBB. A compressão de todas as amostras foi destinada ao perfil Main do padrão H.264/AVC, com as seguintes configurações de parâmetros de codificação: Profile_IDC = 77, Level_IDC = 40, SymbolMode = CABAC, ME_SEARCH = 32, RDO = ON.

As imagens apresentadas a seguir, correspondendo as Figuras 5.2 até 5.14, trazem a representação visual de todas as sequências de vídeo citadas na Tabela 5.3.

Figura 5.2: Sequências – Akiyo, Bridge-close, Bridge-far, Carphone.

Figura 5.3: Sequências – Grand-mother, Hall, Foreman, Container.

Figura 5.4: Sequências – Coastguard, Highway, Miss-america, Waterfall.

Figura 5.5: Sequências – Mobile, Mother-daughter, News, Salesman.

Figura 5.6: Sequências – Silent, Suzie, Bus, Flowers.

Figura 5.7: Sequências – Abstract, Concert, ArtAnt.

Figura 5.8: Sequências – Rafting, Fórmula 1, Chips.

Figura 5.9: Sequências – Ice, Rugby, Parkrun.

Figura 5.10: Sequências – Towers, Leaves, Letters.

Figura 5.11: Sequências – Football, Sea Wall, Bluesky.

Figura 5.12: Sequências – Pedestrian, Riverbed, Rush-hour.

Figura 5.13: Sequências – Station2, Sunflower, Tractor.

Figura 5.14: Sequências – Paris, Stefan, Tempete.

5.2.2 Observações sobre o Bitstream

As modificações realizadas sobre o decodificador presente no *software* de referência (JM versão 10.2) visaram possibilitar a coleta de dados sobre o processo de decodificação, formando um conjunto de amostras adequado para caracterizar o comportamento de alguns trechos específicos dos algoritmos do CABAD, através da análise de caso médio. A lista de informações coletadas é bastante extensa e foi organizada em categorias formadas a partir da combinação dos tipos de *slices*, tipos de macroblocos e tipos de SEs. A Figura 5.15 apresenta as quatro categorias de agrupamento criadas com os níveis de detalhamento de cada uma delas.

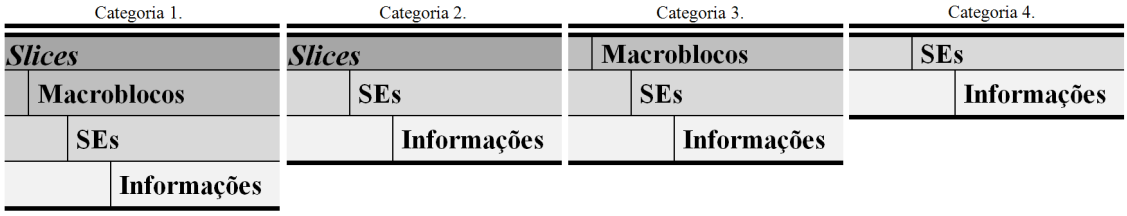


Figura 5.15: Categorias de agrupamento das informações coletadas a partir do fluxo de decodificação do CABAD.

A segmentação da coleta de informações de acordo com os tipos de *slices*, macroblocos e SEs foi adotada com o objetivo de mapear possíveis peculiaridades do fluxo de decodificação. Os experimentos realizados coletaram informações relacionadas com etapas internas do CABAD, as quais foram organizadas em cinco grupos, os quais serão discutidos na próxima seção. Contudo, além de informações sobre as etapas internas alguns dados adicionais foram produzidos como, por exemplo, o número de bits consumidos pelo CABAD em relação ao número de *bins* produzidos e a taxa de compressão do CABAC em relação ao CAVLC.

Para realizar a comparação entre as taxas de compressão dos dois métodos foi necessário repetir o processo de codificação de todas as sequências de vídeo, indicadas pela Tabela 5.3, porém, neste caso, trocando o método codificação de entropia de CABAC para CAVLC. Os resultados obtidos com as comparações entre as taxas de consumo e de produção do CABAD e a taxa de compressão do CABAC frente ao CAVLC foram separados por resolução, por sequência e por valor dos parâmetros de quantização. Foram criadas duas tabelas para apresentar esses resultados, sendo que na primeira, os dados são totalizados por sequência de vídeo codificada, utilizando uma média dos valores produzidos para os diferentes pares de parâmetro de quantização - esses dados podem ser observados na Tabela 5.4 – enquanto a segunda tabela apresenta resultados para os diferentes pares de parâmetros de quantização que foram produzidos a partir da soma dos valores gerados para cada sequência em cada par de QP.

Tabela 5.4: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução QCIF – resumido por sequência.

Sequências	Comparação entre Taxas de Compressão							
	CAVLC		CABAC					
	Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Entrada (MB)	Saída (MB)	Razão
Akiyo	0,78	89,23	0,70	90,24	9,39	0,70	0,87	1,24
Bridge-close	3,06	57,54	2,71	62,30	11,21	2,66	3,69	1,39
Bridge-far	3,04	57,74	2,67	62,90	12,21	2,66	3,51	1,32
Carphone	2,18	69,66	2,03	71,79	7,00	2,07	2,63	1,27
Claire	1,43	80,16	1,33	81,57	7,10	1,31	1,61	1,23
Coastguard	2,61	63,69	2,51	65,08	3,83	2,41	3,39	1,40
Container	1,71	76,19	1,53	78,77	10,83	1,53	1,91	1,25
Foreman	2,17	69,84	2,06	71,43	5,26	2,04	2,63	1,29
Grand-mother	1,94	73,02	1,76	75,60	9,56	1,76	2,17	1,24
Hall	2,47	65,67	2,31	67,86	6,36	2,30	2,93	1,27
Highway	3,00	58,33	2,67	62,90	10,95	2,66	3,57	1,34
Miss-america	1,57	71,43	1,46	73,51	7,27	1,46	1,80	1,24
Mobile	3,31	53,97	3,27	54,56	1,29	3,10	4,44	1,43
Mother-Daughter	1,77	75,40	1,64	77,18	7,26	1,55	2,01	1,30
News	1,21	83,20	1,16	83,93	4,33	1,25	1,47	1,18
Salesman	1,73	75,99	1,56	78,37	9,92	1,54	1,94	1,26
Silent	1,63	77,38	1,56	78,37	4,39	1,54	1,96	1,27
Suzie	1,51	72,47	1,40	74,55	7,55	1,40	1,74	1,24
Média	2,06	70,57	1,91	81,09	7,56	1,89	2,46	1,30
Maior valor	4,20	92,50	4,30	93,06	15,38	-	-	1,68
Menor valor	0,54	41,67	0,50	40,28	-2,44	-	-	0,92

A primeira coluna da Tabela 5.4 (da esquerda para direita) contém o título (nome) das diferentes sequências de vídeo na resolução QCIF que foram codificadas e comparadas. As demais colunas são organizadas em dois grupos sendo que um agrupa as colunas com os dados obtidos a partir da codificação utilizando o CAVLC como método de codificação de entropia, enquanto o segundo grupo apresenta resultados para o processo de codificação e decodificação das mesmas sequências de vídeo, porém utilizando o CABAD com o método de codificação de entropia. A segunda coluna apresenta o tamanho do arquivo produzido ao final do processo de codificação - saída do codificador utilizando o CAVLC - para cada uma das sequências. A terceira coluna traz o percentual de redução sobre o tamanho da amostra de entrada. A partir da quarta coluna temos as informações relacionadas ao CABAC e CABAD, sendo que a quarta coluna apresenta o tamanho do arquivo produzido ao final do processo de codificação - saída do codificador utilizando o CABAC. Na quinta coluna o percentual de redução

sobre o tamanho da amostra de entrada é ilustrado. A sexta coluna representa o ganho ou, em alguns casos, perda em termos da taxa de compressão do CABAC em relação ao CAVLC. As colunas seis, sete e oito são relacionadas aos dados produzidos pelo processo de decodificação (CABAD). A coluna seis apresenta o total de bits da sequência codificada que foram processados pelo CABAD, enquanto a coluna sete apresenta os bits produzidos pela saída dos MCAB. Por fim, a última coluna à esquerda da Tabela 5.4 apresenta a razão entre a taxa de bits consumidos na entrada e produzidos pela saída do CABAD, ou seja, representa a expansão do processo de decodificação.

As últimas três linhas da Tabela 5.4 apresentam a média dos resultados para codificação e decodificação das sequências de vídeo na resolução QCIF e os indicadores de maiores e menores valores para cada informação analisada. Além disso, cabe mencionar que os dados apresentados são uma média dos quatorze processos de codificação realizados para cada amostra, sendo sete processos com o CABAC e sete com o CAVLC. Os dados apresentados na Tabela 5.4 estão na unidade de megabytes (MB) ou em percentual (%), exceto a coluna oito.

A Tabela 5.5 traz o mesmo conjunto de informações apresentadas pela Tabela 5.4, porém aqui os dados são agrupados de acordo com o valor dos parâmetros de quantização, ao invés de serem agrupados por sequência de vídeo. A observação dos dados agrupados dessa forma permite visualizar a evolução das taxas de compressão e ganhos em razão das alterações nos parâmetros de QP.

Tabela 5.5: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução QCIF – resumido por QP.

Parâmetros Quantização		CAVLC		CABAC			
		Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Razão
QP0	Média	2,81	59,98	2,63	62,54	6,72	1,32
	Maior	4,20	84,72	4,30	86,53	11,90	1,51
	Menor	1,10	41,67	0,97	40,28	-2,38	1,22
QP6	Média	2,78	60,40	2,58	63,22	7,52	1,32
	Maior	4,10	84,72	4,20	86,67	13,04	1,49
	Menor	1,10	43,06	0,96	41,67	-2,44	1,22
QP12	Média	2,27	67,65	2,07	70,46	7,97	1,26
	Maior	3,60	88,47	3,40	89,44	14,29	1,35
	Menor	0,83	50,00	0,76	52,78	0,00	1,20
QP18	Média	1,88	73,12	1,72	75,41	8,13	1,24
	Maior	3,20	90,56	3,00	91,25	13,79	1,55
	Menor	0,68	55,56	0,63	58,33	0,00	0,93
QP24	Média	1,65	76,41	1,53	78,25	7,37	1,27
	Maior	2,90	91,53	2,80	92,08	14,29	1,68
	Menor	0,61	59,72	0,57	61,11	0,00	0,92
QP30	Média	1,56	77,77	1,44	79,44	7,50	1,28
	Maior	2,70	92,08	2,70	92,64	13,64	1,44
	Menor	0,57	62,50	0,53	62,50	0,00	1,15
QP36	Média	1,50	78,66	1,38	80,25	7,59	1,32
	Maior	2,50	92,50	2,50	93,06	15,38	1,52
	Menor	0,54	65,28	0,50	65,28	0,00	1,18
Geral	Média	2,06	70,57	1,91	81,09	7,56	1,30
	Maior	4,20	92,50	4,30	93,06	15,38	1,68
	Menor	0,54	41,67	0,50	40,28	-2,44	0,92

Observando a coluna seis da Tabela 5.5 é possível constatar que o ganho na taxa de compressão do CABAC sobre o CAVLC aumenta com os parâmetros quantização

iniciando em QP0 até QP18 e depois cai abruptamente no QP24, voltando a subir gradativamente até o QP36. Por outro lado, a razão entre a taxa de bits consumidos e os bits produzidos pelo CABAD (apresentado na coluna sete) tem um comportamento diferente. Essa razão começa alta no QP0 e vai reduzindo até QP18, onde começa novamente a subir até o QP36. Outra constatação interessante diz respeito ao menor e maior ganho possível, pois os relatos encontrados na literatura não fazem menção sobre casos onde o CABAC ofereça resultados inferiores ao CAVLC, porém nos experimentos realizados esse fato foi observado; tendo maior incidência onde os valores do parâmetro de quantização são menores (tipicamente QP0 e QP6).

Os processos de codificação e decodificação que foram aplicados sobre as sequências de vídeo na resolução QCIF também foram realizados com sequências de vídeo na resolução CIF. A Tabela 5.6 apresenta o mesmo conjunto de informações apresentado pela Tabela 5.4, porém os dados compilados nesta tabela são providentes dos processos de codificação e decodificação das sequências de vídeo na resolução CIF.

Tabela 5.6: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução CIF – resumido por sequência.

Sequências	Comparação entre Taxas de Compressão							
	CAVLC		CABAC					
	Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Entrada (MB)	Saída (MB)	Razão
Akiyo	12,56	56,70	11,86	59,11	5,57	11,27	16,71	1,48
Bridge-close	12,53	56,80	11,30	61,05	9,83	11,00	15,56	1,41
Bridge-far	8,31	62,21	8,07	63,31	2,93	7,86	10,90	1,39
Carphone	11,44	60,54	10,89	62,46	4,87	10,60	15,80	1,49
Claire	8,03	72,32	7,44	74,34	7,30	7,25	10,23	1,41
Coastguard	10,79	62,81	10,41	64,09	3,44	10,14	13,29	1,31
Container	9,54	67,09	9,10	68,62	4,63	8,86	11,81	1,33
Foreman	10,69	63,15	10,10	65,18	5,49	10,26	13,77	1,34
Grand-mother	11,89	59,01	11,01	62,04	7,38	11,36	15,29	1,35
Hall	13,09	54,88	12,99	55,22	0,76	11,90	18,14	1,52
Highway	7,56	73,94	6,89	76,26	8,88	6,69	8,81	1,32
Miss-america	5,00	82,76	4,73	83,69	5,43	5,10	5,97	1,17
Mobile	8,86	69,46	8,14	71,92	8,06	8,00	10,83	1,35
Mother-daughter	7,77	73,20	7,04	75,71	9,38	6,77	9,03	1,33
News	5,14	63,27	5,08	63,70	1,19	5,61	7,03	1,25
Salesman	11,91	58,92	11,73	59,54	1,52	11,23	16,00	1,42
Silent	9,99	65,57	9,37	67,70	6,19	9,59	11,94	1,25
Suzie	12,56	56,70	11,86	59,11	5,57	11,27	16,71	1,48
Média	9,71	64,95	9,18	66,85	5,42	9,03	12,42	1,38
Maior valor	17,00	87,24	18,00	87,93	14,93	-	-	2,94
Menor valor	3,70	41,38	3,50	37,93	-5,88	-	-	0,61

Comparando os dados apresentados na Tabela 5.4 contra os dados apresentados na Tabela 5.6 é possível perceber que a razão média desta última é superior a primeira (razão de 1,38 contra 1,30). Entretanto, isso não significa que o processo de compressão de vídeo do padrão H.264/AVC, utilizando o CABAC como método de codificação de entropia, seja mais eficiente quando aplicado a sequências de vídeo na resolução CIF do que sobre sequências de vídeo na resolução QCIF, pois a contagem dos bits de saída considera apenas os bits produzidos nos MCAB e não no valor dos SE restaurados. Além disso, observa-se que a variação entre a maior e a menor razão também é mais acentuada uma vez que a maior razão para a resolução CIF atinge 2,94 bits produzidos para cada bit consumido enquanto essa razão fica em 1,68 para resolução QCIF. Com a

menor razão ocorre o processo contrário, pois para resoluções QCIF fica em 0,92 bits produzidos por bit consumido enquanto para CIF é de apenas 0,61.

A Tabela 5.7 apresenta um conjunto de informações similar àquelas apresentadas pela Tabela 5.5, porém os dados produzidos foram obtidos a partir de sequências de vídeo na resolução CIF. Comparando as duas tabelas citadas é possível observar que o aumento da razão entre dados consumidos e produzidos pelo CABAD de acordo com o crescimento do QP, conforme observado na Tabela 5.5, não se repete. Contudo, a relação entre o aumento dos valores do QP com o aumento do ganho na taxa de compressão do CABAC sobre o CAVLC é mantida, entretanto em um patamar menor, já que o ganho médio, neste caso, é de 5,42% contra 7,56 da Tabela 5.5.

Tabela 5.7: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução CIF – resumido por QP.

Parâmetros Quantização		CAVLC		CABAC			
		Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Razão
QP0	Média	13,15	52,53	12,78	53,89	2,95	1,39
	Maior	17,00	76,55	18,00	77,59	10,91	1,50
	Menor	6,80	41,38	6,50	37,93	-5,88	1,23
QP6	Média	13,09	52,74	12,46	55,03	5,02	1,40
	Maior	17,00	76,55	17,00	77,93	12,73	1,53
	Menor	6,80	41,38	6,40	41,38	-1,47	1,28
QP12	Média	10,88	60,72	10,10	63,54	6,99	1,30
	Maior	15,00	81,03	14,00	82,41	14,93	1,41
	Menor	5,50	48,28	5,10	51,72	0,00	1,18
QP18	Média	9,04	67,37	8,41	69,63	6,77	1,30
	Maior	13,00	84,83	12,00	85,52	12,17	1,40
	Menor	4,40	55,17	4,20	58,62	0,00	1,21
QP24	Média	7,73	72,10	7,26	73,79	6,08	1,35
	Maior	11,00	86,21	10,90	87,24	9,89	1,46
	Menor	4,00	62,07	3,70	62,41	0,25	1,23
QP30	Média	7,10	74,37	6,63	76,09	6,57	1,51
	Maior	9,90	86,90	9,70	87,24	10,80	2,94
	Menor	3,80	65,86	3,67	66,55	2,02	0,61
QP36	Média	6,98	74,82	6,65	75,98	4,71	1,40
	Maior	9,70	87,24	9,30	87,93	9,08	1,58
	Menor	3,70	66,55	3,50	67,93	-1,58	1,26
Geral	Média	9,71	64,95	9,18	66,85	5,42	1,38
	Maior	17,00	87,24	18,00	87,93	14,93	2,94
	Menor	3,70	41,38	3,50	37,93	-5,88	0,61

As Tabela 5.8 e Tabela 5.9 trazem os dados obtidos a partir das sequências de vídeo na resolução D1. Na Tabela 5.8 é possível perceber que a razão média entre a taxa de bits consumidos e produzidos pelo CABAD cresceu em relação às sequencias de vídeo nas resoluções QCIF e CIF passando de 1,38 para 1,42. Em relação às diferenças entre as taxas de compressão do CABAC frente ao CAVLC, o ganho ainda se manifesta, porém em um percentual inferior ao observado para as resoluções QCIF e CIF visto que, neste caso, o ganho médio atinge apenas 3,93%.

Ao analisar os dados apresentados pela Tabela 5.9 é possível identificar que em todas as resoluções ocorreram casos onde a redução da taxa de compressão oferecia pelo CABAC foi inferior a redução oferecida pelo CAVLC, embora o resultado médio tenha sido favorável ao CABAC. Casos peculiares, como a sequência “Rugby” (ver Tabela 5.8), foram determinantes para os dados de caso médio tão “inesperado”.

A Tabela 5.10 e a Tabela 5.11 contém os dados obtidos a partir da análise das sequências de vídeo na resolução HD1080. Novamente a razão entre bits consumidos e produzidos pelo CABAC cresce em relação às resoluções de dimensões inferiores. Um fato que merece destaque é que para as sequências de vídeo analisadas, na resolução HD1080, o CABAC numca perde para o CAVLC em relação à taxa de compressão.

Tabela 5.8: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução D1 – resumido por sequência.

Sequências	Comparação entre Taxas de Compressão							
	CAVLC		CABAC					
	Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Entrada (MB)	Saída (MB)	Razão
Abstract	49,86	49,54	48,71	50,69	2,29	45,43	66,43	1,46
Artant	6,77	93,15	6,63	93,29	2,11	6,44	10,29	1,60
Chips	34,71	64,86	33,57	66,02	3,29	32,86	46,14	1,40
Concert	41,86	57,63	41,00	58,50	2,05	38,86	56,57	1,46
F1	41,57	57,92	39,83	59,69	4,19	39,43	61,14	1,55
Football	39,57	59,95	38,43	61,10	2,89	37,43	52,57	1,40
Ice	32,43	67,18	30,57	69,06	5,73	30,29	40,14	1,33
Leaves	9,14	90,75	8,50	91,40	7,03	8,50	10,76	1,27
Letters	16,29	83,52	15,49	84,33	4,91	15,43	20,86	1,35
Mobile	43,14	56,33	41,57	57,92	3,64	39,86	56,71	1,42
Parkrun	41,71	57,78	40,57	58,94	2,74	39,29	55,43	1,41
Rafting	38,86	60,67	38,29	61,25	1,47	36,43	53,14	1,46
Rugby	43,86	55,61	44,66	54,80	-1,83	41,29	62,86	1,52
Seawall	33,57	66,02	30,71	68,91	8,51	30,57	40,00	1,31
Suzie	29,86	69,78	27,29	72,38	8,61	27,29	35,00	1,28
Tempete	42,29	57,20	40,86	58,65	3,38	39,29	55,57	1,41
Towers	27,86	71,80	25,71	73,97	7,69	25,71	32,71	1,27
Waterfall	36,00	63,56	33,00	66,60	8,33	32,86	42,43	1,29
Média	33,85	65,74	32,52	67,08	3,93	31,51	44,38	1,41
Maior valor	65,00	95,04	65,00	95,14	11,76	-	-	1,75
Menor valor	4,90	34,21	4,80	34,21	-4,76	-	-	1,20

Tabela 5.9: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução D1 – resumido por QP.

Parâmetros Quantização		CAVLC		CABAC			
		Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Razão
QP0	Média	42,52	54,43	41,58	55,44	2,07	1,38
	Maior	65,00	91,50	65,00	91,09	9,52	1,75
	Menor	8,40	34,21	8,80	34,21	-4,76	1,25
QP6	Média	41,97	55,03	41,01	56,06	2,45	1,37
	Maior	64,00	91,50	64,00	91,50	8,70	1,75
	Menor	8,40	35,22	8,40	35,22	-2,98	1,25
QP12	Média	35,45	62,01	33,46	64,15	5,38	1,30
	Maior	56,00	92,41	52,00	92,71	10,53	1,53
	Menor	7,50	43,32	7,20	47,37	0,20	1,25
QP18	Média	29,71	68,16	28,12	69,86	5,47	1,27
	Maior	48,00	93,22	45,00	93,52	11,76	1,51
	Menor	6,70	51,42	6,40	54,45	-2,44	1,24
QP24	Média	26,07	72,06	24,77	73,45	5,14	1,28
	Maior	42,00	93,83	40,00	94,23	9,09	1,50
	Menor	6,10	57,49	5,70	59,51	0,00	1,20
QP30	Média	24,12	74,15	23,23	75,10	4,10	1,32
	Maior	38,00	94,53	38,00	94,84	9,52	1,55
	Menor	5,40	61,54	5,10	61,54	-2,42	1,25
QP36	Média	23,13	75,21	22,64	75,73	2,60	1,35
	Maior	36,00	95,04	37,00	95,14	9,09	1,65
	Menor	4,90	63,56	4,80	62,55	-2,81	1,26
Geral	Média	33,85	65,74	32,52	67,08	3,93	1,41
	Maior	65,00	95,04	65,00	95,14	11,76	1,75
	Menor	4,90	34,21	4,80	34,21	-4,76	1,20

Tabela 5.10: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução HD1080 – resumido por sequência.

Sequências	Comparação entre Taxas de Compressão							
	CAVLC		CABAC					
	Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Entrada (MB)	Saída (MB)	Razão
Blue_sky	248,86	58,37	236,57	60,43	4,94	225,43	337,57	1,50
Pedestrian	214,14	64,18	196,43	67,14	8,27	194,86	271,14	1,39
Riverbed	257,86	56,87	251,86	57,87	2,33	236,29	348,00	1,47
Rush_hour	206,29	65,49	187,86	68,58	8,93	187,86	253,14	1,35
Station2	236,57	60,43	221,00	63,03	6,58	215,29	305,71	1,42
Sunflower	220,71	63,08	200,00	66,54	9,39	198,43	275,71	1,39
Tractor	262,43	56,10	254,86	57,37	2,89	238,71	352,29	1,48
Média	235,27	60,64	221,22	62,99	5,97	213,84	306,22	1,43
Maior valor	345,00	75,41	344,00	77,75	11,05	-	-	1,54
Menor valor	147,00	42,29	133,00	42,46	0,29	-	-	1,33

Tabela 5.11: Comparação entre as taxas de compressão obtidas pelo padrão H.264/AVC para codificar sequências de vídeo na resolução HD1080 – resumido por QP.

Parâmetros Quantização		CAVLC		CABAC			
		Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Razão
QP0	Média	312,71	47,7	300,14	49,79	4,25	1,45
	Maior	345,00	53,0	344,00	56,84	8,19	1,54
	Menor	281,00	42,3	258,00	42,46	0,29	1,36
QP6	Média	309,86	48,17	297,14	50,29	4,33	1,45
	Maior	342,00	53,50	340,00	57,18	7,91	1,54
	Menor	278,00	42,79	256,00	43,12	0,30	1,35
QP12	Média	261,71	56,22	243,43	59,28	7,16	1,40
	Maior	291,00	61,36	277,00	64,87	9,88	1,48
	Menor	231,00	51,32	210,00	53,66	4,18	1,33
QP18	Média	219,71	63,25	203,29	65,99	7,67	1,40
	Maior	246,00	68,22	235,00	71,23	10,40	1,45
	Menor	190,00	58,85	172,00	60,69	4,12	1,34
QP24	Média	189,71	68,26	175,71	70,61	7,60	1,41
	Maior	215,00	73,07	205,00	75,58	10,73	1,46
	Menor	161,00	64,03	146,00	65,71	3,33	1,34
QP30	Média	182,86	69,41	169,43	71,66	7,59	1,42
	Maior	207,00	73,90	198,00	76,58	11,05	1,47
	Menor	156,00	65,37	140,00	66,88	2,97	1,36
QP36	Média	170,29	71,51	159,43	73,33	6,61	1,45
	Maior	191,00	75,41	185,00	77,75	9,88	1,50
	Menor	147,00	68,05	133,00	69,05	2,69	1,36
Geral	Média	235,27	60,64	221,22	62,99	5,97	1,43
	Maior	345,00	75,41	344,00	77,75	11,05	1,54
	Menor	147,00	42,29	133,00	42,46	0,29	1,33

Um diferencial nos dados coletados sobre as sequências na resolução HD1080 é que nenhuma das sequências utilizadas em nenhuma combinação de valores de QP apresentou taxa de compressão obtida pelo CABAC inferior à taxa obtida pelo CAVLC. Esse dado reforça a importância de utilizar o CABAC na codificação de sequências com maior resolução, pois nestes casos mesmo um ganho percentual pequeno pode representar economia de muitos Mbps.

Os resultados de caso médio para cada uma das resoluções analisadas entre as Tabelas 5.5 e 511 são apresentados na Tabela 5.12.

Tabela 5.12: Resultados da avaliação para cada uma das resoluções analisadas.

Resolução		CAVLC		CABAC			
		Tamanho (MB)	Redução (%)	Tamanho (MB)	Redução (%)	Ganho (%)	Razão
QCIF	Média	2,06	70,57	1,91	81,09	7,56	1,30
	Maior	4,20	92,50	4,30	93,06	15,38	1,68
	Menor	0,54	41,67	0,50	40,28	-2,44	0,92
CIF	Média	9,71	64,95	9,18	66,85	5,42	1,38
	Maior	17,00	87,24	18,00	87,93	14,93	2,94
	Menor	3,70	41,38	3,50	37,93	-5,88	0,61
D1	Média	33,85	65,74	32,52	67,08	3,93	1,41
	Maior	65,00	95,04	65,00	95,14	11,76	1,75
	Menor	4,90	34,21	4,80	34,21	-4,76	1,20
HD1080	Média	235,27	60,64	221,22	62,99	5,97	1,43
	Maior	345,00	75,41	344,00	77,75	11,05	1,54
	Menor	147,00	42,29	133,00	42,46	0,29	1,33

5.2.3 Avaliação do Processo de Decodificação

Esta seção contempla a análise do processo de decodificação baseada nas relações estabelecidas entre subalgoritmos que compõem o CABAD. Essas relações foram organizadas em quatro grupos de informações, sendo: I – taxa de ocorrência de cada tipo de SE e dos *bins* produzidos por eles; II – taxa de utilização de cada MCAB; III – produção de *bins* consecutivos pelo mesmo MCAB; IV – comprimento do *binstring* e valores mínimos e máximos dos SEs.

A Tabela 5.13 apresenta a nomenclatura adotada para os SEs, indicando seu significado. Os SEs ora adotados diferem daqueles apresentados em (ITU-T, 2003), pois alguns SEs foram agrupados. Os coeficientes de transformadas quantizados foram agrupados apenas pelo tipo de componente (cor/luminosidade) ao invés de serem separados em função do tipo de nível (AC/DC) e do processo de predição pelo qual foram gerados (Intra/Inter). Nos MVD optou-se por agrupar as componentes X e Y em um mesmo elemento, fazendo distinção apenas entre as listas 0 e 1. Enquanto os índices para quadros de referência foram todos agrupados em um único SE a despeito da lista.

Tabela 5.13: Nomenclatura adotada para os diversos tipos de SEs.

TIPO DE SE	SIGNIFICADO DA NOMENCLATURA ADOTADA
MBTYPE	Tipo de macrobloco
REFIDX	Índices para os frames de referência (Lista 0 e Lista 1)
CBP	Padrão de Codificação dos Blocos 8X8 (4 blocos para luminância e 2 blocos para crominância)
COEF_LUM	Coeficientes de transformada quantizados gerados pelos resíduos da predição (amostras de luminância)
COEF_CHR	Coeficientes de transformada quantizados gerados pelos resíduos da predição (amostras de crominância)
QP_DELTA	Variação do parâmetro de quantização no nível de macrobloco
COEF_SIG	Conjuntos de flags que compõem o mapa de significância (um flag para cada coeficiente de bloco 4x4)
COEF_LAS	Conjuntos de flags que compõem o mapa de significância (um flag para cada coeficiente significante)
CBF	Padrão de codificação dos blocos 4x4 dentro de uma partição 8X8 do macrobloco
INTRA_FLAG	Flag para indicar variação entre o modo de codificação do bloco Intra atual em relação ao anterior
INTRA_MODE	Modo de codificação do Intraquadros empregado pelo bloco de predição
SUB_MBTYPE	Tipo de sub-macrabloco, ou seja, forma como cada partição de um macrobloco é organizada
MVD_L0	Componentes X e Y para vetor de movimento diferencial relacionados a lista 0
MVD_L1	Componentes X e Y para vetor de movimento diferencial relacionados a lista 1

Os dados que serviram de subsídio para a avaliação do processo de decodificação foram obtidos através de rotinas de extração e/ou coleta introduzidas no JM10.2 – conforme descrito anteriormente – as quais foram acionadas durante as etapas de decodificação realizadas para cada uma das amostras de vídeo indicadas na Tabela 5.3. Esses dados foram sumarizados em arquivos sendo que estes, por sua vez, passaram por um processamento posterior – um *parser* desenvolvido especialmente para essa tarefa – que sumarizou os dados de todas as sequências para cada uma das resoluções consideradas. Esses dados sumarizados serão apresentados a seguir.

A análise realizada foi segmentada de acordo com os tipos de SEs apresentados pela Tabela 5.13, enquanto os parâmetros avaliados foram selecionados com base nos quatro grupos citados no início desta seção.

O primeiro grupo aborda a taxa de ocorrência de cada tipo de SE e dos *bins* produzidos por eles e, para isso, utiliza três informações: primeiro o percentual de ocorrências de cada tipo de SE em relação a todos os elementos produzidos, indicado pela coluna "(%) BINS"; depois, o percentual de *bins* produzidos pelo processo de decodificação aritmética para cada tipo de SE processado em relação ao total de *bins*

produzidos para todos SEs, indicado pela coluna “BINS (%)”; e, por fim, o número médio de *bins* produzidos para cada tipo de SE, indicado pela coluna “BINS/SE”.

O segundo grupo trata da taxa de utilização dos módulos de codificação aritmética através de duas informações: o percentual dos *bins* produzidos pelo módulo de codificação regular para um determinado tipo de SE em relação ao total de *bins* produzidos para o mesmo tipo de SE, indicado pela coluna “Regular (%)”; e o percentual de *bins* produzidos pelo módulo bypass para um determinado tipo de SE em relação ao total de *bins* produzidos pelo mesmo tipo de SE, indicado pela coluna “Bypass (%)”.

O terceiro grupo aborda a incidência de *bins* consecutivos sendo gerados pelo mesmo módulo de codificação aritmética, ou seja, sem alteração do módulo de codificação aritmética entre *bins* subsequentes. Para tanto, apresenta duas informações: o percentual de incidência de *bins* consecutivos no módulo regular, indicado pela coluna “Regular Consec.”; e o percentual de incidência de *bins* consecutivos produzidos pelo módulo bypass, indicado pela coluna “Bypass Consec.”.

O quarto grupo contempla informações relacionadas com os limites do *bitstream* através de três informações: o mínimo valor produzido por cada tipo de SE, indicado pela coluna “VMÍN”; o máximo valor produzido por cada tipo de SE, indicado pela coluna “VMÁX”; e o tamanho máximo da sequência de *bins* produzidos para representar o valor de cada tipo de SE, indicado pela coluna “BMÁX”.

As análises foram realizadas com base nos tipos de SE definidos pela Tabela 5.13 e nos grupos de informações descritos acima. Assim, todos os dados produzidos por cada uma das sequências de vídeo descritas na Tabela 5.3, em todas as variações de parâmetros de quantização, foram agrupados e sumarizados de acordo com suas resoluções. Dessa forma, foram geradas quatro tabelas, sendo uma para cada resolução avaliada QCIF, CIF, D1 e HD1080. Esse agrupamento dos dados produzidos por todas as sequências de uma mesma resolução com as variações de parâmetros de quantização teve por objetivos simplificar a quantidade de dados a serem apresentados ao mesmo tempo em que fornece uma amostra mais genérica, visto que os dados apresentados são uma média de tudo que foi produzido nos diversos cenários de avaliação.

A Tabela 5.14 contempla informações coletadas para as sequências de vídeo na resolução QCIF. Nela é possível observar que os SE dos tipos COEF_LUM, COEF_CHR, COEF_SIG e COEF_LAS juntos somam mais de 94% de todas as ocorrências dos SEs, assim como de *bins* produzidos, porém cabe salientar que os *bins* produzidos pelos SEs COEF_LUM e COEF_CHR representam mais que 57% do total de *bins,* enquanto a ocorrência destes SEs em relação ao total de SEs que ocorrem é de pouco mais de 40%; por outro lado, os *bins* produzidos pelos SEs COEF_SIG e COEF_LAS representam, aproximadamente, 36% do total frente uma representação de mais de 54% em termos de ocorrência de SEs. Essa constatação comprova as afirmações de (YU HE, 2005) sobre alguns SE ocorrerem muitas vezes, porém geram poucos *bins* a cada ocorrência, enquanto outros geram muitos *bins* a cada ocorrência.

Tabela 5.14: Resultados da avaliação para sequências com resolução QCIF.

Tipo de SE	Grupo I			Grupo II		Grupo III		Grupo IV		
	SE (%)	BINS (%)	BINS/SE	Regular (%)	Bypass (%)	Regular Consec.	Bypass Consec.	VMÍN	VMÁX	BMÁX
MBTYPE	0,13	0,35	4,18	99,67		98,73		0	48	14
REFIDX	0,21	0,20	1,41	100,00		100,00		0	3	4
CBP	0,12	0,49	5,94	99,63		95,39		0	47	6
COEF_LUM	27,5	42,48	2,32	71,80	28,19	69,97	13,94	-1059	998	36
COEF_CHR	12,8	15,43	1,81	69,99	29,57	67,06	7,32	-1484	1757	36
QP_DELTA	0,12	0,08	1,00	100,00		100,00		0	0	1
COEF_SIG	33,7	22,44	1,00	100,00				0	1	1
COEF_LAS	20,6	13,76	1,00	100,00				0	1	1
CBF	2,91	1,95	1,01	99,46				0	1	2
INTRA_FLAG	0,39	0,26	1,00	100,00				-1	1	1
INTRA_MODE	0,29	0,56	2,94	100,00		100,00		0	7	3
SUB_MBTYPE	0,23	0,41	2,69	100,00		100,00		0	12	6
MVD_L0	0,73	1,34	2,76	75,85	24,15	80,77	32,83	-134	131	22
MVD_L1	0,14	0,25	2,60	76,89	23,11	78,30	26,82	-127	145	22

Os dados da Tabela 5.14 mostram que a taxa de utilização da codificação aritmética executada pelo módulo regular atinge 83% do total de *bins* produzidos. Essa informação confirma as indicações encontradas na literatura sobre a maior importância deste módulo de codificação em relação aos módulos bypass (com apenas 16%) e o terminate (com apenas 0,8%). Além disso, é possível observar que apenas os SEs dos tipos COEF_LUM, COEF_CHR, MVD_L0 e MVD_L1 fazem uso do módulo de codificação bypass. Dada a pequena representatividade do módulo *terminate* os dados de percentagem correspondentes a ele foram obtidos do grupo II das tabelas de resultados, porém podem facilmente ser deduzidos através da diferença entre 100% e a soma das duas colunas apresentadas no grupo II.

O grupo III presente da Tabela 5.14 apresenta uma análise sobre a incidência de ocorrências consecutivas de decodificação através dos módulos regular ou bypass. Neste grupo, é possível observar que as linhas correspondentes aos SEs dos tipos COEF_SIG, COEF_LAS, CBF e INTRA_FLAG não possuem indicação do percentual em nenhuma das duas colunas, pois esses SEs são flag que, quando ocorrem, produzem apenas um *bin* sendo o motivo pelo qual não faz sentido analisar sua ocorrência consecutiva entre os módulos de codificação aritmética. Neste grupo, merecem destaques os SEs dos tipos COEF_LUM, COEF_CHR, MVD_L0 e MVD_L1 para os quais a ocorrência de codificações consecutivas através do mesmo módulo de codificação aritmética é comum, tanto no módulo regular quanto no módulo bypass.

Os limites de tamanho de *binstring* produzidos por cada tipo de SE são apresentados no grupo IV da Tabela 5.14. Novamente, os SEs dos tipos COEF_LUM, COEF_CHR, MVD_L0 e MVD_L1 merecem destaque, pois são esses elementos que produzem as maiores cadeias de *binstring*, sendo 36 para os dois primeiros e 22 para os dois últimos.

A Tabela 5.15, a Tabela 5.16 e a Tabela 5.17 apresentam os mesmos conjuntos de informações encontradas na Tabela 5.14, porém relacionadas às sequências de vídeo nas resoluções CIF, D1 e HD1080, respectivamente.

Tabela 5.15: Resultados da avaliação para sequências com resolução CIF.

Tipo de SE	Grupo I			Grupo II		Grupo III		Grupo IV		
	SE (%)	BINS (%)	BINS/SE	Regular (%)	Bypass (%)	Regular Consec.	Bypass Consec.	VMÍN	VMÁX	BMÁX
MBTYPE	0,12	0,29	4,23	99,57		98,69		0	48	14
REFIDX	0,19	0,15	1,33	100,00		100,00		0	3	4
CBP	0,12	0,40	5,95	99,79		95,21		0	47	6
COEF_LUM	26,98	49,38	3,18	74,18	25,81	75,66	22,43	-1415	3200	38
COEF_CHR	12,51	13,74	1,91	71,04	28,55	68,41	6,99	-1030	1065	36
QP_DELTA	0,12	0,07	1,00	100,00		100,00		0	0	1
COEF_SIG	32,89	18,91	1,00	100,00				0	1	1
COEF_LAS	22,43	12,90	1,00	100,00				0	1	1
CBF	2,82	1,63	1,01	99,50				0	1	2
INTRA_FLAG	0,46	0,27	1,00	100,00				-1	1	1
INTRA_MODE	0,36	0,60	2,94	100,00		100,00		0	7	3
SUB_MBTYPE	0,20	0,31	2,67	100,00		100,00		0	12	6
MVD_L0	0,64	1,11	2,98	73,43	26,57	80,93	41,03	-234	189	22
MVD_L1	0,15	0,24	2,75	74,79	25,21	77,84	33,19	-184	132	22

Comparando a Tabela 5.15 com a Tabela 5.14 é possível observar um crescimento significativo na ocorrência de *bins* consecutivos para o módulo bypass. Esse fato ocorre em decorrência do aumento da resolução das sequências de vídeo, pois maior é a tendência dos dados residuais e MVDs aumentarem; com isso, o sufixo do *binstring* destes SE, que é gerado através do módulo bypass, sofre maior carga. Outro indício dessa afirmação é o tamanho máximo do *binstring* para o SE do tipo COEF_LUM que passou de 36 na Tabela 5.14 para 38 na Tabela 5.15, assim como os próprios valores dos SEs dos tipos COEF_LUM e MVD_L0 que utilizaram uma faixa mais ampla.

Tabela 5.16: Resultados da avaliação para sequências com resolução D1.

Tipo de SE	Grupo I			Grupo II		Grupo III		Grupo IV		
	SE (%)	BINS (%)	BINS/SE	Regular (%)	Bypass (%)	Regular Consec.	Bypass Consec.	VMÍN	VMÁX	BMÁX
MBTYPE	0,12	0,28	4,36	99,38		98,48		0	48	14
REFIDX	0,15	0,11	1,37	100,00		100,00		0	3	4
CBP	0,12	0,37	5,94	99,58		95,07		0	47	6
COEF_LUM	26,84	52,84	3,80	74,20	25,79	78,98	33,16	-2867	1321	39
COEF_CHR	12,14	13,23	2,11	70,57	29,05	69,12	12,10	-1421	1384	36
QP_DELTA	0,12	0,06	1,00	100,00		100,00		0	0	1
COEF_SIG	32,63	16,89	1,00	100,00				0	1	1
COEF_LAS	22,84	11,82	1,00	100,00				0	1	1
CBF	2,82	1,47	1,01	99,45				0	1	2
INTRA_FLAG	0,82	0,42	1,00	100,00				-1	1	1
INTRA_MODE	0,59	0,90	2,92	100,00		100,00		0	7	3
SUB_MBTYPE	0,16	0,23	2,86	100,00		100,00		0	12	6
MVD_L0	0,51	1,13	4,23	69,49	30,51	82,74	55,68	-514	552	26
MVD_L1	0,13	0,26	3,89	70,90	29,10	80,05	50,43	-399	290	24

Comparando a Tabela 5.16 com a Tabela 5.15 e com a Tabela 5.14, é possível observar que, novamente, houve crescimento na taxa de ocorrência de *bins* consecutivos para o módulo bypass. Assim, com observado na comparação entre a Tabela 5.15 e a Tabela 5.14, o tamanho máximo do *binstring* para o SE do tipo COEF_LUM aumentou, passando de 38 para 39 enquanto os SEs dos tipos MVD_L0 e MVD_L1 passaram para 26 e 24, respectivamente.

Tabela 5.17: Resultados da avaliação para sequências com resolução HD1080.

Tipo de SE	Grupo I			Grupo II		Grupo III		Grupo IV		
	SE (%)	BINS (%)	BINS/SE	Regular (%)	Bypass (%)	Regular Consec.	Bypass Consec.	VMÍN	VMÁX	BMÁX
MBTYPE	0,12	0,30	4,78	98,50		97,73		0	48	14
REFIDX	0,02	0,01	1,15	100,00		100,00		0	3	4
CBP	0,11	0,35	5,99	99,99		99,00		0	47	6
COEF_LUM	26,43	38,20	2,79	73,23	26,77	75,60	24,89	-2508	2713	38
COEF_CHR	13,69	28,82	4,06	70,53	29,27	82,33	50,79	-2133	2083	38
QP_DELTA	0,12	0,06	1,00	100,00		100,00		0	0	1
COEF_SIG	31,49	16,31	1,00	100,00				0	1	1
COEF_LAS	21,90	11,35	1,00	100,00				0	1	1
CBF	2,98	1,55	1,00	99,66				0	1	2
INTRA_FLAG	1,73	0,89	1,00	100,00				-1	1	1
INTRA_MODE	1,31	1,94	2,87	100,00		100,00		0	7	3
SUB_MBTYPE	0,01	0,02	3,12	100,00		100,00		0	12	6
MVD_L0	0,05	0,11	4,59	65,70	34,30	83,73	60,46	-284	512	24
MVD_L1	0,03	0,07	4,71	65,97	34,03	79,39	59,09	-210	176	22

Por fim a Tabela 5.17 mantém a tendência apresentada nas anteriores, visto que a taxa de ocorrência de *bins* consecutivos no módulo bypass quase dobrou entre a Tabela 5.14 (resolução baixa – QCIF) e a Tabela 5.17 (resolução alta – HD1080).

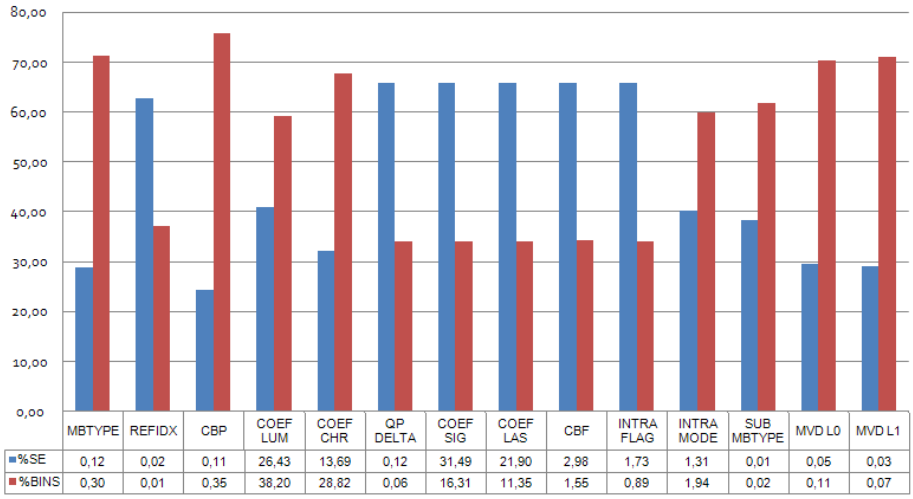
A análise dos dados apresentados entre as Tabela 5.14 e Tabela 5.17 demonstra que os SEs dos tipos COEF_LUM, COEF_CHR, COEF_SIG, COEF_LAS e CBF representam mais de 96% da taxa de ocorrência dos SE, ou seja, quase toda informação processada pelo CABAD é oriunda de dados residuais sendo, aproximadamente, 56% destinada ao mapa de significância – dado pelos SEs dos tipos COEF_SIG, COEF_LAS e CBF – e valor dos coeficientes de transformada – dados pelos SEs dos tipos COEF_LUM e COEF_CHR – respondem por 40%. Entretanto, ao analisarmos os *bins* processados por cada tipo de SE, é possível identificar que 67% do total de *bins* são provenientes do valor dos coeficientes de transformada, enquanto o mapa de significância representa apenas 29%. Essas variações entre o percentual de ocorrências de cada SE e o percentual de *bins* produzidos pelo SE em relação a produção total de *bins* pode ser observada na Figura 5.16.

A taxa de processamento nos módulos de codificação aritmética mostra que, em média, o módulo regular é responsável por 80% dos *bins* produzidos, enquanto o módulo bypass produz apenas 19,8% dos *bins*. Essa relação varia de acordo com as resoluções de vídeo, com tendência de redução na diferença entre as taxas dos dois módulos para as sequências de vídeo de maior resolução, conforme pode ser observado na Figura 5.17.

A produção de *bins* consecutivos através do mesmo módulo de codificação aritmética dentro de um mesmo SE é interessante, pois abre caminho para a exploração do espaço de projeto em termos de arquitetura com taxa de processamento variável com possibilidade de produzir mais de um *bin* por ciclo. Considerando o tamanho médio dos SEs e o tamanho máximo do *binstring* produzido por esses SEs, surge à necessidade de aprofundar a análise sobre a utilização subsequente do mesmo módulo de codificação aritmética.

Para avaliar o impacto de incidências consecutivas com número de repetições superior a duas, foi necessário coletar informações adicionais. Dado o volume de informações essa análise em particular foi restrita aos SE dos tipos COEF_LUM,

COEF_CHR, MVD_L0 e MVD_L1, os quais fazem uso do módulo Bypass de codificação aritmética. A análise foi restringida ao módulo Bypass visto que a complexidade e limitações impostas pelo módulo Regular dificultariam a adoção de uma solução arquitetural com vários destes módulos aninhados.



	MBTYPE	REFIDX	CBP	COEF LUM	COEF CHR	QP DELTA	COEF SIG	COEF LAS	CBF	INTRA FLAG	INTRA MODE	SUB MBTYPE	MVD L0	MVD L1
%SE	0,12	0,02	0,11	26,43	13,69	0,12	31,49	21,90	2,98	1,73	1,31	0,01	0,05	0,03
%BINS	0,30	0,01	0,35	38,20	28,82	0,06	16,31	11,35	1,55	0,89	1,94	0,02	0,11	0,07

Figura 5.16: Relação entre o percentual de ocorrência de um SE e o percentual total de *bins* produzidos pelo mesmo SE – ambos em relação ao total geral.

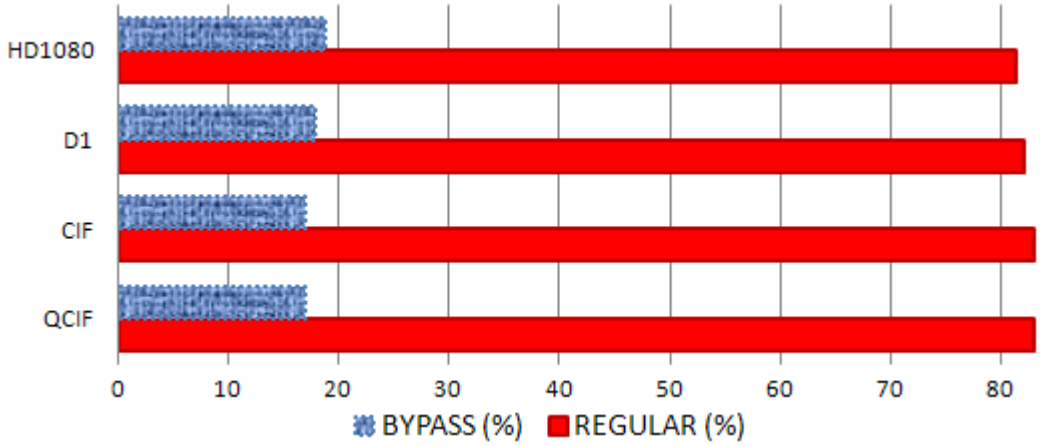


Figura 5.17: Relação entre as taxas de processamento nos módulos de codificação aritmética, Regular X Bypass, para as diferentes resoluções de vídeo utilizadas.

Os dados apresentados pela Tabela 5.18 resumem o processo de análise sobre as ocorrências consecutivas no módulo Bypass. Ela está segmentada por resolução e tipo de SE analisado. A coluna intitulada "BINS (%)" indica a taxa representativa dos *bins* produzidos por cada SE em relação ao total de *bins* produzidos por todos os SEs. A coluna intitulada "BINS no Bypass (%)" indica a taxa representativa dos *bins* produzidos por cada SE através do módulo Bypass em relação ao total de *bins* produzidos por todos os SE em todos os módulos de codificação aritmética. A coluna intitulada "Consec. (%)" apresenta o percentual dos *bins* produzidos pelo módulo Bypass que são processados de forma consecutiva. A coluna subsequente expressa o percentual de *bins* que podem ser produzidos de forma consecutiva através do módulo bypass em relação ao total de *bins*. Por fim, a última consulta traz o percentual de *bins* que podem ser produzidos de forma consecutiva, tendo mais que ocorrências subsequentes.

Tabela 5.18: Análise da produção de *bins* pelo módulo de codificação Bypass de forma consecutiva por mais que duas repetições.

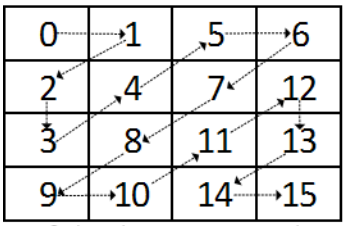
Resolução	Tipo de SE	BINS (%)	BYPASS			
			BINS no Bypass (%)	Consec. (%)	BINS Consec (%)	>2 *bins* Consec (%)
HD1080	COEF_LUM	38,20	10,23	24,89	2,54	1,65
	COEF_CHR	28,82	8,44	50,79	4,29	3,53
	MVD_L0	0,11	0,04	60,46	0,02	0,02
	MVD_L1	0,07	0,02	59,09	0,01	0,01
	Sub-Total	67,20	18,73	-	6,87	5,22
D1	COEF_LUM	52,84	13,63	33,16	4,52	2,93
	COEF_CHR	13,23	3,84	12,10	0,46	0,38
	MVD_L0	1,13	0,34	55,68	0,19	0,18
	MVD_L1	0,26	0,08	50,43	0,04	0,04
	Sub-Total	67,46	17,89	-	5,21	3,53
CIF	COEF_LUM	49,38	12,75	22,43	2,86	1,86
	COEF_CHR	13,74	3,92	6,99	0,27	0,23
	MVD_L0	1,11	0,29	41,03	0,12	0,11
	MVD_L1	0,24	0,06	33,19	0,02	0,02
	Sub-Total	64,47	17,02	-	3,27	2,21
QCIF	COEF_LUM	42,48	11,98	13,94	1,67	1,08
	COEF_CHR	15,43	4,56	7,32	0,33	0,28
	MVD_L0	1,34	0,32	32,83	0,11	0,10
	MVD_L1	0,25	0,06	26,82	0,02	0,01
	Sub-Total	59,50	16,92	-	2,13	1,47

Os totais apresentados na Tabela 5.18, ao final de cada resolução, permitem visualizar o potencial de produção de *bins* de forma simultânea através do módulo Bypass. Deve-se destacar que os ganhos mais significativos são obtidos pelas maiores resoluções, atingindo até 5,22% do total de *bins* produzidos para a resolução HD1080.

Outro aspecto analisado foi a incidência dos SEs dos tipos COEF_SIG e COEF_FLAG, pois, de acordo com os dados apresentados pelas Tabelas 5.14, 5.15, 5.16 e 5.17, o número de *bins* produzido para esses dois SEs corresponde a, aproximadamente, 30% (em média) do total de bins produzidos. Dessa forma, foi desenvolvido um estudo sobre o comportamento do processo de decodificação para estes dois SEs. A Figura 5.18 é composta por três imagens (identificadas como “a”, “b” e “c”) que visam apresentar um resumo do processo de formação do mapa de significância para o um bloco 4x4 com valores dos coeficientes de transformada.

A Figura 5.18a ilustra um bloco 4x4 com as setas indicando a ordem de processamento dos coeificientes. Na Figura 5.18b são apresentados (como exemplo) valores de coeficientes para um bloco 4x4. Na Figura 5.18c a formação do mapa de significância para o bloco 4x4 de exemplo é apresentado.

Através da Figura 5.18c é possível observar que são gerados dois *flags* para cada coeficiente do bloco 4x4. O primeiro *flag* corresponde ao SE do tipo COEF_SIG que indica se um dado coeificiente possui valor diferente de zero. O segundo *flag* corresponde ao SE do tipo COEF_LAST e indica se o coeficiente é último coeficiente com valor diferente do zero para o bloco 4x4 (na ordem de processamento Zig-Zag). Esses pares de valores ocorrem até que o último coeficiente com valor diferente de zero seja processado. Entretanto, se algum coeficiente antes do último válido possuir valor igual a zero, não será gerado o SE do tipo COEF_LAST para esse coeficiente.



a. Ordem de processamento dos blocos 4x4.

24	2	1	0
0	1	0	0
-1	0	0	0
3	0	0	0

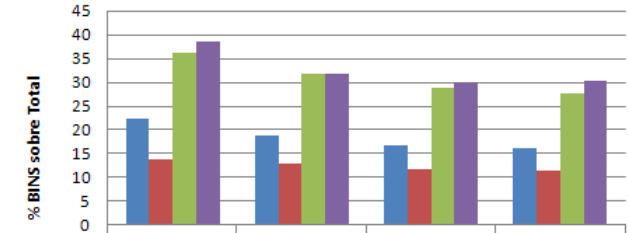
b. Exemplo de valores para os coeficientes de um bloco 4x4.

	Coeficientes processados na ordem do Zig-Zag															
Índice	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valor	24	2	0	-1	1	1	0	0	0	3	0	0	0	0	0	0
Flag SIG	1	1	0	1	1	1	0	0	0	1						
Flag LAST	0	0		0	0	0				1						

c. Geração do mapa de significância para os valores de um bloco 4x4.

Figura 5.18: Formação do mapa de significância a partir dos valores de coeficientes para um bloco 4x4.

A Figura 5.19 apresenta a ocorrência média de *bins* para estes dois SEs em relação ao total de *bins* para as diferentes resoluções analisadas. Além disso, apresenta a diferença percentual entre as ocorrências de *bins* do SE do tipo COEF_SIG para o SE do tipo COEF_LAST. Essa informação está relacionada com o fato dos coefientes com valor igual a zero não produzirem um SE do tipo COEF_LAST e pode ser útil para definição de uma estratégia de controle para acelerar a decodificação deste tipo de SE.



	QCIF	CIF	D1	HD1080
COEF_SIG	22,44	18,91	16,89	16,31
COEF_LAS	13,76	12,90	11,82	11,35
Total	36,21	31,81	28,71	27,66
Diff.	38,67	31,80	29,99	30,44

Figura 5.19: Relações de incidência de *bins* para os SEs COEF_SIG e COEF_LAST.

Os dados e observações coletados durante o processo de análise de comportamento estático e dinâmico, discutidos neste capítulo, forneceram o embasamento para as decisões arquiteturais que guiaram o desenvolvimento da arquitetura de *hardware* para o CABAD que será apresentada no próximo capítulo.

6 ARQUITETURA DO CABAD EM *HARDWARE*

Este capítulo tem por objetivo descrever a arquitetura de *hardware* desenvolvida durante a realização deste trabalho, bem como o seu respectivo processo de validação. Por se tratar de um *hardware* dedicado ao CABAD a definição das funcionalidades do projeto, bem como suas restrições, foram pautadas na seção 9.3 da norma de especificação do padrão H.264/AVC (ITU-T, 2003), considerando o perfil "main" no nível 4.0. Em linhas gerais, o funcionamento do subconjunto de ferramentas que compõem o CABAD foi apresentado ao longo do Capítulo 4, no qual a complexidade e extensão dos seus algoritmos foram discutidas. Contudo, as definições para a elaboração da arquitetura desenvolvida também consideram os relatos de outras soluções encontradas na literatura e, principalmente, o processo de análise de comportamentos estáticos e dinâmicos com dados obtidos a partir da codificação e decodificação de uma vasta quantidade de sequências de vídeo padronizadas; conforme apresentado no Capítulo 5.

Para detalhar a arquitetura desenvolvida, optou-se por segmentar a apresentação da mesma em cinco seções. Na primeira seção, a arquitetura será apresentada de forma mais abrangente visando fornecer um panorama geral do projeto. Na segunda seção o controle e sequenciamento das operações da arquitetura são discutidos através da apresentação de suas máquinas de estados. A terceira seção aborda a parte operativa (o *datapath*) decomposto em seus módulos principais. A arquitetura de memória, suas dimensões, estruturação e utilização são objeto da quarta seção. Por fim, na quinta seção, o processo de validação dos blocos arquiteturais é apresentado.

6.1 Visão Geral

O processo de decodificação realizado pelo CABAD trata o *bitstream* apenas no nível de macrobloco, ou seja, apenas os SEs contendo informações relacionadas à camada de macrobloco são decodificados. Dessa forma, o CABAD trabalha como "escravo" do bloco de controle global do decodificador que fica encarregado da sincronização entre os processos de decodificação do bitstream na camada de *slice* (através dos blocos de *parser* e Exp-Golomb) com a camada de macroblocos (através dos blocos de entropia).

A definição de uma estrutura arquitetural para a implementação de um *hardware* dedicado ao CABAD passa, obrigatoriamente, pela avaliação dos gargalos e cenários de aplicação para que possam ser obtidas as restrições de funcionalidade e desempenho envolvidas. Além disso, nas propostas encontradas na literatura observa-se uma forte

tendência por arquiteturas com capacidade de decodificação de um número variável de *bins* por ciclo, ou seja, a arquitetura geral da solução foi definida com base na estratégia de organização e no número de instâncias de um dos módulos de codificação aritmética.

A técnica de instanciar múltiplos módulos de codificação aritmética para possibilitar a produção de um número variável de *bins* por ciclo é uma alternativa frente às limitações impostas pela dependência de dados inerente ao conjunto de algoritmos presentes no CABAD, as quais dificultam a exploração do paralelismo. Contudo, a utilização desta técnica causa impacto sobre a organização arquitetural da solução exigindo um acréscimo de *hardware* em outros blocos como, por exemplo, na geração de contextos e no gerenciador de memória de macroblocos.

A arquitetura desenvolvida neste trabalho foi centrada na utilização de múltiplos módulos de codificação aritmética, dispostos em um arranjo em forma de árvore, conforme proposta originalmente apresentada em Yu e He (2005). Entretanto, a arquitetura desenvolvida neste trabalho estende o modelo original alterando o número de módulos de codificação aritmética presentes no ramo Bypass.

A execução do processo de decodificação realizada pela arquitetura desenvolvida é dividida em quatro fases, sendo elas: parâmetros de inicialização, inicialização de contextos, decodificação dos macroblocos e encerramento do *Slice*. A temporização entre a execução de cada uma destas fases pode ser observada na Figura 6.1.

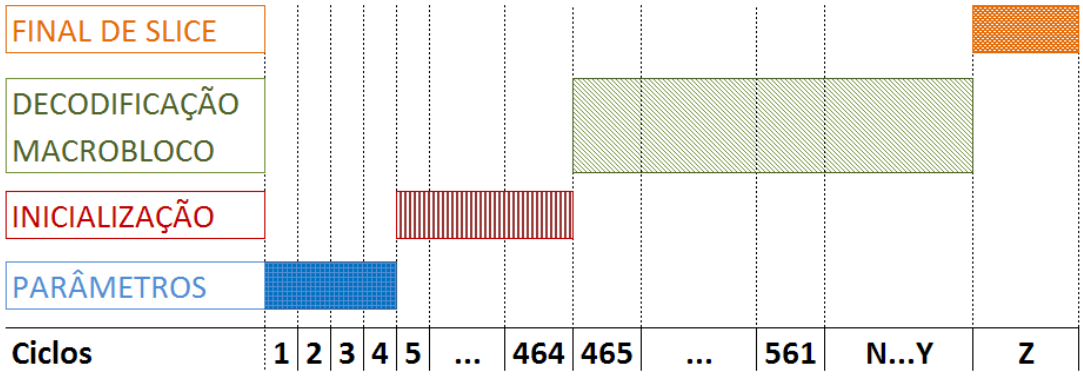


Figura 6.1: Diagrama de tempo entre as fases durante o fluxo de execução do CABAD para o processo de decodificação de um *slice*.

Na fase de recepção dos parâmetros de inicialização, o CABAD recebe do bloco superior (o *parser*) quatro parâmetros conforme a ordem que eles são decodificados. O primeiro parâmetro é o IDC, que indica a forma de inicialização da memória de contextos, que pode ser fixa ou adaptativa, variando de acordo com o tipo de *slice*. O segundo é o tipo do *slice* que deve ser decodificado a seguir. O terceiro item é o parâmetro de quantização (QP) do *slice*. O quarto parâmetro é o buffer de *bitstream* que fornece os primeiros bits provenientes da camada de macrobloco para que o CABAD possa realizar a inicialização do registrador base utilizado pelos módulos de codificação aritmética.

Na fase de inicialização de contextos a memória correspondente é inicializada com os valores gerados de acordo com os parâmetros recebidos na fase anterior. O processo de inicialização da memória de contextos ocorre uma única vez para cada *slice* decodificado e consome um total de 464 ciclos, pois trabalha com vazão de um contexto por ciclo dentro de um *pipeline* de quatro estágios, com uma latência de quatro ciclos.

A fase de decodificação de macroblocos ocupa a maior parte do tempo de execução, pois nela são processados todos os SEs de cada um dos macroblocos presentes dentro do *slice*. Considerando as restrições definidas para o nível 4.0 do perfil *main* do padrão H.264/AVC, observa-se que o número máximo macroblocos que podem ocorrer dentro de um *slice* é de 8160. Além disso, cabe salientar que cada um destes macroblocos contém diversos SEs e que para cada SE existe um processo de decodificação diferenciado.

Outro fator de grande impacto nas limitações de desempenho do CABAC está relacionado com o fato do processo de decodificação do SE ser realizado bit-a-bit, ou seja, para decodificar o valor de um SE são necessários múltiplos ciclos de execução para que todos os bits que compõem o valor do SE sejam processados. Dessa forma, a duração precisa (número de ciclos) da fase de decodificação não pode ser determinada, porém é possível afirmar que sua execução pode representar mais de 98% do tempo total de processamento de um *slice*.

Na última fase, realiza-se o encerramento do processo de decodificação do *slice*, sendo necessário esvaziar o buffer de bitstream e eliminar os bits excedentes que foram adicionados ao bitstream durante o processo de codificação, para garantir o alinhamento do bitstream em bytes inteiros e para evitar a emulação de código devido à possibilidade de falhas durante a transmissão.

Para implementar as quatro fases do processo de decodificação descrito acima as tarefas foram segmentadas em blocos, os quais foram dispostos na organização arquitetural ilustrada pela Figura 6.2. A arquitetura foi dividida em treze blocos principais, sendo nove blocos de lógica e quatro blocos de memória. Os blocos que fazem parte da arquitetura desenvolvida estão dentro do retângulo definido pelas linhas pontilhadas. O bloco representando o *parser* – que é responsável por processar o bitstream com a sequência de vídeo codificada e empacotada de acordo com NAL – é considerado um bloco externo, não fazendo parte da solução desenvolvida, dessa forma sua presença na Figura 6.2 serve apenas para ilustrar as camadas de nível superior que fornecem os dados de entrada para o CABAD.

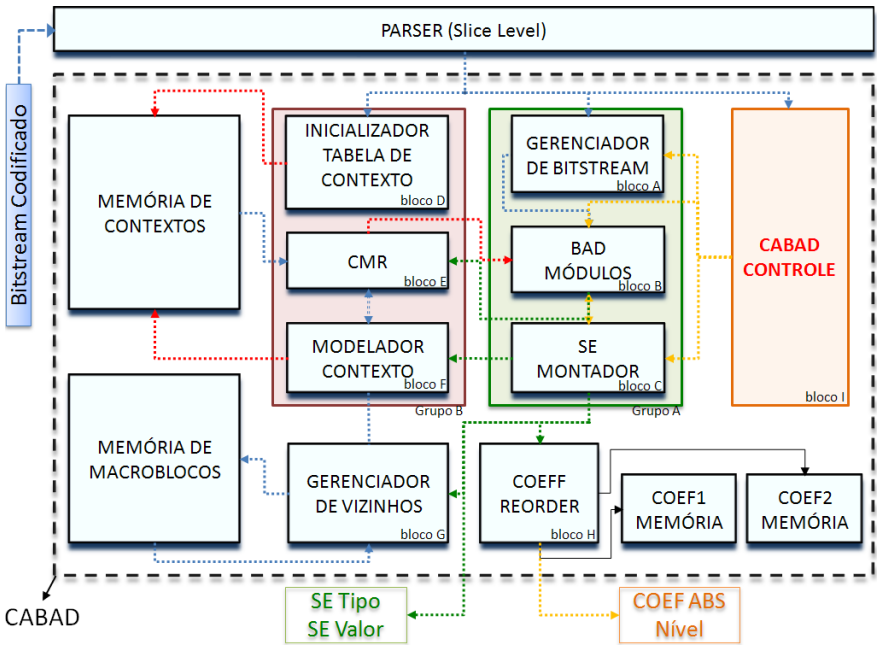


Figura 6.2: Diagrama de blocos para arquitetura do CABAD (nível mais elevado).

Para facilitar a associação entre o nome do bloco e sua representação no diagrama apresentado pela Figura 6.2, todos os blocos possuem indicadores alfabéticos na parte inferior do retângulo correspondente. Além disso, existem dois conjuntos com três blocos cada, os quais são um agrupamento de blocos com funções diretamente relacionadas.

A arquitetura desenvolvida foi organizada de acordo com o modelo clássico de parte operativa (caminho de dados) e parte de controle (controle lógico). Entretanto, dada a complexidade dos algoritmos envolvidos, optou-se pela adoção da técnica de controle autocontido, na qual os principais blocos da parte operativa contêm unidades de controle local, enquanto a parte de controle global fica responsável pelos macroestágios e por ordenar o sequenciamento entre as unidades operativas. As características e funcionalidades dos diversos blocos, bem como o relacionamento entre eles, serão discutidas nas próximas seções.

6.2 Arquitetura dos Blocos Funcionais

Considerando que a arquitetura global foi projetada com base na organização dos módulos de decodificação aritmética optou-se por iniciar a explanação sobre os blocos funcionais por esse bloco. Dessa forma, a seção 6.2.1 dedica-se ao detalhamento dos módulos de decodificação binária aritmética.

6.2.1 Módulos de Decodificação Binária Aritmética

De acordo com os dados obtidos durante o processo de análise de comportamento e baseado nos relatados da literatura, optou-se por um modelo de decodificação utilizando múltiplos módulos de decodificação aritmética, instanciados em uma organização em árvore, conforme proposta originalmente apresentada por (YU, HE, 2005). A idéia básica desta proposta é de aproveitar uma característica do fluxo de decodificação que permite, em alguns casos, a decodificação de um número variável de *bins* no mesmo ciclo. A proposta de Yu e He (2005) agrupa os módulos de decodificação em ramos de acordo com os tipos de cada módulo. A Figura 6.3 apresenta essa organização lógica dos módulos de decodificação em ramos de uma árvore.

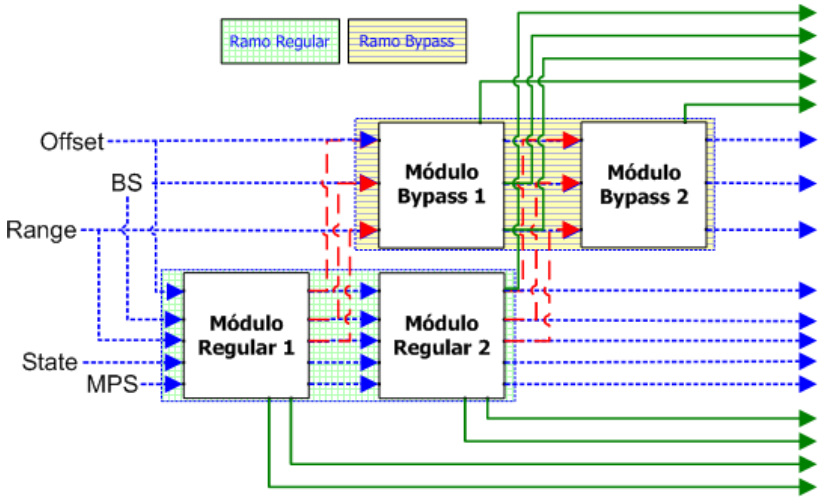


Figura 6.3: Diagrama de organização lógica entre os diversos módulos de decodificação aritmética conforme proposta de (YU, HE, 2005).

A organização arquitetural apresentada pela Figura 6.3 é interessante, pois abre a possibilidade de decodificação de múltiplos *bins* por ciclo. A seleção entre as diferentes combinações é realizada pelo bloco de controle, com base no estágio atual do processo de decodificação. Através da configuração dos caminhos de execução, é possível acionar módulos do ramo regular, ou do ramo *bypass* ou de ambos, obtendo de um a três *bins* em um único ciclo. Os canais de comunicação entre os dois ramos da árvore ilustram as possibilidades de configuração, sendo elas: a) um *bin* através do ramo regular; b) um *bin* através do ramo *bypass*; c) dois *bins* através do ramo *bypass*; d) dois *bins* através do ramo regular; ou e) dois *bins* através do ramo regular e um *bin* através do ramo *bypass*.

A proposta apresentada por (YU, HE, 2005) mostrou-se eficaz, sendo utilizada como base por diversos trabalhos disponíveis na literatura. Entretanto, a análise de comportamento dinâmico, discutida na seção 5.2, mostrou que é possível melhorar a vazão no ramo *bypass* através da adição de duas instâncias do módulo *bypass* no respectivo ramo. Aplicando essa modificação sobre a arquitetura base, ilustrada na Figura 6.3, obteve-se uma versão estendida, a qual pode ser contemplada através da Figura 6.4 (DEPRÁ, ROSA, BAMPI, 2008a).

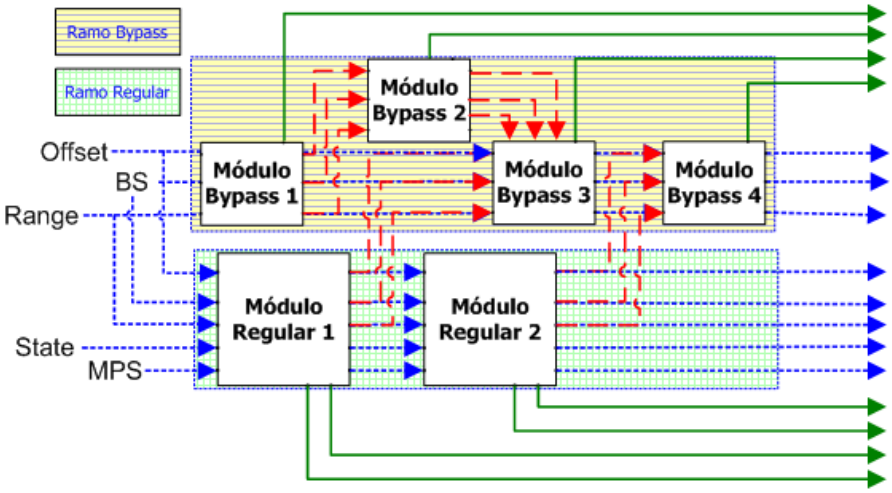


Figura 6.4: Diagrama de organização lógica entre módulos de decodificação aritmética com extensão do número de instâncias no ramo Bypass.

A organização lógica entre os módulos de decodificação aritmética apresentada pela Figura 6.4 resulta em novas possibilidades para a geração de múltiplos *bins*. Além das opções já citadas, é possível produzir até quatro *bins* de forma simultânea utilizando-se apenas do ramo *bypass*. Dada a importância do módulo de decodificação regular frente o módulo *bypass* fica evidente que aumentar o número de instâncias deste módulo poderia ser mais significativo, porém, o módulo regular é mais complexo que o módulo *bypass* e possui uma dependência de dados verdadeira, fato que dificulta a exploração de paralelismo. Além disso, reside no ramo regular o caminho crítico da arquitetura, dessa forma, aumentar o grau de profundidade de lógica combinacional deste bloco traria impacto sobre a frequência de operação da arquitetura como um todo.

Obviamente, o diagrama apresentado pela Figura 6.4 é apenas uma simplificação que enfatiza a alteração na estrutura do ramo Bypass, pois a arquitetura implementada para o bloco BAD contempla todos os três tipos de módulos: *regular*, *bypass* e *terminate*. A Figura 6.5 apresenta a organização destes módulos e seu relacionamento com os registradores internos e externos (DEPRÁ, ROSA, BAMPI, 2008b).

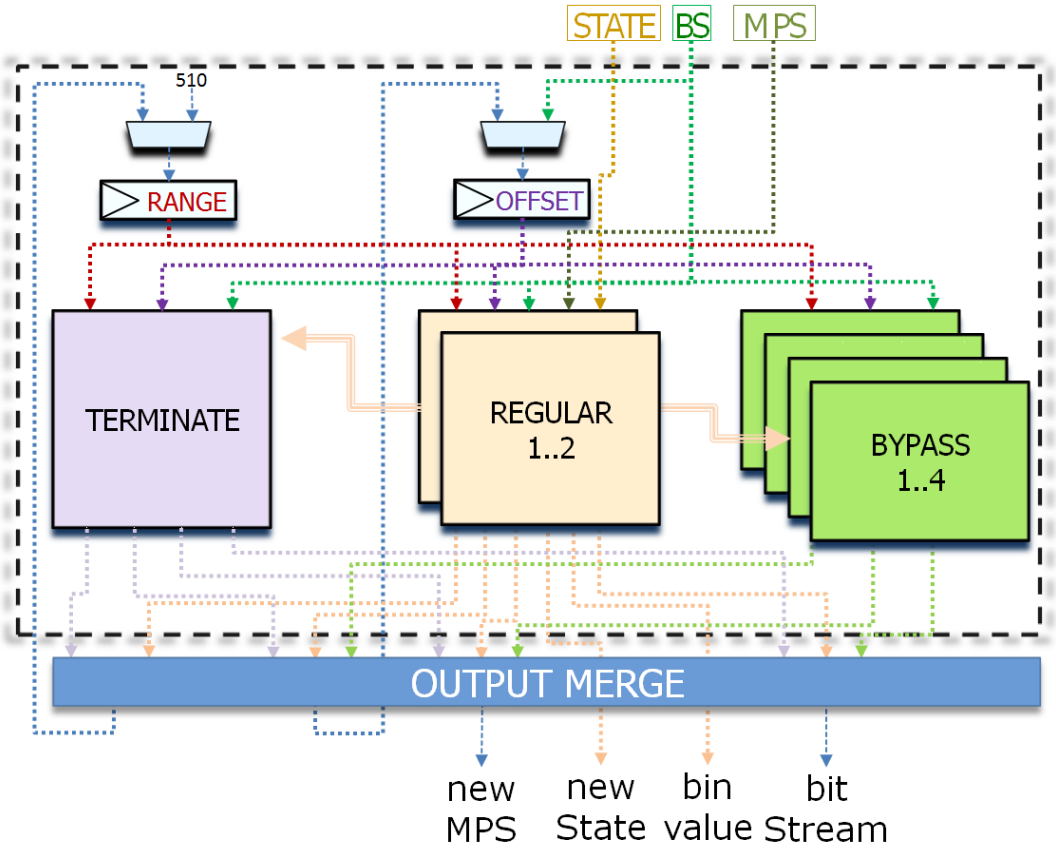


Figura 6.5: Diagrama de blocos para estrutura interna dos módulos BAD – Bloco B.

No interior do bloco representado pela Figura 6.5 existe, além dos módulos de decodificação aritmética, um conjunto de registradores que representam o estado interno do processo de decodificação. Os registradores de *offset* e *range* podem ser considerados os principais, pois garantem a precisão do processo de decodificação mantendo o intervalo de probabilidades ao longo da cadeia de subdivisões sucessivas. O valor destes registradores é utilizado para realizar os cálculos aritméticos, sendo necessário assegurar sua consistência entre execuções subsequentes. Assim, torna-se imperativo que as saídas de um módulo sejam concatenadas com as entradas do módulo adjacente, de forma que o valor alterado em um módulo seja refletido nos demais. Além disso, existem várias configurações possíveis para a produção de múltiplos *bins* e, para atender todas as combinações, são necessários multiplexadores que permitam a escolha seletiva para a entrada de cada um dos sinais.

O bloco com os módulos BAD necessita, além dos registradores internos, acessar os registradores *state*, *mps* e *bs*, os quais são considerados registradores externos. Os dois primeiros são provenientes do estágio de modelagem de contextos, enquanto o último é proveniente do bloco de manipulação do *bitstream*. O registrador *BS* pode ser utilizado por qualquer um dos módulos de decodificação aritmética, pois quando o registrador *range* atinge um valor crítico, abaixo da metade da resolução total, é necessário executar um processo de renormalização, o qual consome bits do *bitstream* para manter o registrador de *range* dentro de uma faixa válida. Os registradores *STATE* e *MPS* são acessados exclusivamente pelo módulo regular, que é o único módulo onde os dados da modelagem de contextos são necessários.

Os sinais de controle gerados pelo controle global são responsáveis pela seleção dos caminhos de decodificação e ativação dos módulos de decodificação além de selecionar as saídas apropriadas que servirão para atualizar os dados dos registradores internos e contextos para a execução da próxima etapa.

A utilização de dois módulos de decodificação em paralelo no ramo *regular* permite a decodificação de até dois bins por ciclo neste ramo. Essa possibilidade é ideal para decodificação de SE com múltiplos *bins* ou para os *flags* do mapa de significância que, tipicamente, ocorrem em pares de SEs dos tipos COEF_SIG e COEF_LAST. Entretanto, a partir das observações sobre o comportamento do mapa de significância, discutidas no Capítulo 5, identificou-se que, na média, em 30% dos casos esse par de SEs não ocorre, pois o coeficiente associado a eles possui valor igual a zero e não é o último coeficiente com valor válido no bloco 4x4.

Para explorar a aceleração do processo de decodificação do mapa de significância, foi adotada uma estratégia que permite decodificar dois *flags* do mapa de significância no mesmo ciclo, independente de ser um COEF_SIG ou COEF_LAST. Essa estratégia é baseada no fornecimento de três contextos distintos para o ramo *regular*. No fluxo de decodificação normal, primeiro seria decodificado um SE do tipo COEF_SIG, depois um SE do tipo COEF_LAST. Entretanto, para alguns casos o SE do tipo COEF_LAST não é gerado, fato que faz com que o próximo SE a ser decodificado seja um SE do tipo COEF_SIG, fazendo com o contexto fornecido ao segundo módulo de decodificação do ramo regular fique incorreto e a saída deste módulo tenha que ser descartada.

Uma alternativa para resolver esse prolema seria fornecer os dois contextos possíveis para o segundo módulo de decodificação *regular*. Assim, um simples multiplexador com a seleção vinculada ao resultado do primeiro módulo *regular* poderia ser adotado. Essa solução foi implementada e o relacionamento entre os dois módulos de decodificação *regular* pode ser observado através da Figura 6.6.

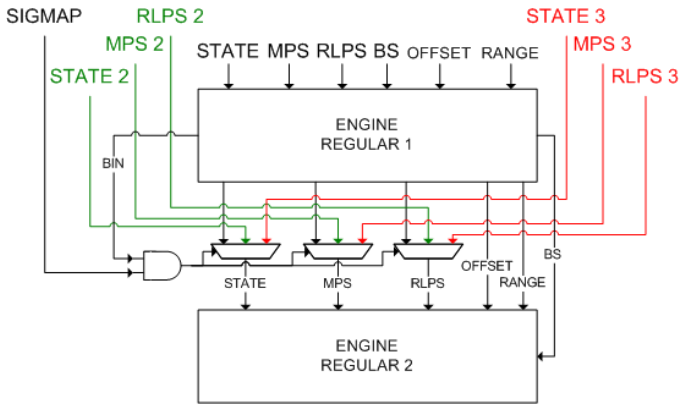


Figura 6.6: Diagrama de blocos com relacionamento entre módulos no ramo *Regular*.

Os detalhes de implementação de cada um dos módulos de decodificação aritmética serão discutidos a seguir.

6.2.1.1 Módulo Regular

O projeto do módulo *regular* é de extrema importância para o desempenho da arquitetura geral do CABAD, pois esse bloco está no caminho critico do *datapath*. É importante observar que algumas das dependências de dados que afetam o CABAD se manifestam neste bloco como, por exemplo, a dependência de dados relacionada com a

etapa de modelagem de contextos que é decorrente da necessidade de acessar as tabelas de resultados pré-calculados que são indexadas pelo registrador *state*, o qual poderá ser alterado durante a execução do módulo *regular* e servirá de base para a próxima decisão. Além disso, há a questão da renormalização, que pode necessitar de um número variável de ciclos e provocar alterações no valor dos registradores *offset* e *range*.

A Figura 6.7 ilustra a arquitetura desenvolvida para o módulo de decodificação *regular*. Visando reduzir ao máximo o atraso combinacional, todas as operações que são realizadas por esse bloco foram otimizadas, com destaque especial ao sub-bloco responsável pela renormalização e pelo sub-bloco de geração dos sinais de próximo estado e próximo RLPS. Como resultado final, obteve-se uma arquitetura eficiente na utilização dos recursos de *hardware*.

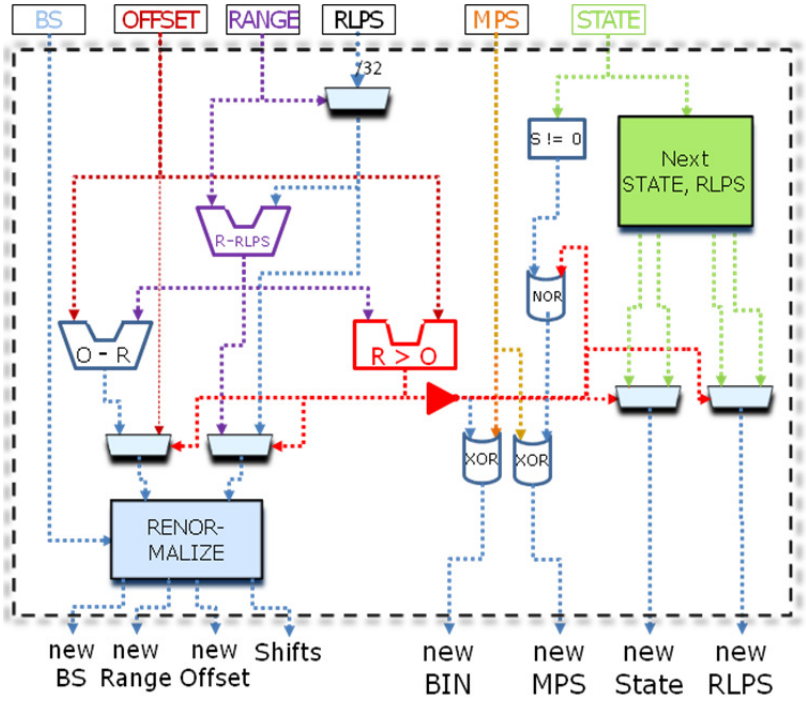


Figura 6.7: Diagrama de blocos para estrutura interna do módulo *Regular*.

Para minimizar o impacto causado pela necessidade de acessar uma ROM durante a etapa de codificação aritmética optou-se por copiar esses dados para a RAM associada com os contextos. Assim, o processo de inicialização de contextos fica responsável por inicializar cada contexto com os valores do estado, elemento mais provável e o seu respectivo valor de RLPS para o estado inicial, eliminando-se, assim, a necessidade de fazer acesso a ROM durante o processo de decodificação aritmética. Porém, a cada execução do processo de decodificação aritmética, um novo estado é gerado e cada estado precisa acessar um novo valor de RLPS, assim, é necessário gerar o novo valor de RLPS para o próximo estado. Dessa forma, continua-se com a necessidade de acessar uma ROM durante a etapa de decodificação aritmética, porém, essa informação será utilizada apenas na próxima utilização do módulo regular. A Figura 6.8 ilustra a organização interna da ROM que representa o sub-bloco de próximo estado e próximo RLPS. Essa memória é endereçada pelo registrador de estado provido pelo estágio de modelagem de contexto.

[6][75:70]	[8][69:62]	[8][61:54]	[8][53:46]	[8][45:38]	[6][37:32]	[8][31:24]	[8][23:16]	[8][15:8]	[8][7:0]
STATE	QUANTIZED RLPS				STATE	QUANTIZED RLPS			
NEXT MPS					NEXT LPS				

Figura 6.8: Estrutura organizacional da ROM que contém dados para o próximo estado e o próximo RLPS para ambas as situações: MPS ou LPS.

Para resolver a questão do número variável de ciclos necessários para executar a renormalização, adotou-se a técnica da detecção do primeiro um produzido – FOD (do inglês, *first one detect*) de forma similar ao processo descrito por (KIM, PARK, 2006). Além disso, foram empregados dois FOD, cada um com metade do comprimento total do registrador *range*, possibilitando uma aceleração adicional, pois o processo de detecção para o "primeiro um" foi reduzido pela metade e uma simples operação lógica "OR" é capaz de decidir entre a saída da parte inferior ou da parte superior. No final, o bit mais significativo do registrador *range* é utilizado para decidir se a saída do bloco de renormalização será resultado das modificações e concatenações internas ou será apenas a entrada original não modificada. O diagrama apresentado pela Figura 6.9 ilustra a estrutura interna do bloco de renormalização conforme descrito anteriormente.

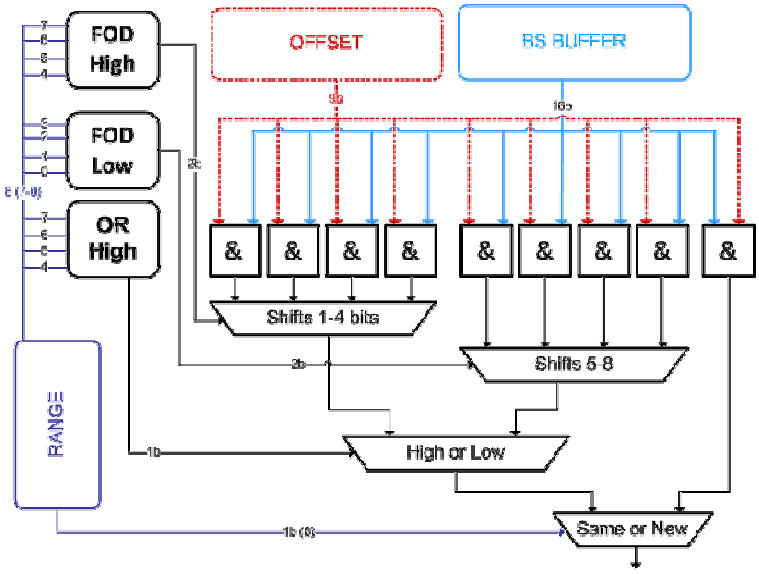


Figura 6.9: Diagrama de blocos para estrutura interna do bloco de renormalização.

A estratégia aplicada ao bloco de renormalização permitiu a execução do processo de renormalização em um único passo, com baixo impacto na frequência de operação, fato que possibilita sua integração ao estágio de decodificação aritmética. Além disso, a eliminação da necessidade de acesso a ROM para obter os valores de RLPS e do registrador de próximo estado resultaram em uma redução significativa do atraso deste módulo.

6.2.1.2 Módulo Bypass

A arquitetura do módulo *Bypass* é composta, basicamente, por um bloco de subtração, um comparador, uma operação de concatenação e uma operação de deslocamento. Essa estrutura pode ser observada na Figura 6.10. Este módulo, ao contrário do módulo *Regular,* não precisa de nenhuma informação de contexto, por isso, sua dependência de dados se dá apenas sobre o registrador *offset*. O registrador *bs buffer*, também sofre alteração, porém, por se tratar de uma simples operação de

deslocamento é possível iniciar a execução de uma segunda instância deste módulo em paralelo com a primeira.

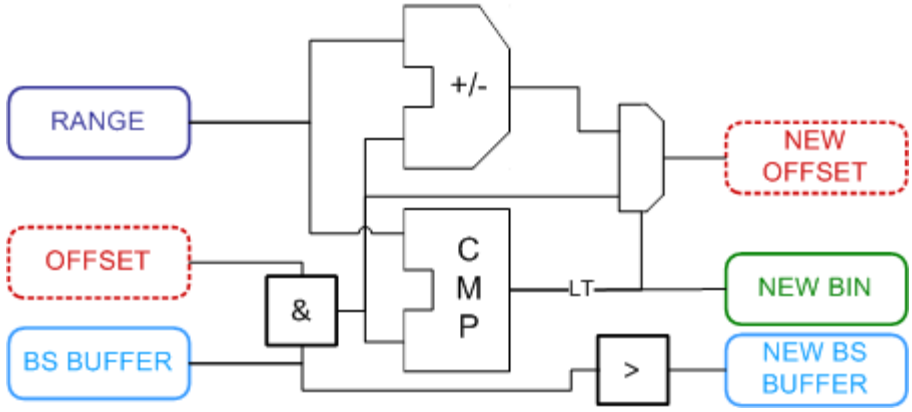


Figura 6.10: Diagrama de blocos que representa a estrutura interna do módulo *Bypass*.

6.2.1.3 Módulo Terminate

A Figura 6.11 ilustra a estrutura do módulo *Terminate*. Este módulo possui complexidade maior que o módulo Bypass, porém inferior ao módulo *Regular*. Durante sua execução, esse módulo pode alterar o valor dos registradores *offset*, *range* e *bs buffer*. A diferença principal entre os módulos *Terminate* e *Bypass* reside na presença do bloco de renormalização, o qual não é encontrado no módulo *Bypass*.

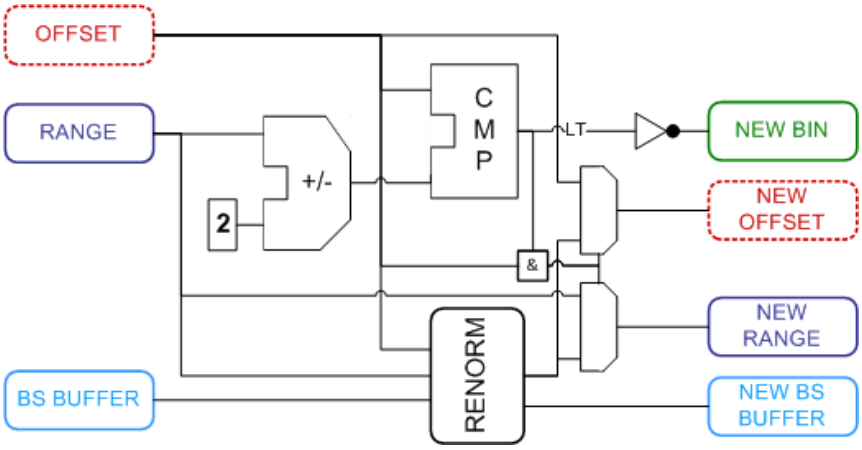


Figura 6.11: Diagrama de blocos que representa a estrutura interna do módulo *Terminate*.

A eliminação da necessidade de acesso à ROM para obter os valores de RLPS e do registrador de próximo estado resultaram em uma redução significativa do atraso deste módulo.

6.2.2 Gerenciamento do *BitStream – BS Buffer*

A manipulação do *bitstream* deve prover um mecanismo capaz de atender ao consumo de um número variável de bits a cada ciclo. O consumo de bits do *bitstream* ocorre apenas nos módulos *Bypass* e durante a etapa de renormalização, que está presente nos módulos *Regular* e *Terminate*. A arquitetura desenvolvida, para atender a essa necessidade de fornecer um número variável de bits, é baseada em uma estrutura de cascata de registradores de forma similar a técnica apresentada por (MEI-HUA, et al.,

2007). A estrutura da arquitetura desenvolvida para o bloco de gerenciamento do *bitstream* é apresentada na Figura 6.12.

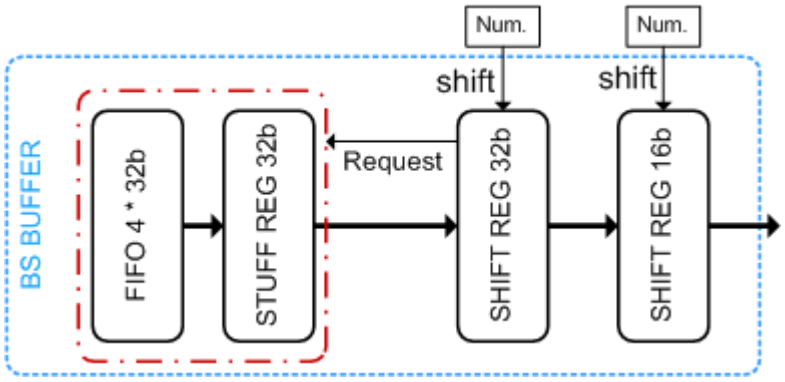


Figura 6.12: Diagrama de blocos que representa a estrutura interna do bloco Gerenciador de *Bitstream* – Bloco A.

Observando a composição da arquitetura apresentada pela Figura 6.12, é possível identificar uma FIFO de quatro posições de 32 bits cada, um registrador temporário de 32 bits que captura a saída da FIFO e dois registradores de deslocamento capazes de efetuar o deslocamento de um número variável de bits definidos pelo sinal de *shift_num*. O registrador *shift reg 16b* fica acessível aos módulos de decodificação aritmética referenciado pelo nome *BS*. Ao iniciar o processo de decodificação de um macrobloco, a arquitetura solicita ao bloco de *Parser* seis linhas de 32 bits, que serão utilizadas para inicializar os registradores internos do gerenciador de *bitstream*. Com isso, a FIFO ficará totalmente preenchida, passando duas linhas de 32 bits para a saída, que será capturada pelo registrador *stuff_reg_32b* e repassada ao registrador *shift_reg 32b*, o qual fornecerá 16 bits ao registrador *shift_reg_16b*.

Durante o processo de decodificação do macrobloco, os módulos de decodificação aritmética irão consumir bits do *bitstream*. A cada iteração, poderão ser consumidos nenhum ou vários bits e isso será indicado pelos sinais de shifts_num. Sempre que ocorrer o consumo de um bit, o registrador *shift reg 32b* sofrerá o deslocamento equivalente ao número de bits consumidos, sendo que os bits que saírem deste registrador serão adicionados ao registrador *shift reg 16b*, o qual também sofrerá um deslocamento para possibilitar a "absorção" dos novos bits. Esses dois registradores são controlados por dois contadores que mantêm o número de bits válidos presente neles. Sempre que o registrador *shift_reg_32b* atingir valor zero no contador de bits válidos, será enviado um sinal de *request* ao controlador da FIFO, que irá despachar sua saída no registrador de *stuff_reg 32b,* o qual irá recarregar o registrador *shift_reg_32b*, possibilitando que o processo de decodificação prossiga sem interrupções.

6.2.3 Construção da Tabela de Contextos

Durante o processo de decodificação, o CABAD utiliza uma memória de contextos para manter atualizadas as probabilidades de ocorrência para todos os SE. A memória de contexto contém 460 posições e em cada posição são armazenadas duas informações: o estado atual (sinal com seis bits) e o MPS (sinal com um bit). Porém, antes que esses contextos possam ser utilizados pelos módulos de decodificação aritmética, se faz necessário inicializar cada contexto com o valor de estado e MPS adequados. Esse processo é realizado uma única vez para cada *slice* processado.

O processo de inicialização da memória de contexto implementado nesta arquitetura possui uma peculiaridade em relação ao processo definido na sub-seção 4.3.3.2. Em virtude das exigências dos múltiplos módulos de decodificação aritmética, deve-se prover, além dos sinais de estado atual e MPS, o valor RLPS. Assim, torna-se necessário incluir na memória de contextos a informação sobre o RLPS para cada estado, informação que originalmente ficava armazenada em uma ROM, que era consultada durante a etapa de decodificação aritmética. Essa alteração traz implicações ao tamanho da memória de contexto e no seu respectivo processo de inicialização.

A Figura 6.13 ilustra a estrutura do bloco de inicialização para a memória de contextos. Na Figura 6.13, é possível observar que as operações são balanceadas em um *pipeline* de três estágios. As alterações mais significantes nesta arquitetura aparecem no terceiro estágio do *pipeline*, pois depois de gerar o valor para o sinal de estado, esse valor é utilizado como índice de uma ROM, que contém o valor do símbolo RLPS para cada estado possível, sendo que o valor do símbolo MPS, do estado gerado e o valor do símbolo RLPS recuperado a partir da ROM, são concatenados e gravados juntos na memória de contextos. Além disso, existe um comparador responsável por verificar o endereço do contexto a ser inicializado, pois a memória de contextos é dividida em duas memórias. Assim, quando o limite de endereçamento da primeira memória é atingido o registrador de endereço deve ser reinicializado e os novos dados e endereços gerados devem ser encaminhados para a outra memória.

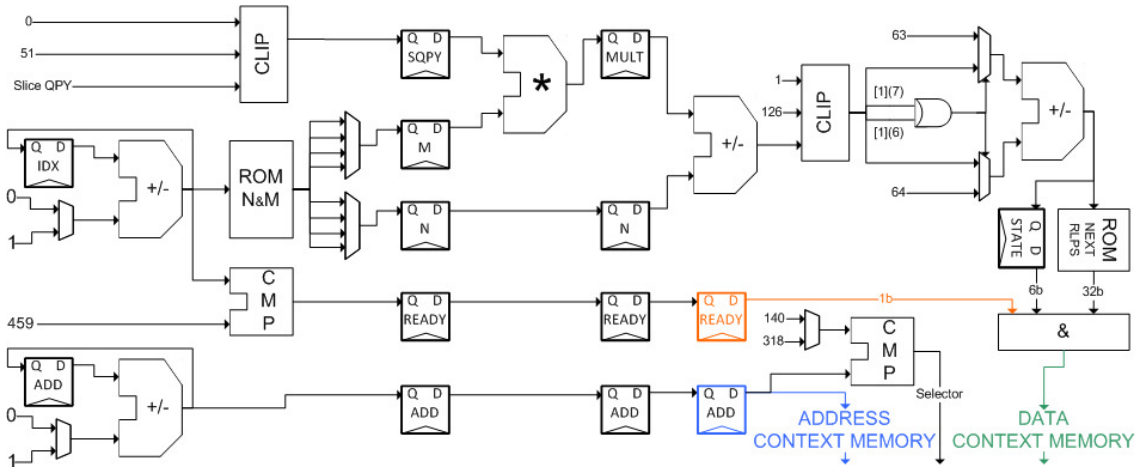


Figura 6.13: Diagrama arquitetural que representa o bloco de inicialização da memória de contextos – Bloco D.

6.2.4 Gerenciamento dos Vizinhos

Durante o processo de decodificação aritmética o acesso a memória de contextos é necessário para a utilização dos módulos de codificação *regular*, porém, para acessar um determinado contexto, é necessário calcular seu endereço e esse cálculo é realizado de forma diferenciada para cada tipo de SE, conforme especificado na sub-seção 4.3. Contudo, em linhas gerais, cada SE possui um endereço base ao qual deve ser somado um valor de incremento. O valor deste incremento é definido com base no valor do mesmo SE localizado no macrobloco superior e no macrobloco a esquerda do macrobloco atual. Dessa forma, para produzir o endereço de contexto, é necessário realizar, ao menos, dois acessos a memória de macroblocos.

O acesso à memória de macroblocos pode se tornar um gargalo no decodificador de vídeo, se for compartilhada por vários blocos. Por outro lado, se cada bloco utilizar uma

memória interna para armazenar os dados necessários ao seu gerenciamento de macroblocos, é possível resolver o problema de concorrência pelo acesso, porém, ao custo do desperdício de área para armazenar cópias duplicadas dessa informação. A alternativa adotada foi elaborar uma organização eficiente para os dados dos SEs que precisam ser mantidos na memória de macroblocos e empregar um conjunto de banco de registradores para guardar toda a informação de cabeçalho de cada um dos macroblocos vizinhos, além de um banco de registradores para os SEs do macrobloco atual. A Figura 6.14 ilustra de forma simplificada a arquitetura do bloco de gerenciamento de vizinhos com destaque para os bancos de registradores.

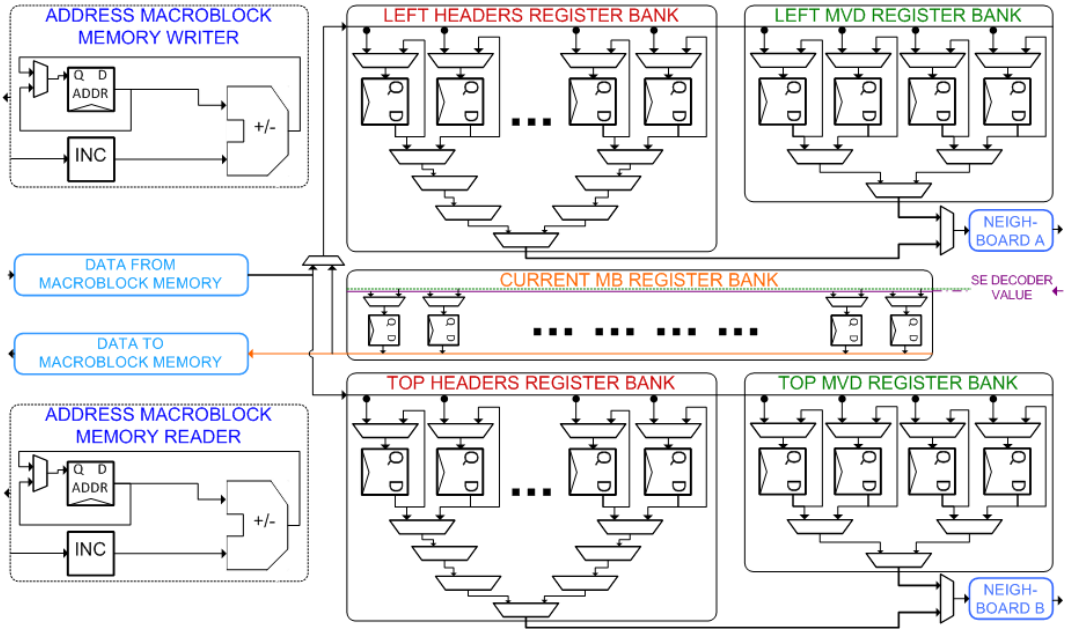


Figura 6.14: Diagrama para o bloco de Gerenciamento de Vizinhos – Bloco G.

A arquitetura desenvolvida visa minimizar a quantidade de acessos a memória de macroblocos para possibilitar a recuperação da informação de vizinhança para geração dos endereços de contexto com o menor número de ciclos possível. Assim, no banco de registradores denominado "*current MB register bank*" são mantidos os valores dos SEs para o cabeçalho do macrobloco atual, assim como um par de componentes de vetor de movimento diferencial, valores produzidos pelo bloco responsável de montagem os SEs – bloco C da Figura 6.2.

Quando um novo macrobloco começa a ser processado, o gerenciador de vizinhos salva a informação contida no banco de registradores denominado "*current MB register bank*" na memória de macroblocos e copia essa informação para o banco de registradores denominado "*left header register bank*", pois os dados produzidos durante a decodificação do macrobloco anterior serão os vizinhos da esquerda para o novo macrobloco. Dessa forma, é possível evitar o acesso à memória de macroblocos para recuperação desta informação. A seguir, a informação relativa ao vizinho superior é recuperada da memória de macroblocos e armazenada no banco de registradores denominado "*top header register bank*". Depois, os SEs do macrobloco atual podem começar a ser decodificados, pois toda a informação necessária para calcular o endereço do contexto estará disponível em apenas um ciclo.

O processo de decodificação dos MVD recebe um tratamento ligeiramente diferenciado, pois ao invés de guardar informações de todos os MVD nos macroblocos

vizinhos, superior e da esquerda, dentro de registradores, guarda-se apenas um par de componentes para cada vizinho. Conforme cada MVD é decodificado, o próximo par de vizinhos é recuperado a partir da memória de macroblocos. Dessa forma, é possível garantir um bom compromisso entre o custo em área e eficiência de decodificação.

6.2.5 Gerenciamento de Contextos

A tarefa de gerenciamento de contextos é uma das etapas mais complexas executadas pelo CABAD. As atribuições deste bloco incluem a geração do endereço de cada contexto a ser utilizado e garantia de que as alterações realizadas durante o processo de decodificação sejam corretamente armazenadas na memória de contextos. Além disso, o gerenciador de contextos deve prover até dois contextos em um mesmo ciclo para atender as necessidades do bloco de decodificação binária aritmética.

A arquitetura desenvolvida para contemplar as tarefas de gerenciamento de contextos foi dividida em duas partes, ou blocos, denominados de Bloco E e Bloco F conforme indicado na Figura 6.2. O Bloco F é responsável pela geração do endereço de contexto, com base na informação de vizinhança, no tipo de SE que está sendo decodificado e em seus registradores internos. A Figura 6.15 apresenta o diagrama arquitetural para o bloco de geração dos endereços de contexto.

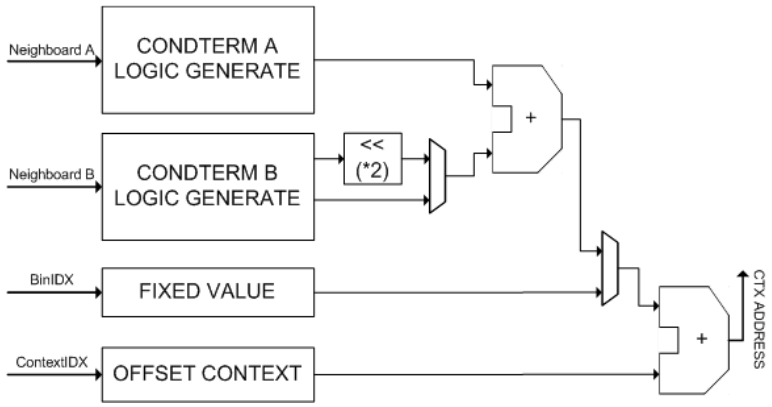


Figura 6.15: Diagrama arquitetural para o bloco de Modelagem de Contexto (geração dos endereços de contexto) – Bloco F.

O Bloco F produz o endereço do contexto que deve ser recuperado a partir da memória de contexto, porém, para otimizar o tempo do processo de decodificação e reduzir o número de acessos a memória de contextos, foi desenvolvido o Bloco E, que é um repositório para contextos recentemente acessados. Esse repositório funciona de forma similar a uma memória *cache*, pois permite que um pequeno número de contextos seja mantido fora da memória, eliminando a necessidade de acessar a memória de contextos quando o contexto desejado já estiver carregado no repositório. A Figura 6.16 ilustra a arquitetura do repositório de contextos.

Para cada endereço gerado pelo Bloco F, o Bloco E é acionado para verificar se o endereço desejado já está disponível no repositório de contextos. O repositório é capaz de manter até quatro contextos distintos. Para cada contexto é necessário manter 39 bits relativos ao contexto e mais 8 bits relativos ao endereço deste contexto na memória de contextos.

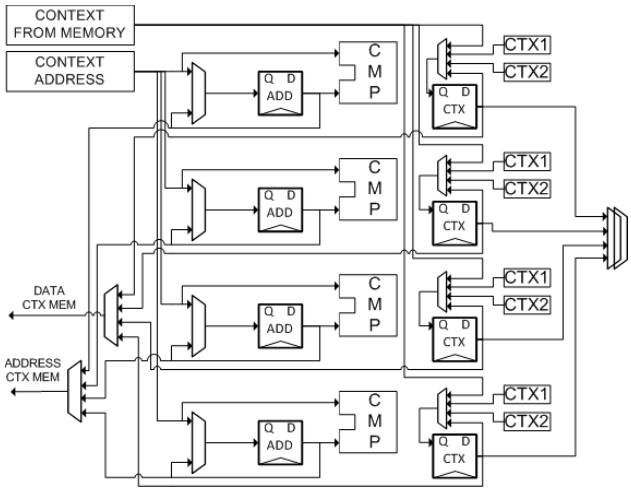


Figura 6.16: Diagrama arquitetural para o repositório de contextos – Bloco E.

O repositório de contexto é capaz de fazer a comparação do endereço desejado com o endereço de todos os quatro contextos de forma paralela. Sempre que o endereço desejado estiver disponível no repositório ele será repassado ao bloco de decodificação aritmética, caso contrário, ocorrerá uma bolha no *pipeline* de decodificação, então o contexto adequado deverá ser localizado na memória de contextos. O repositório pode fornecer até dois contextos de forma simultânea. Sempre que um novo contexto for recuperado o repositório verifica se há uma posição livre para armazená-lo. Não havendo tal posição uma das posições será liberada através de uma política de substituição baseada em LRU (*last recent updated*) e o contexto corresponde a ela será armazenado na memória de contextos.

6.3 Hierarquia de Memória

O gerenciamento de memória proposto utiliza três memórias distintas, de acordo com o propósito de utilização de cada uma delas. A memória de macroblocos é utilizada para guardar informações dos macroblocos decodificados anteriormente. A memória de contextos mantém as informações sobre o índice de estado e valor do MPS, os quais garantem que o processo de decodificação se mantenha atualizado. A memória de coeficientes armazena o valor dos coeficientes de transformada decodificados, que deverão ser processados pelos blocos de quantização inversa e transforma inversa. As características destas memórias serão descritas a seguir.

6.3.1 Memória de Macroblocos

A estrutura da memória de macroblocos deve oferecer suporte a um número de macroblocos equivalente à maior resolução que o decodificador pode tratar. Neste caso, consideramos um decodificador compatível com padrão H.264/AVC para o perfil *main* no nível 4.0, com as seguintes restrições: tratar apenas vídeos progressivos; sem suporte para quadros do tipo *field*; e limitado a resolução HDTV 1080p.

Para a decodificação de todos os SEs de cada macrobloco, o CABAD precisa acessar informações sobre os SEs decodificados nos macroblocos localizados espacialmente acima e à esquerda (denominados: vizinho superior e esquerdo) do macrobloco atual. O vizinho superior é o macrobloco localizado na posição horizontal do macrobloco atual, porém uma linha de macroblocos acima do mesmo. O vizinho da esquerda é o macrobloco localizado na mesma posição vertical, porém um macrobloco à

esquerda do atual, sendo, normalmente, o macrobloco decodificado imediatamente antes do atual.

A determinação da quantidade de memória necessária para armazenar as informações de vizinhança depende da dimensão horizontal da máxima resolução suportada, pois manter todas as informações sobre a vizinha do macrobloco atual implica em guardar, na memória de macroblocos, uma linha horizontal inteira. Na Figura 6.17 são apresentadas duas linhas de macroblocos para um vídeo com a resolução QCIF, o qual possui, no total, onze (11) linhas de macroblocos com nove (9) colunas de macroblocos por linha. Neste exemplo, o macrobloco "MB6" da linha "L2" é considerado o macrobloco atual, tendo como vizinho superior o macrobloco "MB6' da linha "L1" e vizinho esquerdo o macrobloco "MB5" da linha "L2".

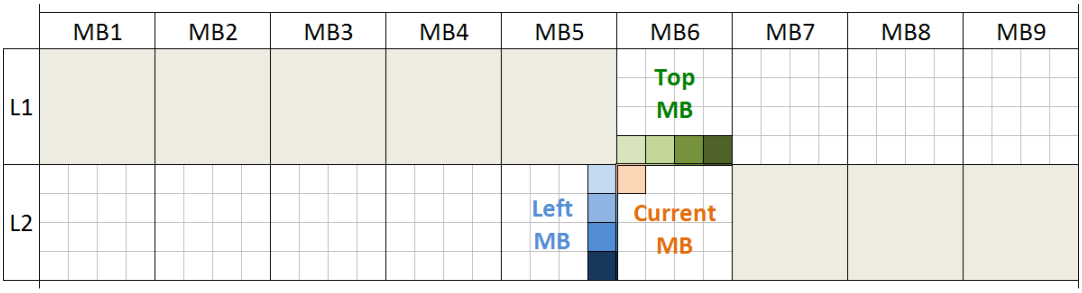


Figura 6.17: Representação da vizinhança necessária para decodificar um macrobloco, considerando um vídeo na resolução QCIF.

Considerando que a arquitetura desenvolvida oferece suporte para vídeos na resolução HDTV 1080p, a memória de macroblocos precisa comportar até 120 macroblocos. Cada macrobloco deve manter todos os SEs de cabeçalho, sendo eles: MBSKIP[1], MBFIELD[1], MBTYPE[6], SUBMBTYPE[4], QPDELTA[7], REFIDX[4], MVD[16], INTRA_FLAG[1], INTRA_MODE[3]. O número entre colchetes ao lado do SE indica o número de bits necessário para representar o valor do mesmo. Os SEs relacionados com a forma de predição de movimento têm sua ocorrência vinculada ao tipo de macrobloco. Assim, os SEs dos tipos REFIDX e os MVD só ocorrem em macroblocos dos tipos P e B, enquanto os SEs dos tipos INTRA_FLAG e INTRA_MODE são exclusivos para macroblocos do tipo I. Essa informação é útil para minimizar o tamanho da memória de macroblocos.

Para armazenar todos os SEs para os 120 macroblocos são necessários 127,5 Kb de memória, que foram organizados em 2040 linhas com 64 bits por linha. O tamanho da palavra de memória foi escolhido em razão dos MVD, pois cada componente (X, Y) do vetor é representado por 16 bits e cada um dos 16 blocos 4x4 pode conter um vetor para cada lista (L0 e L1); para cada MVD podemos ter 16*2*2 = 64 bits (uma linha de memória). Os demais SEs que formam o cabeçalho de um macrobloco Inter ocupam apenas 62 bits, ou seja, podem ser armazenados em uma única linha da memória que somada às outras 16 linhas exigidas para armazenar todos os MVD, totalizam as 17 linhas necessárias para guardar as informações de um macrobloco do tipo Inter. Para guardar os SEs que formam o cabeçalho de um macrobloco do tipo Intra, necessita-se de apenas 87 bits, que podem ser alocados em duas linhas de memória. A Figura 6.18 ilustra a organização da memória de macroblocos, com destaque para formação diferenciada dos macroblocos do tipo Inter e Intra.

A quantidade de memória necessária para representar os macroblocos dos tipos Intra e Inter é diferente, porém, optou-se por utilizar a mesma quantidade para ambos, dado

que a diferenciação tornaria o cálculo de endereço dos macroblocos mais complexo, levando a necessidade de uma tabela de endereço base para os macroblocos ou outro mecanismo mais elaborado. Além disso, para o pior caso os 127,5 Kb de memória ainda seriam necessários. A memória de macroblocos possui duas portas para escrita e/ou leitura que visam minimizar os problemas relacionados ao acesso simultâneo para ler e gravar informações no mesmo ciclo.

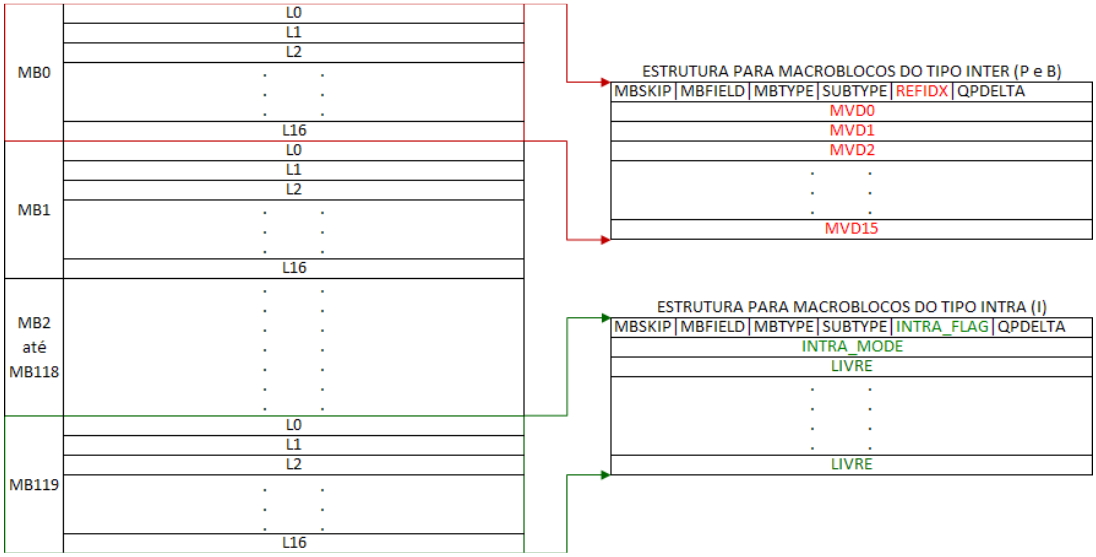


Figura 6.18: Estrutura da memória de macroblocos com variações na forma de interpretação de acordo com o tipo de macrobloco.

6.3.2 Memória de Coeficientes

A memória de coeficientes é utilizada para armazenar o valor dos coeficientes de transformadas decodificados pelo CABAD que serão, posteriormente, processados pelos blocos de quantização inversa e transformadas inversas. A organização desta memória foi baseada na proposta de Chen, Chang e Lin (2005), na qual são utilizadas duas memórias de porta simples, cada qual com capacidade de armazenar os coeficientes para um único macrobloco. Cada macrobloco possui 256 coeficientes de luminância, 64 coeficientes de crominância azul e 64 coeficientes de crominância vermelha, totalizando 384 coeficientes, cada qual com 16 bits para sua representação. No total são necessários 6 Kb para cada memória, organizados em 96 linhas de 64 bits. Essa organização com palavras de 64 bits é interessante, pois permite agrupar quatro coeficientes por linha, ou um bloco 4x4 a cada quatro linhas. A Figura 6.19 ilustra a estrutura das duas memórias de coeficientes com a sua organização, de acordo com as componentes de cor, começando pelas amostras de luminância, seguidas pela crominância azul e, por fim, crominância vermelha.

O CABAD faz apenas o processo de escrita nestas memórias, enquanto os blocos de quantização e inversa e transformadas inversas fazem o processo de leitura. A adoção de duas memórias visa evitar problemas de concorrência no acesso, assim enquanto o CABAD faz o processo de escrita em uma memória, os blocos de quantização inversa e transformadas inversas acessam a outra memória. Quando encerrado o processamento de um macrobloco, o CABAD troca a memória de destino, liberando a última memória utilizada. Esse mecanismo de alternância de utilização entre cada uma das memórias de coeficientes funciona de forma similar ao *ping-pong*, ou seja, para cada macrobloco decodificado ocorre uma troca entre as memórias de leitura e gravação.

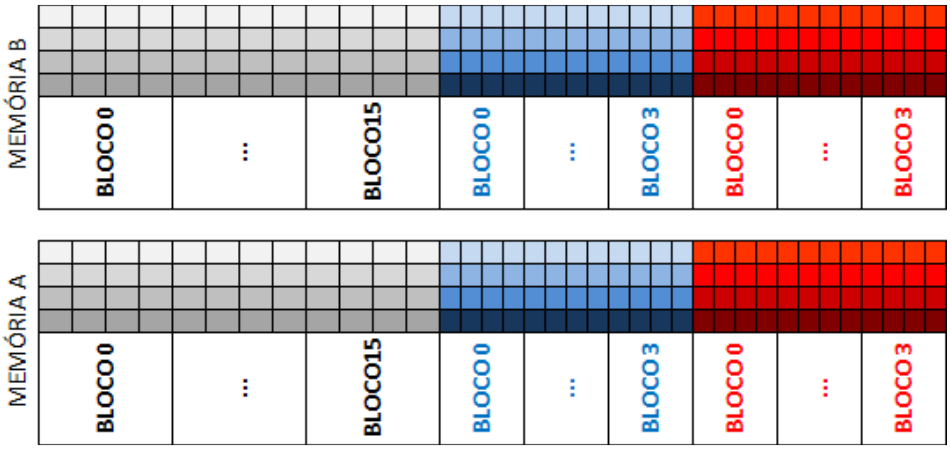


Figura 6.19: Organização das memórias para coeficientes de transformada.

6.3.3 Memória de Contextos

A organização da memória de contextos merece atenção especial, pois essa memória precisa ser acessada duas vezes para cada *bin* processado pelo CABAD. Logo, para a utilização em um mecanismo de *pipeline* é necessário utilizar uma memória de porta dupla, pois a cada ciclo será necessário realizar uma leitura e uma escrita. Além disso, é necessário atender aos requisitos de funcionalidade do bloco de decodificação binária aritmética que, para a arquitetura desenvolvida, prevê a possibilidade de tratar até dois contextos diferentes no mesmo ciclo. A solução adotada para esse problema foi baseada na proposta de Chen e Lin (2007) na qual a memória de contexto é segmentada em duas memórias, uma destinada aos contextos relativos ao SE do tipo SIG_MAP e outra para os contextos vinculados a outros SEs.

O emprego da estratégia de separação da memória de contextos em duas memórias, aliado à utilização do repositório de contextos, minimiza o problema de fornecimento de contextos para o bloco de decodificação aritmética binária. Entretanto, ainda resta o problema relacionado ao acesso para ROM com o valor do RLPS para cada estado, implicando em um ciclo adicional para cada *bin* processado. Esse problema foi resolvido através de uma nova abordagem, onde o estado e o valor de seu símbolo RLPS correspondente são associados na mesma entrada da memória de contextos. A Figura 6.20 apresenta a organização das memórias de contextos adotada nesta arquitetura.

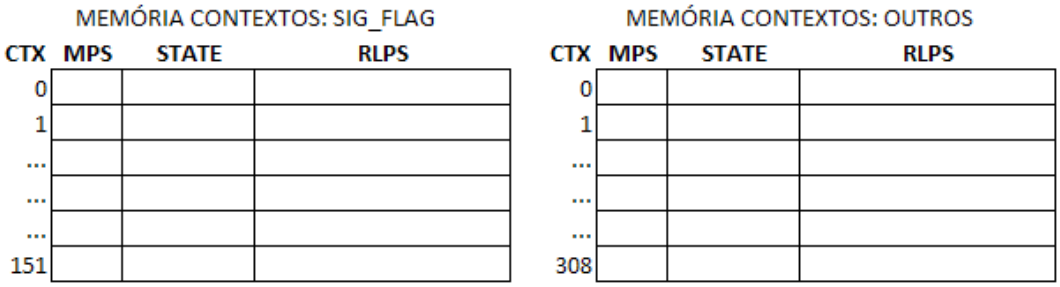


Figura 6.20: Organização das memórias de contexto.

Ambas as memórias possuem os mesmos 39 bits de tamanho de palavra. A memória dedicada aos contextos relacionados com o SE do tipo SIG_FLAG possui 152 linhas, enquanto a memória com os contextos relativos aos demais SE possui 308 linhas. No total, são utilizados 17,5 Kb entre as duas memórias, um número significativo quando comparado ao mínimo tamanho necessário que seria de 3,14 Kb se o valor do símbolo

RLPS não fosse armazenado nestas memórias, porém, esse gasto adicional se justifica pela possibilidade de eliminação de um ciclo do processo de decodificação de cada *bin*.

6.4 Máquinas de Controle

O processo de decodificação realizado pelo CABAD é caracterizado por uma lógica típica de fluxo de controle. Para cada etapa do processamento um conjunto de regras condicionais deve ser avaliada para que as ações subsequentes possam ser determinadas. Esse tipo de circuito normalmente implica circuitos de grande complexidade e custo em termos de recursos de *hardware* dedicados a lógica aleatória. Para tratar os problemas inerentes a natureza desta classe de problemas optou-se por uma abordagem de controle hierárquico, bastante utilizado nestes cenários.

A máquina de controle principal é composta por 13 macroestados, organizados de acordo com as fases do processo de decodificação apresentadas na Figura 6.1. Esses macroestados da máquina de controle principal acionam os estados nas máquinas de controle secundárias, que possuem estados especializados para suas funções. A estrutura de iteração entre os estados da máquina de controle principal pode ser observada através da Figura 6.21. A legenda inserida no retângulo da parte inferior da Figura 6.21 indica quais estados pertencem a que fase do processo de decodificação.

Os estados apresentados na Figura 6.21 são números em ordem crescente e o nome de cada estado é dado pela concatenação do prefixo “S” com o número do estado no sufixo. O primeiro estado, denominado “S0”, é um estado de espera onde o CABAD não executa nenhuma ação, apenas mantém o valor dos seus registradores internos. A transição do estado “S0” para o estado “S1” ocorre quando o CABAD recebe um sinal para capturar os parâmetros do *slice* recuperados pelo bloco de *parser* durante análise do *bitstream*; esse processo de captura dos parâmetros consume quatro ciclos, do estado “S1” até o estado “S4”. A seguir, o CABAD entra na fase de inicialização da memória de contextos, na qual serão consumidos 464 ciclos de processamento. Essa fase de inicialização da memória de contextos será controlada por uma máquina secundária enquanto a máquina principal permanece no estado “S5” aguardando o sinal de conclusão da fase de inicialização.

A fase de decodificação é composta por sete macroestados, iniciando no macroestado “S6” e terminando no macroestado “S10”. Todo macroestado desta fase é composto por um conjunto de estados nos quais é realizado o processo de decodificação dos diversos SEs. No macroestado “S6” o SE MBSKIP é decodificado e dependendo do valor obtido o processo de decodificação é desviado para o próximo macroestado ou próxima fase. Se o valor decodificado para o SE MBSKIP for igual a zero significa que o macrobloco atual não é do tipo *skip*, logo ele precisará ser decodificado e a transição de estado o levará ao macroestado “S7”. Caso contrário, se o valor do SE MBSKIP for igual a um então o macrobloco atual é do tipo *skip* e não haverá informação a ser decodificada para o macrobloco atual; assim, o processo decodificação será desviado para o estado “S11” que corresponde a fase de decodificação final do *slice*.

No macroestado “S7” o SE MBFIELD é decodificado. A função deste SE é determinar o tipo de codificação do macrobloco atual, se ele será do tipo *frame* ou do tipo *field*. Entretanto, a arquitetura desenvolvida neste trabalho só permite trabalhar com macroblocos do tipo *frame*. Após a decodificação deste SE a transição de estado levará ao estado “S8”.

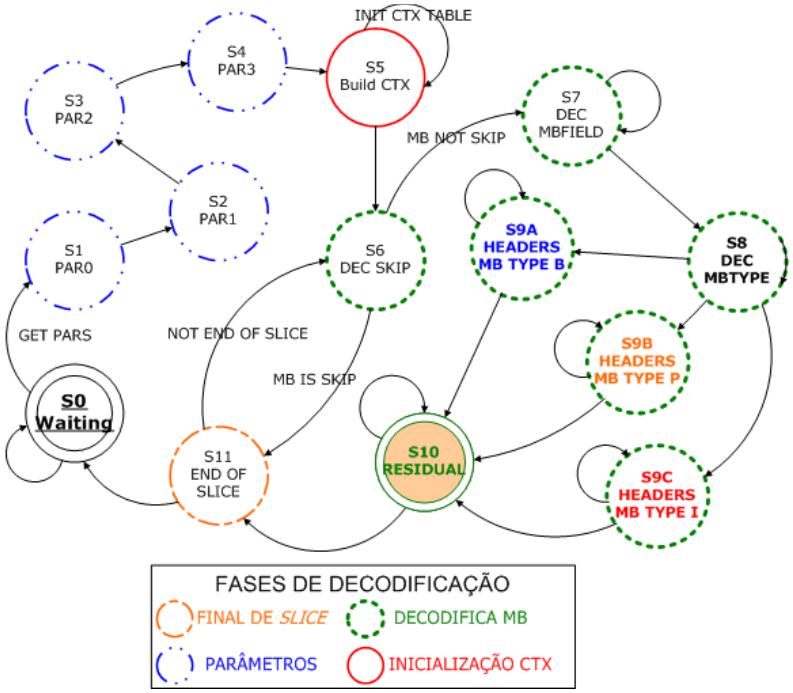


Figura 6.21: Diagrama de estado para a máquina de controle principal.

No macroestado “S8” o SE MBTYPE será decodificado. De acordo com o tipo do macrobloco obtido o fluxo de decodificação poderá seguir caminhos diferenciados. Assim, a transição do macroestado “S8” poderá conduzir a três macroestados subsequentes diferentes. A transição de estado irá conduzir ao estado “S9A” quando o macrobloco for do tipo “B” ou para o estado “S9B” quando o macrobloco for do tipo “P” ou para o estado “S9C” quando o macrobloco for do tipo “I”.

Nos macroestados “S9A”, “S9B”, “S9C” os SE com as informações de cabeçalho são decodificados. A diferença básica entre esses macroestados reside nos SEs que devem ser decodificados. Entretanto, após a decodificação dos respectivos SEs a transição de estado irá conduzir ao estado “S10” onde os SEs relacionados com a informação residual serão decodificados.

Apesar de algumas pequenas variações no que tange ao acesso aos contextos o processo de decodificação da informação residual ocorre de forma praticamente igual entre os diferentes tipos de macroblocos, ou seja, os mesmos tipos SEs ocorrem e são decodificados na mesma ordem. Entretanto, devido à quantidade de informações associadas a esses SEs a maior parte do tempo de decodificação de um macrobloco é gasto neste macroestado.

Detalhar as variações no controle do processo de decodificação para cada tipo de macrobloco e para as informações residuais serão objeto das próximas seções.

6.4.1 Decodificação de Cabeçalhos - Macroblocos do Tipo Intra

Na decodificação de macroblocos do tipo “Intra” os SEs a serem decodificados são os flags e modos de predição (INTRA_FLAG e INTRA_MODE), o padrão de codificação dos blocos 8X8 (CBP) e parâmetro de quantização do macrobloco (QPDelta). O fluxo de decodificação para esses SEs é controlado por uma máquina de estados específica que é ilustrada pela Figura 6.22.

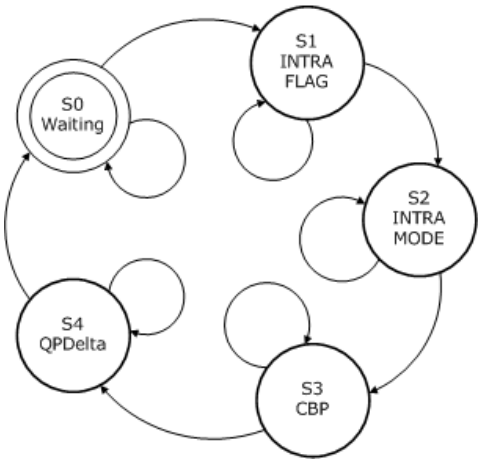


Figura 6.22: Diagrama de estados para decodificação de SE de cabeçalho em macroblocos do tipo “Intra”.

O diagrama de estados apresentado na Figura 6.22 é composto por cinco macroestados, denominados de “S0”, “S1”, “S2”, “S3” e “S4” os quais são uma representação simplificada, pois cada um dos macroestados é composto por estados que realizam as operações necessárias a decodificação do SE correspondente.

6.4.2 Decodificação de Cabeçalhos - Macroblocos do Tipo Inter

Na decodificação de macroblocos do tipo “Inter” os SEs a serem decodificadas são o tipo dos submacrobloco (SubMBTYPE), o índice do quadro de referência (REFIDX), os vetores de movimento diferenciais (MVD), o padrão de codificação dos blocos 8X8 (CBP) e parâmetro de quantização do macrobloco (QPDelta). O fluxo de decodificação para esses SEs é controlado por uma máquina de estados específica que é ilustrada pela Figura 6.23.

Um aspecto importante no que tangue a decodificação dos macroblocos do tipo “Inter”, ou seja, tipos “P” é que a lógica de controle para os SEs dos tipos “SUB-MBTYPE”, “REFIDX” e “MVD” ainda não foi implementada. Dessa forma, no estado atual, a decodificação destes tipos de macroblocos ainda não é suportada pela arquitetura desenvolvida.

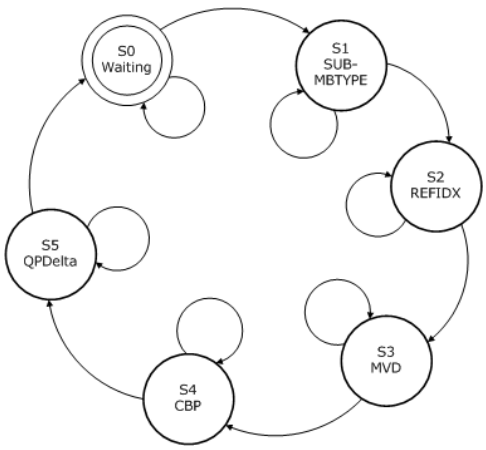


Figura 6.23: Diagrama de estados para a decodificação de SE de cabeçalho em macroblocos do tipo “Inter”.

6.4.3 Decodificação de Dados Residuais

Na decodificação dos dados residuais os SEs a serem decodificados são o flag de indicação de bloco válido (CBF), o mapa de significância composto pelo flag de indicação de coeficiente válido (SIG FLAG) e o flag de indicação de último coeficiente válido (LAST SIG), o valor do coeficiente (COEF VALUE) e o sinal do coeficiente (COEF SIGN). O fluxo de decodificação para os dados residuais é semelhante entre os macroblocos do tipo "Inter" ou "Intra", sofrendo uma variação no cálculo dos contextos que, no entanto, não alteram o fluxo de decodificação. A Figura 6.24 apresenta o diagrama da máquina de estados que controla esse processo.

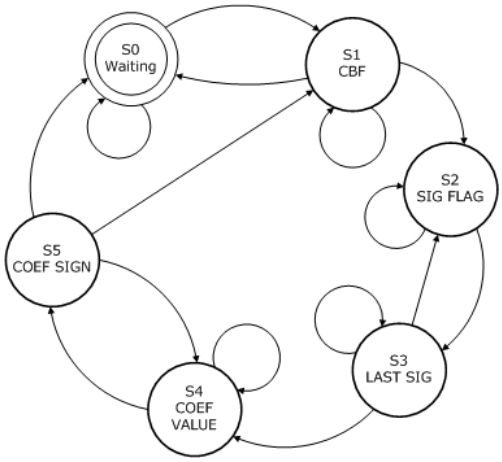


Figura 6.24: Diagrama de estados para decodificação de dados residuais.

O diagrama de estados apresentado na Figura 6.24 é composto por seis macroestados, denominados de "S0", "S1", "S2", "S3", "S4" e "S5", os quais são uma representação simplificada, pois cada um dos macroestados é composto por estados que realizam as operações necessárias a decodificação do SE correspondente. As particularidades desta máquina de estado estão relacionadas com a ordem de decodificação do mapa de significância, pois pode ocorrer um número variável de cada um dos SEs que compõem o mapa, assim como o número de coeficientes a serem decodificados varia de um bloco para outro.

6.5 Resultados de Síntese

A arquitetura desenvolvida para o CABAD foi descrita em VHDL visando à plataforma de dispositivos FPGAs. O código HDL foi sintetizado para os dispositivos FPGA XUPC2VP30 e XC4VLX40 das famílias Virtex-2P e Virtex-4, respectivamente, sendo ambos da Xilinx (XILINX, 2008). As ferramentas utilizadas para a síntese foram ISE 10.3 da Xilinx e Synplify Pro 8.5.1 da Symplicity (SYMPLICITY, 2008). A arquitetura de *hardware* para o CABAD foi simulada e validada através das ferramentas ISE 10.3 da Xilinx e ModelSim 6.0c da Mentor Graphics (MENTOR, 2008).

A Tabela 6.1 apresenta o resultado de síntese para os principais blocos da arquitetura do CABAD. Nela é possível observar que a máxima frequência de operação da arquitetura que ficou limitada à, aproximadamente, 88 MHz em razão da ligação do bloco BADE com o bloco de controle. O consumo dos recursos de *hardware* foi de, aproximadamente, 29% das LUTs, 7% dos Slice FFs, 35% dos Slices e 14% das BRAMs. Nesse cenário, é interessante observar que o número de BRAMs multiplicado pelo de número de bits disponível em cada BRAM supera o número de bits que seriam

necessários para atender as especificações das memórias utilizadas (22*18Kb=396 contra 163Kb). Esse fato ocorre devido à dimensão das memórias utilizadas não serem múltiplos inteiros das dimensões e formatos disponibilizados pelo FPGA em questão.

Tabela 6.1: Resultados de síntese da arquitetura do CABAD no dispositivo XUPC2VP30.

Blocos	Frequência (MHz)	Slices	Slice Flip Flops	4 Input LUTs	RAM Kb/BRAM
BADEs	93	834	468	1520	4,3/4
Controle	88	1394	392	2542	-
Completo	88	4784	2018	8740	~163/22

Dispositivo Virtex2P 2VP30

A Tabela 6.2 é similar a Tabela 6.1, porém nela são apresentados os resultados de síntese da arquitetura para um FPGA da família Virtex4. Os dados obtidos mostram que a frequência de operação teve um ganho significativo, porém houve um acréscimo na utilização total de recursos do *hardware*.

Tabela 6.2: Resultados de síntese da arquitetura do CABAD no dispositivo XC4VLX40.

Blocos	Frequência (MHz)	Slices	Slice Flip Flops	4 Input LUTs	RAM Kb/BRAM
BADEs	103	834	483	1570	4,3/4
Controle	99	1442	404	2630	-
Completo	99	4948	2156	9020	~163/22

Dispositivo Virtex4-XC4VLX40

6.6 Resultados de Desempenho

A etapa de avaliação de desempenho é uma tarefa complexa no desenvolvimento de *hardware*. Normalmente, são utilizados os dados de pior caso e do caso médio para avaliar o desempenho que o projeto deve atingir. Entretanto, no caso do CABAD existem alguns fatores que dificultam essa tarefa como, por exemplo, a produção de um número variável de *bins* para os diferentes SE. Além disso, o número de ocorrências para um determinado SE pode sofrer variações significativas em função do tipo de quadro, tipo de macrobloco, das ferramentas de codificação, do parâmetro de quantização e da estrutura do GOP, além de outros aspectos como a resolução do vídeo e forma de transmissão (progressivo ou entrelaçado).

Para determinar o pior caso, foi necessário determinar duas variáveis: o número máximo de ocorrências para tipo de SE em um dado macrobloco e o número máximo de *bins* que cada um destes SEs pode utilizar. Nesse sentido, cabe salientar que os macroblocos dos tipos *intra* e *inter* tratam SE diferentes. A Tabela 6.3 apresenta esses dados de pior caso para cada tipo de SE, considerando os diferentes tipos de macroblocos. A informação sobre o número máximo de *bins* para alguns tipos de SEs foi obtida através da norma de especificação do padrão H.264/AVC, porém para outros tipos de SEs como, por exemplo, MVD_L0, MVD_L1, COEF_LUM e COEF_CHR esse número não é determinado pela norma. Dessa forma, foram utilizados os valores do

número máximo de *bins* obtidos durante a fase de análise de comportamento dinâmico (descrita ao longo do Capítulo 5).

Tabela 6.3: Avaliação de pior caso para cada tipo de SE através do número máximo de *bins* por SE x número máximo de ocorrências de cada SE.

TIPO DE SE	MAX BINS	Macrobloco INTRA		Macrobloco INTER	
		OCORRÊNCIAS	BINS	OCORRÊNCIAS	BINS
MBTYPE	14	1	13	1	14
REFIDX	4	0	0	8	32
CBP	6	1	6	1	6
COEF_LUM	38	256	9728	256	9728
COEF_CHR	38	128	4864	128	4864
QP_DELTA	7	1	7	1	7
COEF_SIG	1	384	384	384	384
COEF_LAS	1	302	302	302	302
CBF	1	26	52	26	52
INTRA_FLAG	1	17	17	0	0
INTRA_MODE	3	51	153	0	0
SUB_MBTYPE	6	0	0	4	24
MVD_L0	24	0	0	32	768
MVD_L1	22	0	0	32	704
BINS TOTAL			15526		16885

De acordo com os dados apresentados na Tabela 6.3, o total de *bins* produzidos seria de 15526 para macroblocos do tipo INTRA e de 16885 para macroblocos do tipo INTER. Entretanto, se nenhum tipo de compressão fosse aplicado e as amostras de cor fossem transmitidas diretamente a soma dos bits necessários para representar a luminância e crominância totalizaria 3072 bits. Logo, o pior caso teórico não é compatível com o pior caso prático. Dessa forma, vamos considerar que o pior caso prático será dado pela combinação de duas restrições: a primeira provém da norma de especificação do padrão H.264/AVC para o perfil *main* no nível 4.0 que define o limite de 20 Mb/s para a taxa de bits no bitstream (ITU-T, 2005); a segunda provém do limite teórico ideal da compressão de entropia que é de 8 pra 1 (SHANNON, 1948). Dessa forma, o pior caso prático pode ser fixado em 160 M*bins*/s (8 x 20Mb/s).

Para determinar o caso médio foram utilizados os dados do número de *bins* produzidos a partir de cada bit do *bitstream*. Essa informação foi obtida com durante a fase de análise do comportamento do *bitstream*, discutida no Capítulo 5. A Figura 6.25 apresenta essa relação *bins/bit* para a média de todas as sequências de vídeo em todos os parâmetros de quantização avaliados. Combinando o percentual da média de *bins* produzidos para cada *bit*, apresentado na Figura 6.25, com o limite de 20 Mb/s imposto pelo padrão H.264/AVC, é possível definir valor para cada uma das resoluções de vídeo. Os dados de pior caso teórico, pior caso prático e caso médio para os diferentes tipos de macroblocos são apresentados na Tabela 6.4.

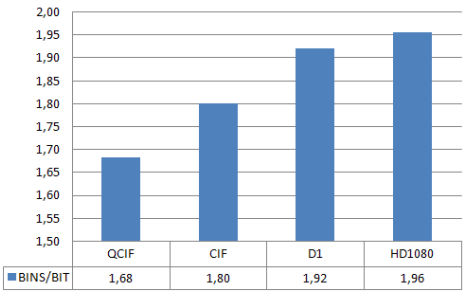


Figura 6.25: Relação entre o número de *bins* produzidos para cada bit recuperado do *bitstream* separados de acordo com a resolução do vídeo.

A Tabela 6.4 também apresenta os resultados de desempenho para as diferentes resoluções de vídeo. Esses dados foram produzidos a partir da divisão da vazão média da arquitetura, que ficou em 1,28 *bins* por ciclo. A frequência de operação requerida para o pior caso prático e para o caso médio é indicada ao lado da coluna, com a taxa de *bins* produzidos em cada uma das situações.

Tabela 6.4: Dados para pior caso teórico, pior caso prático e caso médio em quadros por segundo (indicação de *Mbins/s* produzidos e frequência necessária).

Resolução	Nro. Máx. Macro-blocos/s[1]	Pior caso teórico (M*bins*/s)		Pior caso prático		Caso Médio	
		Intra[2]	Inter[3]	M*bins*/s	Fmin[4] (MHz)	M*bins*/s	Fmin[4] (MHz)
QCIF	2970	43,98	47,83	1,94	1,52	0,41	0,32
CIF	11880	175,90	191,30	7,76	6,07	1,75	1,37
D1	48600	719,60	782,60	31,76	24,82	7,63	5,96
HDTV	244800	3624,69	3941,96	160,00	125,00	39,13	30,57

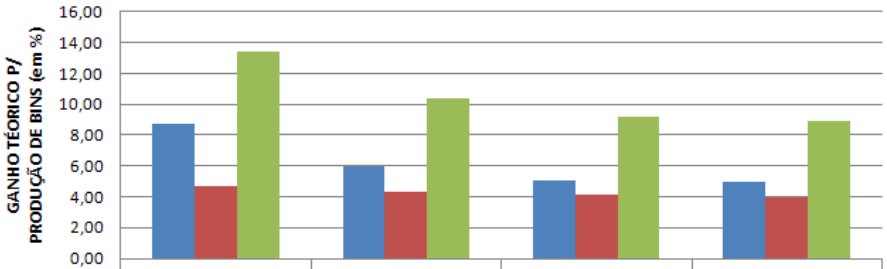
[1] Número máximo de macroblocos na taxa de 30 quadros por segundo.
[2] Macrobloco do tipo Intra.
[3] Macrobloco do tipo Inter.
[4] Frequência mínima requerida.

Analisando os dados apresentados na Tabela 6.4, é possível observar que a máxima frequência de operação fornecida pela arquitetura desenvolvida é suficiente para oferecer capacidade de processamento em tempo real para todas as resoluções analisadas, mesmo quando considerando o resultado do dispositivo mais antigo (88 MHz no XUP2CVP30). Para o pior caso prático, a arquitetura também oferece suporte, exceto para HDTV, na resolução de 1080p, na qual seria necessária uma frequência de operação de 125 MHz. Entretanto, é importante considerar que mesmo o pior caso prático é uma situação extrema e na condução dos experimentos de análise do comportamento estático e dinâmico do *bitstream,* não se observou nenhuma vez a incidência desta situação.

Considerando que a arquitetura desenvolvida foi baseada na prospota apresentada por Yu e He (2005), combinada com ténicas de aceleração do processo de decodificação criadas a partir das observações sobre o comportamento estático e dinâmico do *bitstream,* torna-se interessante analisar o ganho sobre a taxa de processamento oferecido pelas inovações apresentadas frente à proposta original. A Figura 6.26 relaciona o ganho na capacidade de processamento introduzido pelas duas principais contribuições deste trabalho no bloco de decodificação aritmética, para cada uma das resoluções analisadas. As contribuições mencionadas são: adoção de quatro módulos de decodificação no ramo *Bypass* e a utilização do esquema de aceleração na decodificação do mapa de significância.

Na Figura 6.26 é possível observar que a técnica de aceleração da decodificação do mapa de significância oferece um pontencial de ganho maior que a utilização dos quatros módulos de decodificação no ramo *bypass*. Contudo, esse ganho diminui significativamente com o aumento da resolução do vídeo. No total, o ganho potencial

com a utilização das duas técnicas combinadas atingem, aproximadamente, 8,9% para vídeos na resolução HD1080p.



	QCIF	CIF	D1	HD1080
MAPA SIG.	8,75	6,01	5,06	4,97
4 BYPASS	4,68	4,34	4,13	3,99
Total	13,43	10,35	9,19	8,96

Figura 6.26: Ganho pontêncial com as técnicas de aceleração utilizadas.

6.7 Processo de Validação

No ciclo de desenvolvimento de circuitos integrados o processo de validação possui o maior custo, chegando a atingir 70% do tempo de projeto. Essa informação serve de indicativo sobre o desafio envolvido nesta etapa. A abordagem adotada para tentar minimizar o tempo necessário foi realizar uma validação hierárquica e incremental, ou seja, foram executadas diversas etapas de validação de acordo com a complexidade e nível de abstração dos blocos desenvolvidos.

Na primeira etapa os blocos de menor nível de abstração foram validados isoladamente, através de estímulos gerados a partir das definições presentes na norma de especificação do padrão H.264/AVC. Esses estímulos foram injetados individualmente em cada um dos módulos e a verificação foi realizada através do acompanhamento direto das formas de onda no simulador.

Na segunda etapa, os blocos foram agrupados de acordo com sua funcionalidade e sua validação funcional foi realizada através da comparação matemática com casos de uso. Entretanto, para efetuar a validação funcional baseada em caso de uso é necessário, primeiramente, obter os estímulos que devem ser injetados e os resultados esperados para que a comparação possa ser efetuada. Porém, para obter os dados necessários a validação é necessária a construção de um modelo de referência. Neste caso, *software* de referência (Surking, 2009) foi utilizado para esse propósito por se tratar da implementação que valida o padrão H.264/AVC. Contudo, a extração de dados a partir do *software* de referência para validação de blocos individuais é uma tarefa complexa, dada a forma de implementação deste *software*. Portanto, foi necessário desenvolver um conjunto de rotinas auxiliares que foram acopladas ao modelo de referência para possibilitar a extração dos dados desejados. A Figura 6.27 ilustra como o processo de extração dos dados foi realizado.

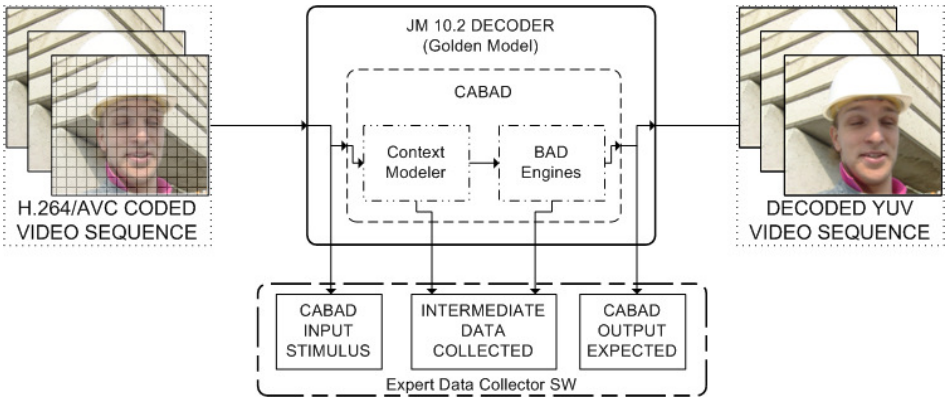


Figura 6.27: Processo de extração de dados para a validação funcional dos blocos individuais e da arquitetura completa.

O processo de extração dos dados de estímulos e resultados esperados foi realizado sobre as mesmas sequências de vídeo padronizadas discutidas no Capítulo 5. Aliás, tanto os dados para validação quanto os dados para análise de comportamento estático e dinâmico foram obtidos no mesmo momento, pois as rotinas responsáveis por essas tarefas foram codificadas através do mesmo conjunto de rotinas auxiliares. Essa abordagem permitiu reduzir significativamente o tempo despendido, dado que o conjunto de sequências de vídeo foi processado uma única vez. Além disso, propiciou consistência entre os dados apresentados no processo de análise do comportamento com o processo de validação.

Após a obtenção dos dados para estímulos e resultados esperados, a segunda etapa de validação funcional dos blocos desenvolvidos seguiu o fluxo ilustrado pela Figura 6.28. Dentro de um arquivo de *test-bench* os dados de estímulos coletados foram injetados no bloco a ser validado (*Design Under Test* – DUT). O bloco é estimulado e os resultados produzidos são capturados e armazenados em um arquivo, o qual é, posteriormente, comparado contra um arquivo de resultados esperados, sendo que o resultado desta comparação indica se o DUT apresenta conformidade funcional com o modelo de referência.

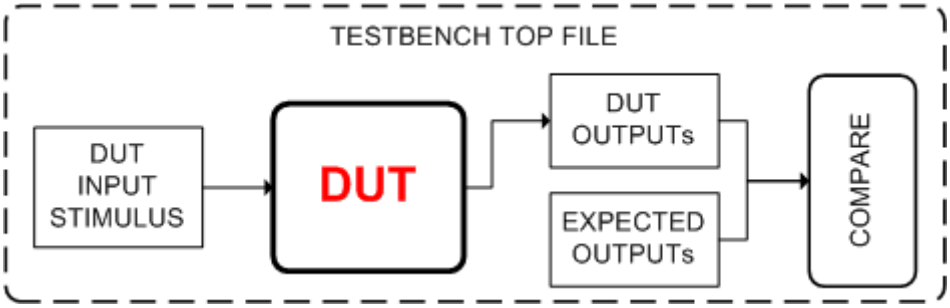


Figura 6.28: Processo de validação dos módulos do CABAD.

7 COMPARAÇÕES COM TRABALHOS DA LITERATURA

Este capítulo apresenta comparações com vários trabalhos relacionados encontrados na literatura. As primeiras seções fornecem uma breve descrição de cada trabalho, com suas características principais. Enquanto, na última seção, são apresentados os resultados comparativos entre este trabalho e os demais.

7.1 Trabalhos Correlacionados

7.1.1 Trabalho de Yu e He

O trabalho Yu e He (2005) é um dos primeiros relatos sobre desenvolvimento de arquitetura de *hardware* dedicados ao CABAD disponíveis na literatura. O artigo que descreve este trabalho teve grande importância para essa dissertação, pois além de conter uma descrição simplificada dos processos de codificação e decodificação do CABAC, também introduziu alguns conceitos como, por exemplo, agrupamento de contexto e utilização de múltiplos módulos de decodificação aritmética que, posteriormente, foram utilizados por outros autores.

O ponto forte do trabalho está no detalhamento sobre a organização dos múltiplos módulos de decodificação binária aritmética e na apresentação de análises de comportamento estático sobre o processo de decodificação, que fornece dados sobre a ocorrência de cada tipo de SE e da utilização dos contextos. O ponto fraco fica na organização das seções do próprio artigo, principalmente pela ausência de uma seção sobre resultados experimentais e/ou resultados de síntese.

Alguns dados de síntese e desempenho como, por exemplo, a tecnologia alvo (0,18um), área (0,3 mm^2), período do caminho crítico (6,7ns) e número médio de ciclos para decodificação de um macrobloco (500) são apresentados, porém faltam detalhes essenciais como as sequências utilizadas para avaliar o número médio de ciclos, o conjunto de parâmetros de codificação adotados e o número de *gates* equivalentes não são fornecidos.

7.1.2 Trabalho de Chen, Chang e Lin

A proposta de Chen, Chang e Lin (2005) visa um sistema de gerenciamento de memória eficiente para o CABAD. A tecnologia alvo foi TSMC 0,13um e a implementação utilizou 138K *gates* equivalentes atingindo 200MHz de frequência máxima de operação.

O trabalho apresenta os dados sobre o número médio de ciclos para decodificar cada um dos tipos de macroblocos (I, P e B), porém sem nenhuma referência sobre quais as sequências de vídeo foram utilizadas para determinar o caso médio, nem os parâmetros de codificação empregados nestes vídeos. Além disso, foram utilizados apenas 12 quadros para validar a arquitetura e nas conclusões os autores afirmam que a arquitetura possui desempenho suficiente para decodificar vídeos na resolução CIF em tempo real.

7.1.3 Trabalho de Kim e Park

O trabalho de Kim e Park (2006) propõem a aceleração do processo de decodificação através da exploração da execução especulativa. A proposta é utilizar dois módulos de decodificação binária aritmética do tipo *regular* de forma especulativa, baseado num estudo de caso sobre a incidência de alterações nos valores de variáveis relacionadas a dependência de dados.

O ponto forte do trabalho está na frequência de operação atingida, pois a arquitetura dos módulos de decodificação e o método de balanceamento dos estágios do *pipeline* (*retiming*) adotados permitem manter atraso do caminho critico extremamente baixo, apenas 3.3 ns. Entretanto, o ponto fraco do trabalho está na ausência de alguns dados fundamentais, como o número de *gates* equivalentes ou a área. Além disso, o trabalho peca no número de amostras utilizadas para avaliar a efetividade do método proposto (apenas duas sequências de vídeo). Dessa forma, não é possível determinar se a taxa de processamento de 0,41 *bins*/ciclos é válida para a média das sequências de vídeo que normalmente seriam utilizadas.

7.1.4 Trabalho de Eeckhaut et al.

A principal proposta do trabalho de Eeckhaut et al. (2006) é uma estratégia de otimização para o laço de renormalização presente nos módulos de decodificação binária aritmética dos tipos *regular* e *terminate*. Além disso, uma reorganização das operações de soma, comparações e deslocamento que fazem parte do módulo *regular* são propostas como forma de redução do atraso do caminho crítico.

A substituição do processo iterativo de renormalização por mecanismos de identificação do final da cadeia de zeros (LZA, do inglês: *Leading Zero Anticipatory*) são o ponto forte do trabalho. A implementação restrita aos módulos de decodificação binária aritmética apresenta frequência de operação de, aproximadamente, 70 MHz e utilização de 1287 elementos lógicos de um FPGA Stratix S25, porém, por não contemplar a arquitetura completa do CABAD, não permite fazer qualquer tipo de estimativa sobre a eficácia da arquitetura proposta para a decodificação de sequências de vídeos reais, constituindo o principal ponto fraco.

7.1.5 Trabalho de Yang et al.

O trabalho de Yang et al. (2006) propõe uma arquitetura VLSI para o CABAD com enfâse na alta vazão. O trabalho apresenta uma estratégia de redução do número de bits para representação do SEs na memória de macroblocos. Além disso, introduz uma técnica para redução do número de ciclos gasto na decodificação do mapa de significância e do valor dos coeficientes de transformada.

Os resultados de síntese indicam que a arquitetura desenvolvida foi sintetizada para a tecnologia TSMC 0,18 um atingindo 120 MHz na máxima frequência de operação com a utilização de, aproximadamente, 83,2 K *gates* equivalentes, incluindo todos os

bits gastos em memória. O desempenho para o caso médio é de 462,4, 307,73 e 253,56 ciclos por macroblocos dos tipos “I”, “P” e “B”, respectivamente.

7.1.6 Trabalho de Bingbo et al.

O trabalho de Bingbo et al. (2007) propõe uma arquitetura VLSI de alto desempenho para o CABAD utilizando uma combinação de unidades em *software* com outras unidades em *hardware*. Para simplificar a complexidade do *hardware*, os autores optaram por implementar o cálculo do endereço de contexto através de *software*, sendo que a comunicação entre as partes de *hardware* e *software* é realizada uma única vez para cada SE, a fim de evitar minimizar atrasos devido a comunicação.

Os resultados de simulação indicam que a arquitetura foi sintetizada para a tecnologia Faraday UMC 0,18 um e sua implementação utilizou 25K *gates* equivalentes atingindo 160MHz de frequência máxima de operação. Entretanto, o pequeno conjunto de sequências de vídeo utilizadas para avaliar o número médio de ciclos necessários para decodificar quadros do tipo *intra* é um dos pontos fracos do trabalho. Além disso, há o problema da ausência de informações sobre os parâmetros de codificação empregados nas sequências de vídeo utilizadas. Contudo, a falha mais gritante do trabalho é que os autores afirmam que a arquitetura proposta é capaz de decodificar vídeos na resolução HD1080i (entrelaçado) operando na frequência de 140MHz, porém os dados de desempenho apresentados se referem a decodificação de quadros do tipo *intra*, ou seja, os autores consideram HD1080i como se fossem vídeos codificados apenas com a predição intraquadros, ao invés de considerar como vídeos entrelaçados.

7.1.7 Trabalho de Zhang et al.

A proposta de Zhang et al. (2007) visa a alta performance para possibilitar a decodificação de vídeo de alta definição em tempo real. Dentre as inovações apresentadas neste trabalho estão à utilização de banco de registradores para grupos de SEs visando reduzir os gargalos ocasionados pela necessidade de recuperação de contexto a partir da memória externa, durante a etapa de decodificação aritmética, e a utilização de múltiplos módulos de decodificação concatenados no caminho crítico (16 em cada no módulo *regular* e 16 no módulo *bypass*).

Os resultados de síntese apresentados indicam que a tecnologia alvo foi ASIC 0,18 um e a implementação utilizou 42K *gates* equivalentes e a arquitetura atingiu 45MHz na máxima frequência de operação. A grande vantagem da proposta é que qualquer SE pode ser decodificado em um único ciclo de relógio e o consumo de bits do *bitstream* ocorre na mesma taxa em que eles são recebidos. Dessa forma, o cálculo da frequência de operação necessária para atender o pior caso é simplificado.

7.1.8 Trabalho de Mei-hua et al.

O trabalho de Mei-hua et al. (2007) apresentam algumas inovações e otimizações para a arquitetura do CABAD. As contribuições mais significativas do trabalho são o mecanismo de gerenciamento do bitstream e a nova estratégia de organização para as ROMs rLPS e transState, que combinadas permitem reduzir um ciclo de acesso a cada *bin* decodificado.

O alvo da proposta foram plataformas de FPGA, sendo que a arquitetura foi sintetizada para o dispositivo StratixTM EP1S40F780C da Altera (ALTERA, 2009). A implementação utilizou 7020 LUTs e atingiu 80 MHz na máxima frequência de

operação. O ponto fraco do trabalho é a falta de informações sobre a taxa de processamento e a não utilização de sequências de vídeo para avaliar o desempenho da arquitetura.

7.1.9 Trabalho de Chen e Lin

A proposta de Chen e Lin (2007) apresenta uma arquitetura de *hardware* para o CABAD com enfase no alto desempenho para atender as restrições de tempo real em sequências de vídeo de alta definição (HD1080). O trabalho é baseado na abordagem de decodificação paralela para produzir um número variável de *bins* por ciclo e utiliza um mecanismo de aceleração para a recuperação de contextos. As decisões arquiteturais são baseadas em uma análise sobre o comportamento do processo de decodificação que apresenta alguns dados como, por exemplo, incidência de *bins* para grupos de SEs nos três tipos de macroblocos.

Os resultados de síntese indicam a opção pela tecnologia TSMC 0,13um. O custo do *hardware* (em *gates* equivalentes) e a máxima frequência de operação para a arquitetura são apresentados, separando os resultados da arquitetura completa do núcleo dos módulos de decodificação aritmética binária. A arquitetura completa consome, aproximadamente, 40,7 K *gates* equivalentes atingindo 137 MHz de frequência máxima, enquanto o núcleo da decodificação binária aritmética utiliza, aproximadamente, 11,5K *gates* equivalentes, atingindo 188 MHz na máxima frequência de operação.

Os resultados experimentais apontam para uma vazão média de 0,8 *bins* por ciclo, com 194 ciclos para a decodificação de um macrobloco inteiro considerando o conjunto de sequências de vídeo utilizadas no experimento. Na tabela 4 são apresentadas as frequências de operações mínimas que são necessárias para atender a decodificação em tempo real para diferentes resoluções, enquanto a tabela 5 apresenta a vazão da arquitetura proposta em termos de macroblocos por segundo e faz comparações com outro trabalho. Essas duas tabelas apresentam um grave erro nos cálculos, pois os autores afirmam que a arquitetura, operando na frequência máxima, pode produzir 740489 MB/s. Entretanto, com uma média de 194 ciclos por macrobloco é possível produzir apenas 706185 MB/s (137.000.000 / 194). Com isso, os dados sobre vazão e desempenho mínimo ficam incorretos – tudo indica que os autores consideraram 137 MHz como se fosse 137 * 1024 * 1024 que dividido por 194, obteve-se os 740489 MB/s.

7.1.10 Trabalho de Yi e Park

A proposta arquitetural de Yi e Park (2007) busca a alta frequência de operação, visando a decodificação de vídeo de alta definição em tempo real. O trabalho apresenta uma discussão sobre os problemas encontrados no fluxo de decodificação convencional e, então, introduz técnicas para minimizar problemas de bolhas no *pipiline*, sendo o repositório de contextos uma das inovações apresentadas.

Os resultados de síntese indicam que a arquitetura desenvolvida foi sintetizada para a tecnologia 0,18 um, atingindo 225 MHz na máxima frequência de operação, com utilização de, aproximadamente, 81,2 K *gates* equivalentes, sem considerar os bits gastos em memórias. O desempenho para o caso médio foi de 0,25 *bins* por ciclo, baseado nas sequências de vídeo utilizadas no estudo de caso; produzindo um total de 56 M*bins*/s.

7.2 Comparação com a Arquitetura Proposta

Os trabalhos discutidos na seção 7.1 são considerados as principais referências disponíveis em termos de arquiteturas de *hardware* destinadas ao CABAD. Entretanto, a comparação de desempenho entre esses trabalhos não é trivial, pois os critérios de avaliação dos resultados adotados variam de forma significativa de autor para autor. Além disso, a maioria utiliza dados baseados em observações de experimentos realizados, porém, nos trabalhos não é possível encontrar informações suficientes para compor todo o cenário de avaliação com os parâmetros que podem interferir no processo.

Para manter o processo de avaliação imparcial e apresentar as diferenças entre as propostas, foram elaboradas duas Tabelas (Tabela 7.1 e Tabela 7.2). A Tabela 7.1 traz dados referentes à tecnologia adotada, à máxima frequência de operação atingida, ao número médio de *bins* por ciclo, à vazão por segundo (em M*bins/s*) e o número médio de ciclos por cada tipo de macrobloco. A tabela contém algumas células sem o preenchimento adequado, pois os trabalhos em questão não contemplavam tais dados. Em outros casos, a informação está preenchida, mas não foi extraída diretamente do trabalho, tendo sido calculada através de estimativas, com base na informação disponível. Essas peculiaridades estão todas indicadas no rodapé da tabela.

Tabela 7.1: Comparação de desempenho entre a arquitetura desenvolvida e as propostas encontradas na literatura.

Trabalho	Tech	FMAX (MHz)	Bins /Ciclo	Vazão (M*bins*/s)	Ciclos por Macrobloco			
					MBI	MBP	MBB	AVG
(CHEN, CHANG, LIN, 2005)	TSMC 0.13u	200	0,40#	80,21#	1661,00	576,00	328,00	413,77#
(CHEN, LIN, 2007)	TSMC 0.13u	137	0,80	109,60	309,00	143,00	130,00	194,00
(MEI-HUA, et al., 2007)	StratixTM EP1S40F780C	80,5	NI					
(YU, HE, 2005)	ASIC 0,18u	149,25	0,32	47,16	0,00	0,00	0,00	500,00
(ZHANG, et al., 2007)	ASIC 0,18u	45,00	1,00#	45,00#	NI			190,66#
(YANG, et al., 2006)	TSMC 0.18u	120,00	NI		463,00	308,00	254,00	269,97#
(YI, PARK, 2007)	ASIC 0,18u	225,00	0,25	55,26#	NI			
(BINGBO, et al., 2007)	UMC 0,18u	160,00	1,28#	204,8#	NI			
(KIM, PARK, 2006)	ASIC 0,18u	303,03	0,41	124,24#	NI			
(EEACKHAUT, et al., 2006)	FPGA STRATIX II S25	71,97	NÃO SE APLICA (APENAS BADEs)					
(ZHENG, et al., 2007)	NI	100,00	0,24#	24,00#	1197,00	471,00	168,00	NI
Arquitetura Proposta	FPGA Virtex-4 XC4VLX40	99,00	1,28	126,72	404,40	NÃO SE APLICA		

#: Informação não fornecida diretamente, porém foi estimada a partir de dados encontrados no trabalho.
NI: Dado não informado pelos autores e o trabalho não apresentava os subsídios para que pudesse ser estimado.

Na Tabela 7.1 existem dois trabalhos com o texto "NÃO SE APLICADA". No caso de Eeackhaut et al. (2007), essa informação aparece pois o trabalho não implementa o decodificador completo, ele fica restrito aos módulos de decodificação aritmética. Dessa forma, não é possível obter as outras informações da tabela. Quanto a arquitetura desenvolvida neste trabalho as informações sobre o número médio de ciclos para decodificar quadros dos tipos P e B não estão disponíveis, pois no momento só é possível decodificar quadros do tipo intra. Outro dado que merece destaque é a vazão de nossa arquitetura frente às demais. Mesmo operando em uma frequência inferior, por se tratar de uma solução para FPGA concorrendo contra soluções em ASIC, a nossa proposta obtém a segunda maior vazão, perdendo apenas para a proposta de Bingbo et

al. (2007) e, mesmo assim, vale salientar que essa vazão atribuída ao trabalho de Bingbo et al. (2007) foi calculada por nós de forma estimada, pois não foi informada no trabalho original.

A Tabela 7.2 tem o objetivo de apresentar as diferenças em termos do consumo de recursos computacionais para implementar cada uma das propostas na sua respectiva tecnologia alvo. Além disso, são apresentadas a resolução que cada proposta visa atender e qual a mínima frequência que cada uma pode operar para atingir seu objetivo. Novamente, devido à falta de informações padronizadas, algumas células não são informadas ou as informações apresentadas no trabalho em questão não continham o detalhamento adequado. Todas essas situações especiais possuem indicações e a respectiva descrição no rodapé da tabela.

Tabela 7.2: Comparação de custo entre a arquitetura desenvolvida e as propostas encontradas na literatura.

Trabalho	Tech	RECURSOS		Resolução Atingida?	Frequência Mínima (MHz)
		QTD.	TIPO		
(CHEN, CHANG, LIN, 2005)	TSMC 0.13u	138,3 K	Gates	CIF/HD1080[2]	4,91/101,29[2]
(CHEN, LIN, 2007)	TSMC 0.13u	40,5 K	Gates	HD1080	45,00[4]
(MEI-HUA, et al., 2007)	StratixTM EP1S40F780C	7020	LUTs	HD[3]	NI
(YU, HE, 2005)	ASIC 0,18u	0,3	mm^2	D1/HD1080[2]	24,3/122,4[2]
(ZHANG, et al., 2007)	ASIC 0,18u	42 K	Gates	HD1080	42,20
(YANG, et al., 2006)	TSMC 0.18u	83,2 K	Gates	HD1080i	66,08[#]
(YI, PARK, 2007)	ASIC 0,18u	81,2 K	Gates	HD[3]	NI
(BINGBO, et al., 2007)	UMC 0,18u	25,22 K	Gates	HD1080i[1]	NI
(KIM, PARK, 2006)	ASIC 0,18u	NI			
(EEACKHAUT, et al., 2006)	FPGA STRATIX II S25	1287	LE	NÃO SE APLICA (APENAS BADEs)	
(ZHENG, et al., 2007)	NI	NI		CIF	NI
Arquitetura Proposta	FPGA Virtex-4 XC4VLX40	9020	LUTs	HD1080	31,00

#: Informação não fornecida diretamente, porém foi estimada a partir de dados encontrados no trabalho.

NI: Dado não informado pelos autores e o trabalho não apresentava os subsídios para que pudesse ser estimado.

[1]Autores afirmam que arquitetura trabalha com HD1080i (entrelaçado), porém só apresentam dados de vídeos codificados com intraquadros.

[2]Autores afirmam que arquitetura suporta a resolução mais baixa, mas através de estimativas é possível verificar que suporta resoluções mais elevadas.

[3] Autores se limitam a afirmar que arquitetura atinge desempenho para alta definição (HD) sem maiores detalhes.

[4] Autores cometeram um erro de cálculo, na verdade a frequência mínima seria de 47,49MHz.

Analisando as informações apresentadas na Tabela 7.2, é possível observar grande variação entre o consumo dos recursos de *hardware* de cada uma das propostas. A principal justificativa para essa diferença é que alguns autores incluem na contagem o custo dos blocos de memória e outros não. Além disso, existem diferenças de tecnologia e de ferramentas de síntese adotadas para cada proposta. Quanto a resolução atingida, algumas apenas citam nas suas conclusões que oferecem suporte para uma determinada resolução, enquanto outros mostram isso através de seus experimentos.

A partir da Tabela 7.2 é possível observar que nossa proposta necessita da menor frequência de operação para decodificar sequências de vídeo na resolução HD1080. Entretanto, deve-se considerar que nossa arquitetura ainda não contempla macroblocos dos tipos P e B, os quais, tipicamente, consumem maior tempo de decodificação e quando implementados deverão baixar a vazão de nossa arquitetura e, consequentemente, aumentar a mínima frequência necessária para decodificar a resolução HD1080p.

8 CONCLUSÕES

Esta dissertação apresentou todo o processo de análise, desenvolvimento e avaliação de uma arquitetura de *hardware* dedicada ao processo de decodificação binária aritmética adaptativa ao contexto, que é um dos métodos de codificação de entropia definidos no padrão H.264/AVC. Ao longo do texto, foram discutidos conceitos sobre compressão de dados e vídeo digital, características do padrão H.264/AVC, o estudo detalhado sobre o comportamento do conjunto de algoritmos que fazem parte do CABAD, uma análise extensiva sobre o comportamento do *bitstream* para diversos cenários de codificação, além da arquitetura de *hardware* desenvolvida.

Um capítulo inteiro foi dedicado para discutir, em detalhes, as características peculiares do extenso conjunto de algoritmos que fazem parte da definição do processo de codificação de entropia realizado pelo CABAC. Um relato minucioso e necessário, dada a complexidade inerente ao tema e escassez de obras na literatura que abordem essa questão no nível de detalhes adequados para fornecer ao leitor uma ampla visão dos desafios envolvidos no processo de desenvolvimento de um *hardware* dedicado para essa finalidade.

As diferenças entre as diversas propostas encontradas na literatura, quanto a melhor abordagem para resolver as limitações de desempenho inerentes ao CABAD, motivaram um extenso processo de análise sobre o comportamento do *bitstream* produzido pelo CABAD a fim de identificar características de natureza estática ou dinâmica que fornecessem subsídios para a definição de uma solução arquitetural inovadora, visando a resolução de alguns dos desafios relacionados à dependência de dados e à eficiência do processo de decodificação. Esse estudo resultou na análise de 55 sequências de vídeo padronizadas, distribuídas entre quatro resoluções diferentes e codificadas com sete combinações diferentes de pares de parâmetros de quantização. Os dados obtidos com os processos de codificação e decodificação das 880 amostras de vídeo tratadas são inéditos na literatura.

Aproveitando o processo realizado para obter os dados comportamentais do CABAD, foi realizada uma etapa de avaliação adicional, na qual todas as amostras foram novamente codificadas, porém, trocando o método de codificação de entropia. Ao invés do CABAC foi utilizado o CAVLC. Com isso, foram obtidos dados que permitiram avaliar as taxas de compressão apresentadas por cada um destes métodos de codificação. Novamente, dados comparativos para essa gama de cenários de codificação não encontrados na literatura e as observações provenientes desta avaliação ajudam a desmistificar a eficiência entre esses métodos de codificação, indicam que os

percentuais de ganho encontrados na literatura não correspondem a casos médios para grandes conjuntos de amostras.

A arquitetura do CABAD desenvolvida em *hardware* foi baseada nos dados oriundos da etapa de análise sobre o comportamento estático e dinâmico do *bitstream*, além de observações sobre as diversas propostas encontradas na literatura. Dessa forma, os blocos da arquitetura desenvolvida combinam técnicas próprias com estratégias encontradas na literatura. Algumas contribuições inovadoras introduzidas pela arquitetura proposta merecem destaque como, por exemplo:

- A utilização de quatro módulos de decodificação no ramo *bypass*;
- A remoção da ROM (com os valores pré-cálculos para a variável rLPS) do interior do módulo *regular* para a memória de contextos;
- A decodificação do mapa de significância em pares de SEs (SIG_FLAG e LAST_FLAG) no mesmo ciclo;
- A estratégia de armazenamento para os SEs que compõem a vizinhança do macrobloco e permitem otimizar o tempo de acesso e a comunicação com a memória de macroblocos.

Algumas destas inovações, como a eliminação da ROM para os valores de rLPS, provocaram um aumento significativo no tamanho da memória de contextos, porém o ganho relacionado à redução de ciclos adicionais necessários ao acesso desta ROM mostraram-se um bom compromisso entre consumo de recursos e eficiência. Outras decisões como, por exemplo, as simplificações nas operações dos módulos de decodificação binária aritmética, especialmente a eliminação do laço de renormalização no interior do módulo *regular*, propiciaram uma redução significativa no número de ciclos necessários para a decodificação de todo SE, sem acarretar custo adicional representativo.

Devido uma decisão de projeto, os SEs foram armazenados na memória de macrobloco com sua representação original, isto é, sem utilizar formas reduzidas de representação que atendam apenas os interesses do CABAD visando à utilização desta memória de macroblocos como única memória de macroblocos para todos os blocos do sistema de decodificação de vídeo. Dessa forma, o desperdício de alguns bits de memória em relação ao mínimo necessário para o processamento do CABAD é, na verdade, uma grande economia do ponto de vista do sistema de decodificação completo. Além disso, a estratégia de redução do número de acessos à memória de macroblocos através da buferização dos SE que estão sendo decodificados para utilização como vizinhos da esquerda do próximo macrobloco a ser decodificado, permitem uma redução significativa na largura de banda necessária para a comunicação do CABAD com a memória de macroblocos.

Os resultados de síntese para a arquitetura desenvolvida, considerando à plataforma FPGA, mostram que a arquitetura atinge desempenho suficiente para decodificar sequências de vídeos na resolução HD1080 em tempo real. A máxima frequência de operação obtida para a arquitetura foi de 99 MHz, porém, os dados apresentados mostram que, no caso médio, é possível executar a decodificação de sequências de vídeo na resolução HD1080 em tempo real operando na frequência de, aproximadamente, 31 MHz.

A comparação com os trabalhos encontrados na literatura mostra que a arquitetura desenvolvida é competitiva em termos de frequência de operação, mesmo competindo com uma síntese voltada para FPGA contra soluções sintetizadas para ASIC. Além disso, o desempenho apresentado pela arquitetura atingiu a segunda melhor vazão, porém há de se considerar que o trabalho ainda não trata tipos de macroblocos P e B, portanto, esse resultado poderá sofrer uma alteração quando esses dois tipos de macroblocos foram suportados.

Os desafios encontrados e discutidos ao longo deste trabalho servem para destacar a complexidade e magnitude dos problemas envolvidos pelo tema. Contudo, o esforço desenvolvido apresentou contribuições visando facilitar a compressão do assunto e abordou novas estratégias para desenvolvimento de propostas alternativas para arquiteturas de *hardware* dedicadas a tarefa de decodificação de entropia. Os resultados obtidos fornecem os subsídios para desenvolvimento de novas pesquisas com intuito de aperfeiçoar a eficiência da proposta arquitetural que foi apresentada.

8.1 Trabalhos Futuros

A lista de tarefas classificadas como trabalho em andamento e/ou trabalhos futuros é bastante extensa. Primeiramente, é necessário concluir o desenvolvimento do gerenciamento de contextos e do processo de *anti-binarização* para que seja possível decodificar os SEs relacionados aos macroblocos dos tipos P e B, sendo eles: SUBMBTYPE, REFIDX, e MVD. Dessa forma, será possível decodificar todos os tipos de macroblocos previstos pelo padrão H.264/AVC, perfil *main* no nível 4.0.

Após concluir essa etapa de desenvolvimento será necessário realizar um conjunto de validações com sequências de vídeo padronizadas para verificar a funcionalidade dos novos blocos desenvolvidos. Logo após, deverá ser realizado um levantamento dos impactos para implementação do suporte a vídeos entrelaçados e com ordenamento de codificação dos quadros do tipo MBAFF.

Uma vez que o suporte para as funcionalidades completas do perfil *main*, nível 4.0 sejam suportadas, é necessário realizar um novo processo de comparação com as propostas existentes e verificar a conformidade com as taxas de desempenho necessárias, além de avaliar o compromisso entre custo do *hardware* e a eficiência do processo de decodificação.

Concluída a etapa de desenvolvimento, ajustes e melhorias do perfil *main* nível 4.0, então deverá ser avaliado qual o rumo a seguir para suportar novas funcionalidades do padrão H.264/AVC. As opções são as mais diversificadas possíveis, da simples adição de suporte a alta fidelidade (9 – 12 bits por componente de cor) até questões de escalabilidade e multivisão. Além disso, existe alternativas como a exploração de arquiteturas de *hardware* para versões alternativas do CABAC que prometem algum ganho adicional na taxa de compressão, porém acarretam num acréscimo significativo da complexidade computacional.

REFERÊNCIAS

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alta Performance Dedicadas à Compressão de Vídeo Segundo o Padrão H.264.** 2007. 173 f. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

ALTERA. **APEX 20K Devices:** System-on-a-Programmable-Chip Solutions. Disponível em: <http://www.altera.com/products/devices/apex/apx-index.html>. Acesso em: out. 2008.

ASHENDEN, P. **The Student's Guide to VHDL**. San Francisco: Morgan Kaufmann, 1998.

BIN, G.; WEI-DONG, W.; YU-LI, S.; HONG, Z. A High Speed CABAC Algorithm Based on Probability Estimation Update. **In: Image and Graphics, 2007. ICIG 2007**. Fourth International Conference on, pp. 195-199. ago. 2007.

BINGBO, L.; DING, Z.; JIAN, F.; LIANGHAO, W.; MING, Z. "A high-performance VLSI architecture for CABAC decoding in H.264/AVC". In: **ASICON '07. 7th International Conference on**, pp. 790-793, Out. 2007.

BHASKARAN, V.; KONSTANTINIDES, K. Image and Video Compression Standards: Algorithms and Architectures. 2. ed. **Boston: Kluwer Academic Publishers**, 1997.

BLOODSHED SOFTWARE. **DEV C++**: Providing Free Software to the Internet Community. Disponível em: <http://www.bloodshed.net/devcpp.html>. Acesso em: abr. 2008.

CHEN, J.; CHANG, C.; LIN, Y. A *Hardware* Accelerator for Context-Based Adaptive Binary Arithmetic Decoding in H.264/AVC. **IEEE International Symposium on Circuits and Systems (ISCAS)**, vol. 5. pp. 4525-4528. mai. 2005.

CHEN, J.; LIN, Y. A High-Performance Hardwired CABAC Decoder. In: **Acoustics, Speech and Signal Processing, 2007**. ICASSP 2007. IEEE International Conference on, **Processing**…, vol. 2., pp. II-37-II-40. abr. 2007.

DEPRÁ, D. A.; ROSA, V. dos S.; BAMPI, S. A novel hardware architecture design for binary arithmetic decoder engines based on bitstream flow analysis. XXI Symposium on Integrated Circuits and Systems Design. SBCCI 2008. XXI Symposium on Integrated Circuits and Systems Design (SBCCI 2008) **Proceedings**..., New York: ACM, 2008a. v.1. pp.239–244.

DEPRÁ, D. A.; ROSA, V. dos S.; BAMPI, S. A novel hardware architecture dedicated to binary arithmetic decoder engines for the H.264/AVC CABAD. XVI IFIP/IEEE International Conference on Very Large Scale Integration. VLSI-SOC 2008. Rhodes Island. XVI IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SOC 2008) **Proceedings**..., 2008b. v.1. pp.519–522.

DEPRÁ, D. A.; ROSA, V. dos S.; BAMPI, S. A Method for HW Functional Verification through HW/SW Co-Simulation in Complex Systems: H.264/AVC Decoder as Case Study. 10th IEEE The Latin-American Test Workshop. LATW 2009. Armação dos Búzios, RJ. 2009 10th Latin American Test Workshop - LATW. 2009. pp. 10–16.

DEPRÁ, D. A.; ROSA, V. dos S.; BAMPI, S. Design and implementation of a high-performance architecture for binarization methods defined by H.264/AVC standard. XIV Workshop Iberchip. IBERCHIP 2008. Puebla. XIV Workshop Iberchip. 2008c. v.1. pp.1–4.

EECKHAUT, H.; CHRISTIAENS, M.; STROOBANDT, D.; NOLLET, V. "Optimizing the critical loop in the H.264/AVC CABAC decoder". In: **Field Programmable Technology**, 2006. FPT 2006. IEEE International Conference on. dez. 2006.

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

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

HA, S. H. V.; et all. Real-time MPEG-4 AVC/H.264 CABAC Entropy Coder. **IEEE Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on**. pp 225-226. Jan. 2005.

HANKERSON, D. R.; HARRIS, G. A.; PETER, J. D.. **Introduction To Information Theory And Data Compression**, 2nd Edition. Boca Raton, FL: Chapman & Hall/crc. 2003.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264 (03/05)**: advanced video coding for generic audiovisual services. [S.l.]. 2005.

INTERNATIONAL TELECOMMUNICATION UNION . **ITU-T Recommendation H.264:** Advanced video coding for generic audiovisual services. [S.l.]. 2004.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264 (05/03)**: advanced video coding for generic audiovisual services. [S.l.]: 2003.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.262 (11/94)**: generic coding of moving pictures and associated audio information - part 2: video. [S.l.]: 1994.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE Xplore:** Dynamic Home Page. Disponível em: < http://ieeexplore.ieee.org>. Acesso em: mar. 2008.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264 (05/03):** advanced video coding for generic audiovisual services. [S.l.], 2003.

INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Recommendation H.264/AVC (03/05):** advanced video coding for generic audiovisual services. [S.l.], 2005.

JOINT VIDEO TEAM (JVT). **ITU-T e ISO/IEC JTC1**, Advanced video coding for generic audiovisual services. (ITU-T Rec. H.264), mar. 2005.

KALVA, Hari. **The H.264 Video Coding Standard**. In: IEEE Computer Society Press. Vol. 13, pp. 86-90, No. 4. Los Alamitos, CA, USA. out. 2006.

KANNANGARA, C.; RICHARDSON, I. Computational Control of an H.264/AVC Encoder through Lagrangian Cost Function Estimation. In: INTERNATIONAL WORKSHOP ON VERY LOW BITRATE VIDEO, 2005. **Proceedings**… [S.l.:s.n.], 2005.

KARCZEWICZ, M.; KURCEREN, R. The SP- and SI-frames design for H.264/AVC. In: **IEEE Trans. Circuits Syst**. Video Techn. 13(7): 637-644, 2003.

KERNIGHAN, B.; RITCHIE, D. C: a Linguagem de Programação Padrão ANSI. Rio de Janeiro: Campus, 1999.

KIM, Chung-Hyo; PARK, In-Cheol. "High speed decoding of context-based adaptive binary arithmetic codes using most probable symbol prediction". In: **Circuits and Systems, 2006. ISCAS 2006**. **Proceedings**… 2006 IEEE International Symposium on. mai. 2006.

KUANG, Shiann Rong; JOU, Jer Min; CHEN, Ren Der; and SHIAU, Yeu Horng. et al. "Dynamic Pipeline Design of an Adaptive Binary Arithmetic Coder". In: **IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing**, Vol. 48, No. 9. set. 2001.

LEONARDO, Spectrum, **Mentor Graphics INC**. Disponível em: <http://www.mentor.com/products/fpga_pld/synthesis/leonardo_spectrum/>. Acesso em: out. 2008

MARPE, D.; SCHWARZ, H.; WIEGAND, T. "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard". In: **IEEE Transactions on Circuits and Systems for Video Technology**, Vol. 13, Nº 7, jul. 2003.

MARPE, D.; WIEGAND, T. A highly efficient multiplication-free binary arithmetic coder and its application in video coding. In: **Image Processing, 2003. ICIP 2003. Proceedings..., 2003 International Conference on**. v. 2, pp. 263-266. set. 2003.

MASSI, D. D.; **Ócio Criativo e Futuro do trabalho - A sociologia do trabalho**. Emissora TVE/Rede Brasil, Programa Roda Vida, 1998.

MEI-HUA, X.; YU-LAN, C.; FENG, R.; ZHANG-JIN, C. "Optimizing Design and FPGA Implementation for CABAC Decoder". In: **High Density packaging and Microsystem Integration, 2007. HDP '07. International Symposium on**. pp. 1-5. jun. 2007.

MENTOR, Graphics, INC. Disponível em: <http://www.mentor.com/company/>. Acesso em: out. 2008.

MIANO, J. **Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP**. New York: Assison-Wesley, 1999.

MODELSIM, **Mentor Graphics INC**. Disponível em: <http://www.altera.com/products/*software*/products/model/eda-ms.html>. Acesso em: out. 2008.

MOFFAT, A.; NEAL, R. M.; WITTEN, I. H. Arithmetic Coding Revisited. **ACM Transactions on Information Systems**, Vol. 16, No. 3, pp. 256–294. [s.l.] jul. 1998.

MRAK, M..; et all. Comparison of context-based adaptive binary arithmetic coders in video compression. **4th EURASIP Conference**. pp. 277-285. jul. 2003.

OSORIO, R. Roberto.; Bruguera, J. D. Arithmetic coding architecture for H.264/AVC CABAC compression system. **Digital System Design, 2004. Proceedings. Euromicro Symposium on**. pp 62-69. set. 2004.

OSTERMANN, J.; BORMANS, J.; LIST, P.; MARPE, D.; NARROSCHKE, M.; PEREIRA, F.; STOCKMMER, T.; WEDI, T. Video coding with H.264/AVC: Tools, Performance, and Complexity. In: **IEEE Circuits and Systems Magazine**, pp. 1540-7977/04, 1ºQ, 2004.

PENNEBAKER, W. B.; MITCHELL, J. L.; LANGDON, G. G. JR.; ARPS, R. B. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. **IBM 1. RES. DEVELOP**. VOL. 32 NO. 6. nov. 1988.

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

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

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

REISSLEIN, M. YUV Video Sequences. **Video Traces Research Group**. Disponível em: <http://trace.eas.asu.edu/yuv/index.html>. Acesso em: mar. 2008.

SAID, A. **Introduction to Arithmetic Coding - Theory and Practice**. Palo Alto: Imaging Systems Laboratory - HP Laboratories. 2004.

SALOMON, D. **Data Compression: The Complete Reference**. 2nd ed. New York: Springer, 2000.

SAYOOD, K. **Lossless Compression Handbook**. [s.l.] Academic Press. 2003.

SHANNON, C. E. A Mathematical Theory of Communication. **Bell Syst. Tech**. J. 27, 379. 1948.

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

SLATTERY, M. J.; MITCHELL, J. L. The Qx-coder. **IBM J. RES. DEVELOP**. VOL. 42 NO. 6. nov. 1998.

SULLIVAN, G. et al. The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. In: **CONFERENCE ON APPLICATIONS OF DIGITAL IMAGE PROCESSING, SPIE**, 2004. **Proceedings…,** Denver: SPIE, 2004.

SULLIVAN, G.; WIEGAND, T. Video Compression – From Concepts to the H.264/AVC Standard. **Proceedings of the IEEE**, [S.l.], v. 93, n. 1, p. 18-31, jan. 2005.

SUHRING, K. H.264/AVC Reference Software. In: **Fraunhofer Heinrich-Hertz-Institute**. Disponível em: <http://iphome.hhi.de/suehring/tml/download/>. Acesso em: mar. 2008.

SYNPLICITY INC. **Synplicity**: Home. Disponível em: <http://www.synplicity.com>. Acesso em: mai. 2008.

VQEG: **The Video Quality Experts Group Web Site**. Disponível em: <www.its.bldrdoc.gov/vqeg/>. Acesso em: nov. 2007.

WIEGAND, T. et al. Overview of the H.264/AVC Video Coding Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 13, n. 7, p. 560-576, jul. 2003.

WIKIPEDIA. CCD. Disponível em: < http://pt.wikipedia.org/wiki/Ccd>. Acesso em: mar. 2009.

WITTEN, I. H.; NEAL, R. M.; and CLEARY, J. G. Arithmetic coding for data compression. Commun. ACM, vol. 30, no. 6, pp. 520–540, jun. 1987.

XILINX. **Xilinx:** The Programmable Logic Company. Disponível em: <http://www.xilinx.com>. Acesso em: out. 2008.

YANG, Y.; LIN, C.; CHANG, H.; SU, C.; and GUO, J. A High Throughput VLSI Architecture Design for H.264 Context-Based Adaptive Binary Arithmetic Decoding With Look Ahead Parsing. In: **(ICME) Multimedia and Expo**, 2006 IEEE International Conference on, pp. 357-360, jul. 2006.

YI, Y.; PARK, I. High-Speed H.264/AVC CABAC Decoding. In: **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY**. VOL. 17, NO. 4, pp. 490-494. abr. 2007.

YU, W.; HE, Y. "A High Performance CABAC Decoding Architecture". In: **IEEE Transactions on Consumer Electronics**, Vol. 51, pp. 1352-1359, No. 4, nov. 2005.

ZHANG, P.; GAO, W.; XIE, D.; WU, D. "High-Performance CABAC Engine for H.264/AVC High Definition Real-Time Decoding". In: **Consumer Electronics, 2007. ICCE 2007. Digest of Technical Papers. International Conference on**. pp. 1-2. Las Vegas, NV, USA. jan. 2007.

Zheng, Y.; Zheng, S.; HUANG, Z.; ZHAO, Z. A TIME AND STORAGE OPTIMIZED HARDWARE DESIGN FOR CONTEXT-BASED ADAPTIVE BINARY ARITHMETIC DECODING IN H.264/AVC. In: **Multimedia and Expo, 2007 IEEE International Conference on**. pp. 1567-1570. jun. 2007.