UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MARCOS RODRIGUES CARVALHO

PROJETO DE DIPLOMAÇÃO

GERENCIAMENTO DE LISTAS DE IMAGENS DE REFERÊNCIA PARA DECODIFICAÇÃO DE VÍDEO NO PADRÃO H.264

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GERENCIAMENTO DE LISTAS DE IMAGENS DE REFERÊNCIA PARA DECODIFICAÇÃO DE VÍDEO PADRÃO H.264

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Altamiro Amadeu Susin
CO-ORIENTADORA: Dra. Leticia Vieira Guimarães

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MARCOS RODRIGUES CARVALHO

GERENCIAMENTO DE LISTAS DE IMAGENS DE REFERÊNCIA PARA DECODIFICAÇÃO DE VÍDEO PADRÃO H.264

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique de Grenoble, França

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique de Grenoble, França

Prof. Dr. Marcelo Götz, UFRGS

Doutor pela Universität Paderborn, Alemanha

Prof. Dr. Tiago Roberto Balen, Unilasalle

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Porto Alegre

2011

DEDICATÓRIA

Dedico este trabalho aos meus pais, em especial pela dedicação e confiança que tiveram para comigo em todos os momentos.

AGRADECIMENTOS

Aos meus pais que sempre me incentivaram.

Aos meus amigos pelo apoio e convívio.

Ao Professor Susin e toda equipe do LaPSI pela oportunidade de trabalharmos juntos.

Aos colegas de curso pelo seu auxílio nas tarefas desenvolvidas durante o curso.

À Universidade, a todos seus professores e funcionários, pelo ambiente de aprendizado e crescimento.

RESUMO

Este trabalho descreve o estudo e a implementação de um módulo componente de um decodificador de vídeo em *hardware*, com a função de gerenciar imagens previamente decodificadas utilizadas na predição da imagem corrente. O projeto também inclui o estudo e aperfeiçoamento de um programa em C usado na geração de resultados esperados para validação. O estudo parte do decodificador de vídeo no padrão H.264/AVC em desenvolvimento no Laboratório de Processamento de Sinais e Imagens – LaPSI, UFRGS, adicionando ao mesmo suporte a funções ainda não implementadas no que cabe à predição temporal entre imagens do vídeo.

Palavras-chaves: Engenharia Elétrica. Decodificador de Vídeo. Televisão Digital. Padrão H.264/AVC. VHDL.

ABSTRACT

This work describes the study and implementation of a module, as part of a hardware video decoder, that manages the already decoded pictures used in the current picture prediction. The project also includes the study and enhancement of a C language software used in reference results generation for validation. The study starts with the H.264/AVC standard video decoder that is been developed in the *Laboratório de Processamento de Sinais e Imagens* – LaPSI, UFRGS, adding support to yet not implemented functions regarding temporal prediction between video pictures.

Keywords: Electrical Engineering. Video Decoder. Digital Television. H.264/AVC Standard. VHDL.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABNT: Associação Brasileira de Normas Tecnicas

AVC: *Advanced Video Coding*

DPB: *Decoded Picture Buffer*

FIFO: *First In, First Out*

FPGA: *Field-programmable Gate Array*

IDR: *Instantaneous Decoding Refresh*

ITU: *International Telecommunication Union*

LaPSI: Laboratório de Processamento de Sinais e Imagens

MBAFF: *Macroblock-Adaptive Frame-Field*

MC: *Motion Compensation*

MMCO: *Memory Management Control Operation*

MPEG: *Moving Picture Experts Group*

NALU: *Network Abstraction Layer Unit*

POC: *Picture Order Count*

PRH.264: Programa de Referência em H.264

RAM: *Random Access Memory*

SBTVD: Sistema Brasileiro de Televisão Digital

UFRGS: Universidade Federal do Rio Grande do Sul

VHDL: *VHSIC Hardware Description Language*

VHSIC: *Very-High-Speed Integrated Circuits*

1. INTRODUÇÃO

Com o advento da televisão digital, tornou-se possível a transmissão de imagens de alta resolução e sem perda de qualidade devido a interferências do meio, porém se fez necessária a compactação das mesmas por limitações na banda do canal de transmissão. Várias formas existem de se fazer isso, assim, cada sistema de televisão digital assumiu o método de compressão mais eficiente disponível no momento. Para o Sistema Brasileiro de Televisão Digital foi escolhido o padrão MPEG-4 Part 10, também conhecido como H.264, que apresenta um ganho de compactação de até 2 vezes em comparação com o anterior, o MPEG-2, utilizado no padrão japonês [1].

O padrão H.264/AVC é hoje o mais avançado em codificação de vídeo. Ele suporta a compensação de movimento com referências a múltiplas imagens, o que aumenta sua eficiência [2], mas também sua complexidade. Ao mesmo tempo, requer uma complexa gerência das listas de referências, assim como do armazenamento das imagens previamente decodificadas que são referenciadas.

A norma brasileira [3] prevê a codificação de vídeo, tanto progressivo quanto entrelaçado, no padrão H.264, e uso do perfil *baseline* para transmissão a dispositivos móveis, com menos complexidade, chamada de *one-seg*, assim como do perfil *high*, com alto rendimento, para a transmissão em alta definição, chamada também de *full-seg*, que faz uso da compensação de movimento com múltiplas referências. O mesmo exige que o decodificador seja capaz de trabalhar com até 16 imagens armazenadas na memória, que podem ser usadas como referência. Além disso, as mesmas devem ser ordenadas em pelo menos 2 listas que devem ser inicializadas e reordenadas para cada novo quadro decodificado.

Em um algoritmo típico de decodificação como descrito na norma ITU [4], essas operações são realizadas em *software*, por um processador, através de rotinas de ordenamento de listas, porém em um decodificador modular em *hardware*, a implementação otimizada

passa a ser mais complexa [5]. Isso se deve a uma variedade de critérios de ordenamento, que dependem do tipo da imagem sendo decodificada, variando ao longo de uma mesma seqüência de vídeo.

Neste trabalho serão apresentados os aspectos da decodificação de vídeo no que diz respeito à formação de listas de imagens previamente decodificadas que são usadas na predição das novas imagens. No segundo capítulo será apresentado o contexto do projeto, juntamente com a motivação do mesmo. A seguir serão introduzidos conceitos fundamentais de codificação e decodificação no padrão H.264/AVC, dentro do tema de predição temporal. No terceiro capítulo é apresentado o conceito básico do decodificador e seus componentes relacionados às listas de referências. O algoritmo de gerenciamento dessas listas de imagens é detalhado no capítulo quarto, e no seguinte é demonstrada a implementação proposta. Finalmente no último capítulo, as considerações finais.

2. CONTEXTO DO PROJETO

Tendo em vista o domínio dos processos de codificação e decodificação de vídeo para televisão digital, com tecnologia nacional, através de vários projetos de pesquisa em várias universidades do país, incluindo a UFRGS, foi criada a Rede H.264 SBTVD [6], com suporte do Governo Federal. Seu objetivo é o desenvolvimento de um codificador e um decodificador no sistema brasileiro, para áudio e vídeo. Especificamente no LaPSI se dá a pesquisa acerca do decodificador de vídeo em *hardware* e do sistema em *chip* para um receptor de TV digital.

Foi adotada uma estratégia de criação modular e incremental, por se tratar de um sistema de grande complexidade. Diversos desenvolvedores trabalham nos módulos necessários, escritos em VHDL, que são integrados em uma segunda etapa. Além disso, a interface entre módulos precisa ser especificada previamente para evitar incompatibilidades durante a integração. Por isso, é desenvolvido no laboratório um *software* para geração de resultados esperados, o Programa de Referência H.264 [7], o PRH.264, mostrado na figura 1.

Atualmente o programa suporta vídeos no perfil básico do padrão, permitindo a decodificação da transmissão em baixa resolução das emissoras, o *one-seg*, voltado para dispositivos móveis. Ao longo do desenvolvimento do projeto, esse suporte foi expandido, tornando o software capaz de decodificar vídeos com entrelaçamento, e assim, aproximando-o do objetivo final de suportar a transmissão em alta definição da Televisão Digital Brasileira.

Também com objetivo de suporte ao projeto, fui desenvolvendo uma interface gráfica baseada no código do PRH.264 já existente, com a finalidade de obter dados relevantes à codificação dos fluxos de vídeo, dando origem a um programa derivado, o Programa de Análise H.264, visto na figura 2. Ele permite a seleção dos parâmetros a serem analisados, filtrando o conteúdo de interesse, mostrando-os em tempo real, sendo capaz de acompanhar a chegada dos dados de transmissão de TV local com ajuda de um sintonizador e um demultiplexador disponíveis no Laboratório.

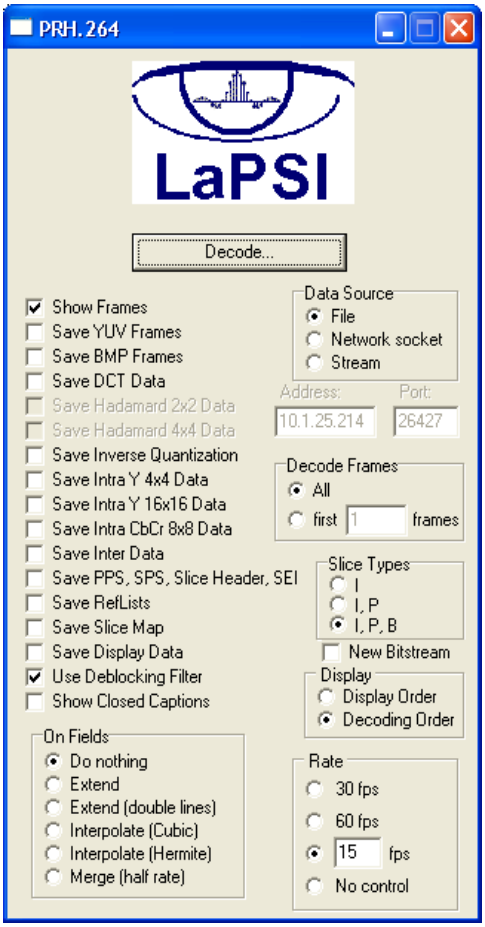


Figura 1 – PRH.264 - Programa de referência desenvolvido no LaPSI

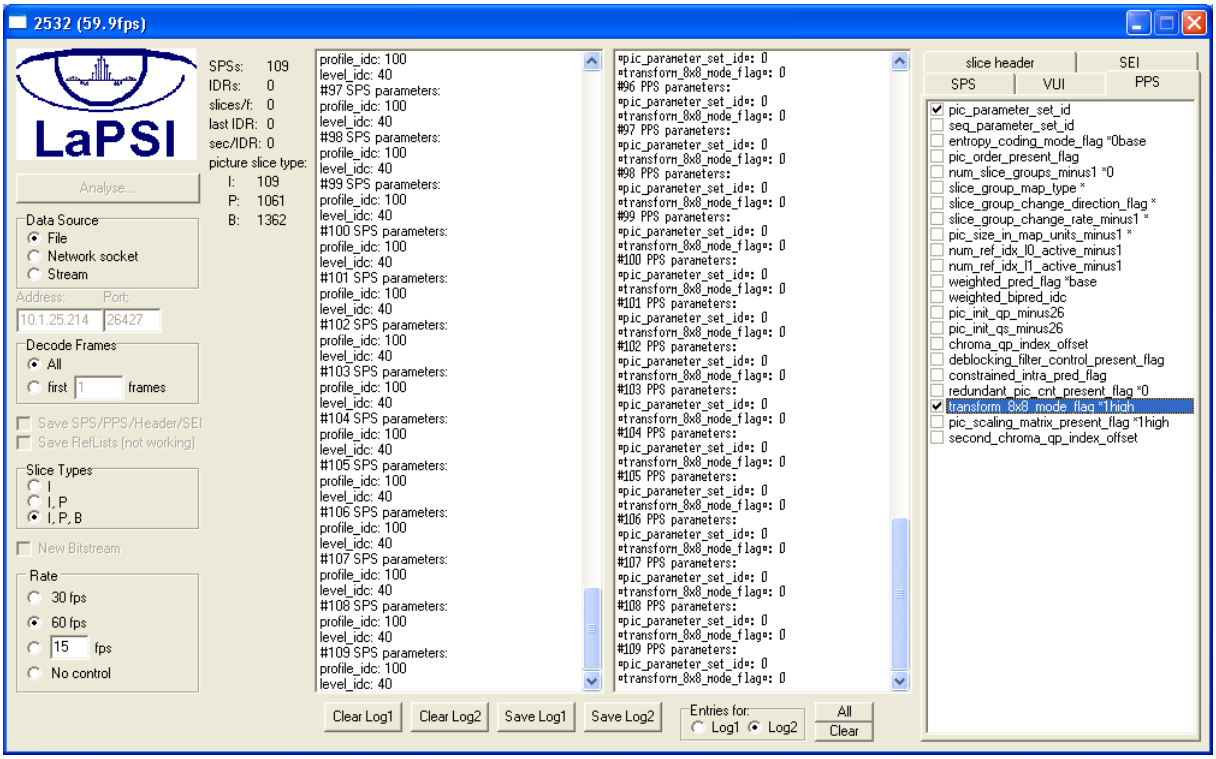


Figura 2 – PAH.264 - Programa de análise derivado do PRH.264

3. PADRÃO H.264

O padrão de vídeo adotado no SBTVD, o H.264/AVC, foi desenvolvido em conjunto por MPEG (*Moving Picture Experts Group*) e ITU (*International Telecommunication Union*), e é descrito pela norma [4]. No caso da sua utilização no Sistema Brasileiro de Televisão Digital, a norma nacional [3] restringe alguns pontos como a resolução do vídeo a ser usado, o tipo de entrelaçamento, o modo de divisão da imagem entre outros.

A compactação do vídeo se baseia em aproveitar as redundâncias espaciais e temporais das imagens, ou seja, representar áreas de imagem semelhantes apenas uma vez, referenciando-a nas demais. Isso pode ser feito em uma mesma imagem, identificando as repetições de padrões no espaço, ou seja, regiões que aparecem num mesmo instante de tempo, ou ainda ao longo da seqüência de vídeo, que costuma ter uma grande correlação entre imagens de instantes adjacentes.

Figura 3 - Redundância Espacial e Temporal [8]

Além disso, toda informação referente a parâmetros de codificação, assim como dados de vídeo mesmo, é comprimida com codificação de entropia e serializada em um único fluxo de dados, o *bitstream*. A codificação de entropia baseia-se na utilização de símbolos mais curtos para dados que aparecem mais freqüentemente.

3.1 Estrutura da Imagem

Para ser codificada, a imagem é subdividida em *slices*, áreas arbitrárias da imagem onde com a principal finalidade de contenção de erros, e esses são divididos em macroblocos, como mostra a figura 4. Cada macrobloco possui 16x16 *pixels*, e eles ficam regularmente distribuídos na imagem, sendo uma unidade fundamental da mesma para o processo de codificação. Uma imagem é formada sempre por um número inteiro de macroblocos, completando-se linhas ou colunas de *pixels* se necessário. Portanto a imagem trabalhada sempre tem dimensões em *pixels* de multiplos de 16.

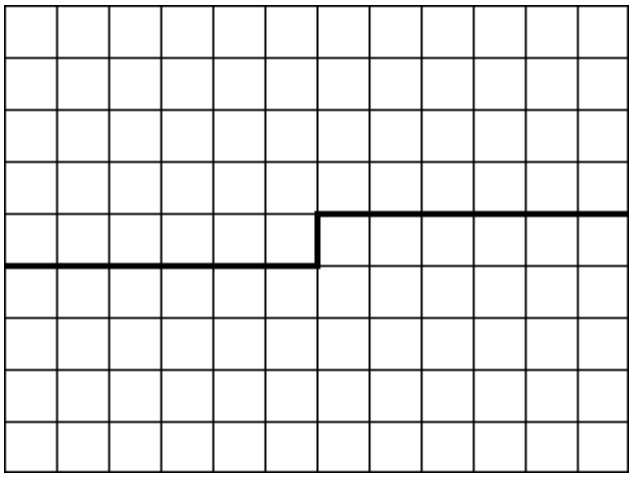


Figura 4 - Imagem dividida em 2 *slices* e seus macroblocos

Ao ser codificado, o macrobloco pode ser subdividido em partições de 16x8, 8x16 ou 8x8 *pixels*, que ainda podem ser divididas em subpartições de 4x4, 4x8 ou 8x4 *pixels*. As subdivisões dos macroblocos são restritas ao modo de predição, ou seja, espacial ou temporal, limitando à espacial subpartições 4x4 ou o próprio macrobloco de 16x16.

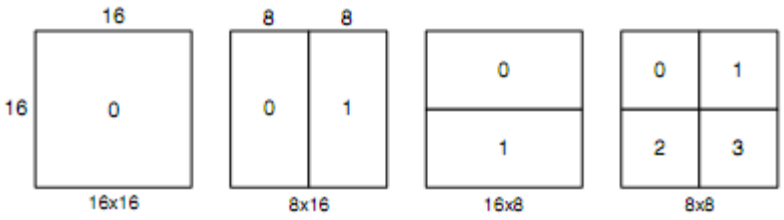


Figura 5 - Exemplo de partições do macrobloco [8]

Uma característica importante de construção da imagem legada das transmissões analogias é o conceito de entrelaçamento de vídeo. Em um vídeo entrelaçado, os quadros são formados por campos, par e impar (*top field* e *bottom field* respectivamente), que contêm metade das linhas do quadro, ou seja, metade da informação visual. A imagem é formada pela alternância de campos amostrados em instantes de tempo diferentes, o que na época da televisão de tubo de raios catódicos combinava-se com o efeito de persistência visual para dar mais fluidez ao movimento sem aumentar a taxa de quadros por segundo.

No caso do vídeo digital, isso significa metade da resolução vertical do quadro, mas não somente isso. Há grande correlação entre campos complementares, ao mesmo tempo que a sua alternância de linhas em posições pares e impares, em termos de coordenadas verticais, torna o entrelaçamento um caso a parte para codificação de vídeo.

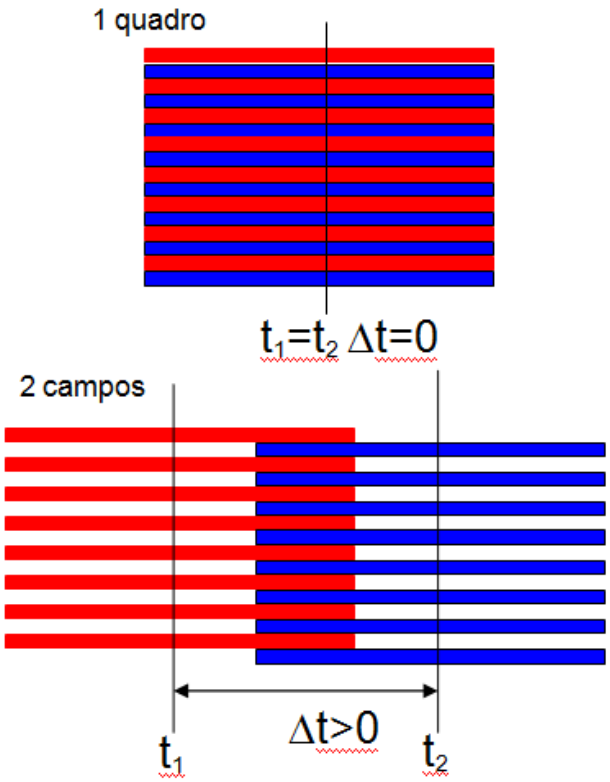


Figura 6 - Diferença da amostragem entre vídeo progressivo e entrelaçado

Para contemplar os materiais de vídeo entrelaçado com maior eficiência, há um modo de codificação que considera os macroblocos em pares verticais, como mostrado na figura adiante, que são individualmente codificados, ora como progressivos ora como entrelaçados, dependendo da correlação espacial das linhas de campos complementares. Esse método recebe o nome de MBAFF (*Macroblock-Adaptive Frame-Field*). Com amostragem em tempos distintos, o modo quadro é mais eficiente quando não há movimento entre os campos, e o outro modo no caso de haver.

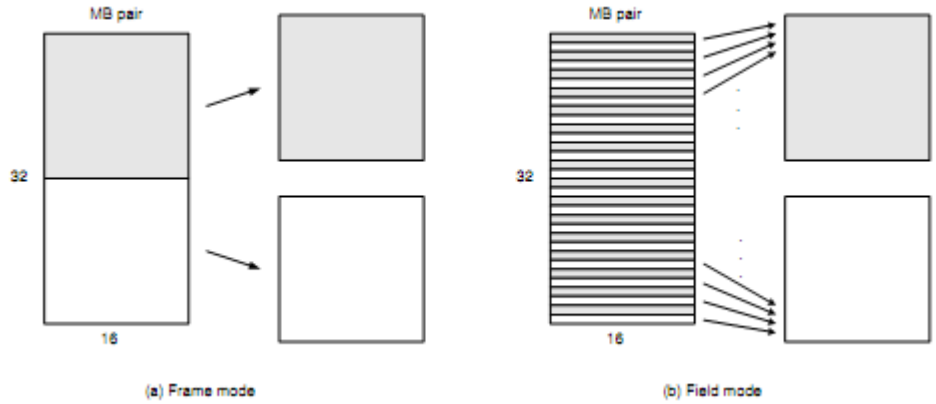


Figura 7 - Pares de macroblocos em um slice com MBAFF [8]

3.2 Tipos de Predição

A predição espacial se baseia na vizinhança do bloco, mostrada na figura 8, já decodificada, que consiste na linha superior e a lateral esquerda, pois a imagem é codificada e decodificada na ordem de leitura ocidental, um macrobloco por vez. Ela é feita aplicando padrões de cópia dos macroblocos vizinhos, verticalmente, horizontalmente ou em diagonais diversas, fazendo médias dos valores das bordas da vizinhança. São chamados de macroblocos do tipo I, pois a predição espacial também é chamada de intra-quadro, e integram um *slice* do tipo I.

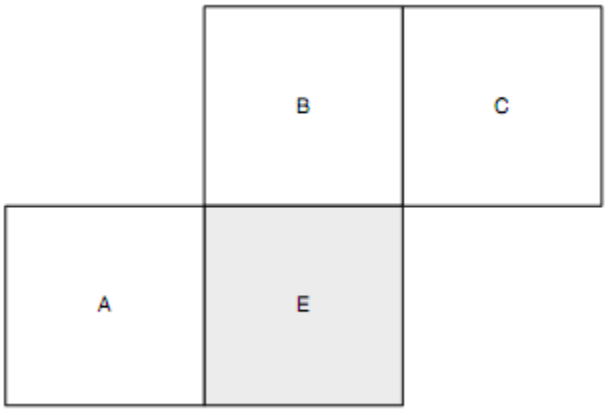


Figura 8 - Exemplo de vizinhança do macrobloco E [8]

Já a predição temporal, aproveita-se da grande semelhança entre imagens consecutivas em uma tomada única de vídeo, localizando áreas da imagem que se moveram ao longo do tempo. Esse processo, chamado de compensação de movimento, ou *Motion Compensation* (MC), começa com a mesma posição do macrobloco, só que em outra imagem, buscando uma que melhor a aproxime. Isso pode se dar em uma imagem a ser exibida antes ou depois da atual, não estando limitada ao passado na linha de tempo do vídeo. O deslocamento resultante compõe os vetores de movimento da predição, que são representados por setas na figura 10. Os slices com macroblocos que utilizam apenas uma referência são chamados de *slices* tipo P.

Figura 9 - Diferença entre quadros consecutivos de um vídeo [8]

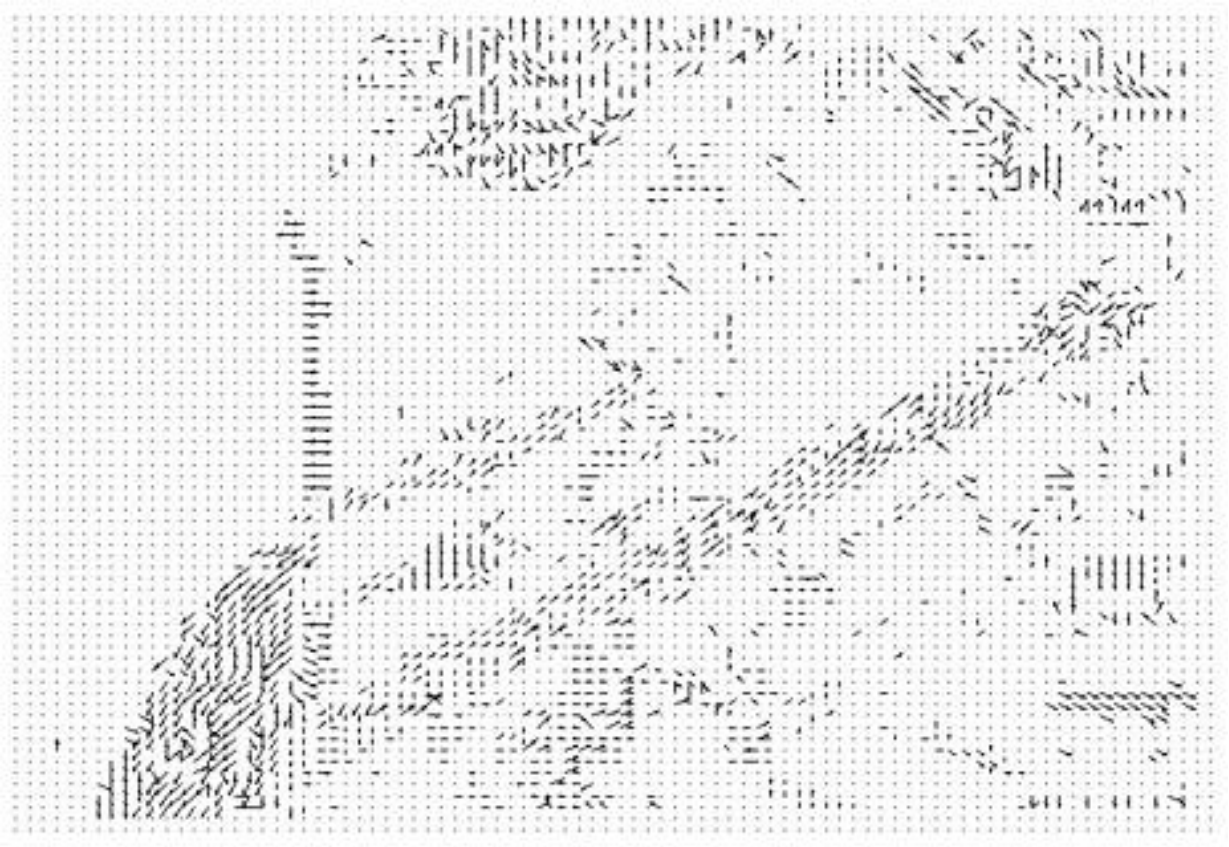


Figura 10 - Vetores de Movimento usados na predição temporal da figura 7 [8]

Como característica inovadora do padrão H.264 [8], as imagens podem ser formadas usando predição de duas diferentes referências, a chamada bi-predição, chamada de predição de *slices* do tipo B, cujos casos são mostrados na figura 11.

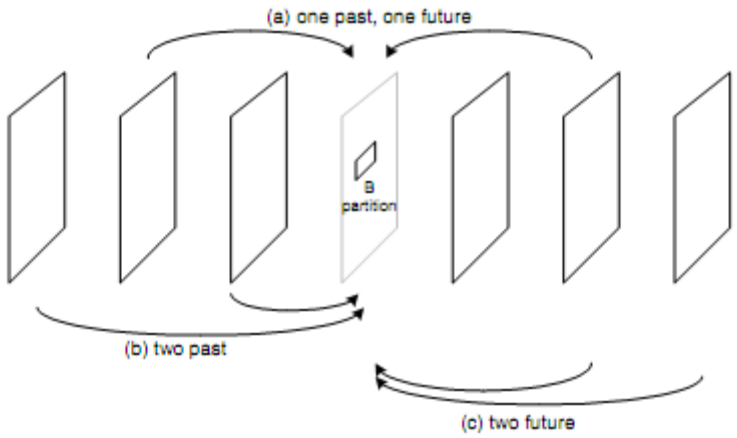
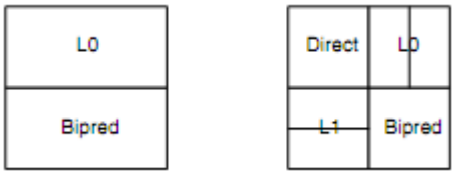


Figura 11 - Imagens de referência usados na predição de um *slice* tipo B [8]

O uso de imagens futuras, do pondo te vista da exibição significa que as imagens são decodificadas em uma ordem diferente da de apresentação, que seria a sua ordem natural. Isso possibilita a predição de uma imagem a partir das duas adjacentes, com um efeito equivalente a uma “interpolação” de vídeo. Macroblocos do tipo B podem ser subdivididos, tendo várias combinações de predições, algumas mostradas na figura 12.



Partition	Options
16 × 16	Direct, list 0, list1 or bi-predictive
16 × 8 or 8 × 16	List 0, list 1 or bi-predictive (chosen separately for each partition)
8 × 8	Direct, list 0, list 1 or bi-predictive (chosen separately for each partition).

Figura 12 - Exemplos de modos de predição usados em macroblocos tipo B [8]

Devida a essa interdependência de imagens, teoricamente não seria possível iniciar a decodificação do vídeo em qualquer ponto, como é o caso de uma transmissão ininterrupta como a difusão televisiva. Para permitir isso, existem *slices* do tipo IDR (*Instantaneous Decoding Refresh*), que sinalizam um ponto na linha de tempo da decodificação antes do qual nenhuma imagem é referenciada, ou seja, a partir da chegada de um IDR é seguro iniciar uma decodificação com predição temporal sem haver falta de imagens de referência.

4 DECODIFICADOR

Após o processo de sintonia e demodulação do sinal de TV, referente ao meio físico, e depois do fluxo (*stream*) fundamental com a informação do áudio, vídeo, legendas e outros dados ser recuperado, ele passa por um demultiplexador, restando para o decodificador H.264 apenas a informação referente ai vídeo, em uma seqüência serial de bits, o *bitstream*. O diagrama de blocos básico do decodificador pode ser visto na figura 14.

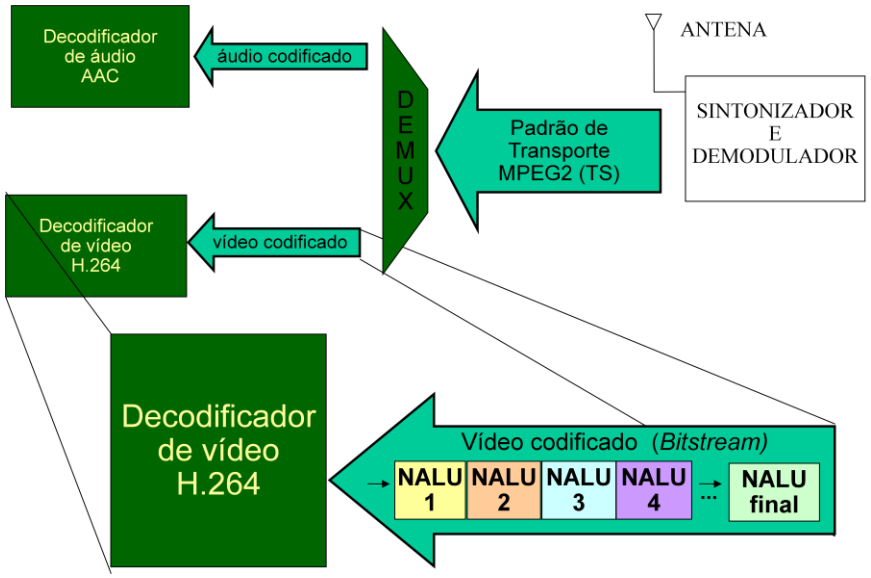


Figura 13 - Esquema da recepção de TV digital

Conforme o diagrama a seguir, pode-se ver que há chaveamento entre predição intra-quadro, ou espacial, e inter-quadro, a temporal, assim como uma realimentação dos quadros decodificados como quadros de referência. Observa-s uma quantização (Q^{-1}) e uma transformada (T^{-1}) inversas com a finalidade de compactar os resíduos, que são os detalhes adicionados à imagem predita.

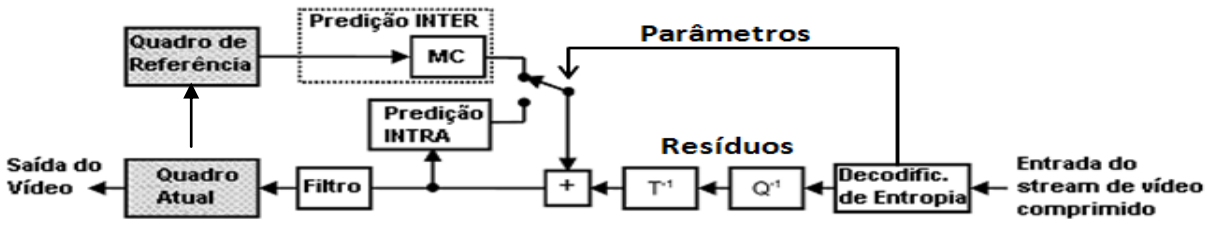


Figura 14 - Diagrama de Blocos Básico do Decodificador H.264 [8]

4.1 *Parser*

A decodificação do *bitstream* de vídeo inicia no módulo *parser* [9], com a descompressão das informações de entrada, usando decodificação de entropia, gerando uma série de parâmetros que são usados pelos diversos módulos para reconstruir cada imagem. Essas informações configuram o decodificador para a correta interpretação dos dados de vídeo a serem decodificados, tendo impacto direto em várias etapas do processo.

Primeiramente, o *parser* identifica as Unidades de Abstração de Rede, ou *Network Abstraction Layer Units* (NAL *units*), quanto ao seu tipo, que podem ser de parâmetros, *slices* de imagens ou delimitadores, que salientam a separação das imagens.

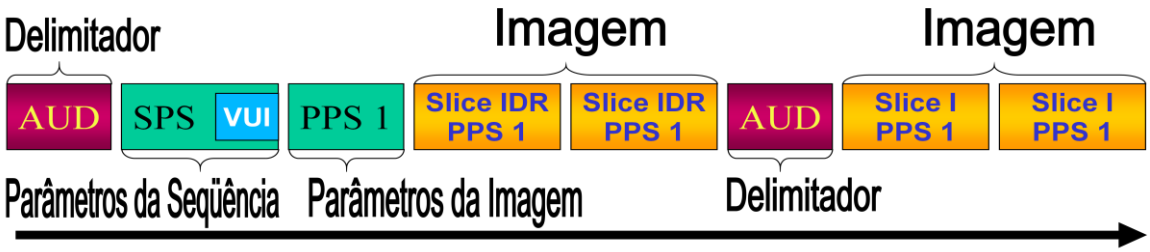


Figura 15 - Seqüência de unidades NAL do bitstream

Os parâmetros de seqüência, do SPS (*Sequence Parameter Set*) contêm parâmetros relevantes a toda seqüência de vídeo, como a resolução dos quadros. Os parâmetros de imagem, do PPS (*Picture Parameter Set*) são mais específicos, com informações relativas a predição, podendo ser atribuídos a uma ou mais imagens da seqüência. As unidades de NAL de *slices* iniciam com um cabeçalho trazendo mais parâmetros, o *slice header*, que define entre outras coisas o tipo de *slice* e quais macroblocos da imagem ele contém, assim como seu *frame_num*, o número que ordena os quadros para exibição. O cabeçalho de cada slice indica qual conjunto de parâmetros de imagem ele vai utilizar, que por sua vez tem a indicação de um conjunto de parâmetros de seqüência ao qual ele pertence.

4.2 *Motion Compensation*

Como visto, a reconstrução das imagens é feita através de predição baseada em áreas próximas dentro da própria imagem, a predição espacial, ou intra-quadro, assim como de imagens previamente reconstruídas, a predição temporal, ou predição inter-quadro, minimizando a informação necessária a ser transmitida. A predição temporal utiliza um módulo de compensação de movimento, MC (*Motion Compensation*), que faz o mapeamento de posições das áreas semelhantes entre diferentes quadros, usados na predição.

A compensação do movimento é feita através de vetores de movimento, que são coordenadas relativas da posição da imagem que está sendo referenciada, sendo assim possível montar imagens em cenas de movimento, em especial quando o movimento é paralelo ao plano da imagem, ou seja, sem deformação dos objetos na imagem. Os vetores de movimento são inferidos pelos vetores das vizinhanças, assim como os vetores das imagens temporalmente próximas, supondo alta correlação também nos movimentos entre as mesmas.

O MC, além dos vetores, precisa de uma indicação de qual a imagem foi utilizada como referencia na codificação, para tanto ele utiliza índices como entrada de uma ou duas listas, dependendo do tipo de *slice*, P ou B respectivamente. Esses índices são passados para cada macrobloco, ou ainda partição de macrobloco, e as listas são criadas no início da decodificação da imagem, e são dependentes de vários parâmetros da mesma.

O que não pôde ser predito, o chamado resíduo, é descomprimido e adicionado ao resultado das predições. Essa descompressão é feita pelo decodificador de entropia. O resíduo é tão menor quanto melhor for a predição feita, o que afeta diretamente a eficiência da codificação.

4.3 *Decoded Picture Buffer*

Finalmente, a imagem gerada é filtrada, para remover os efeitos de bordas dos blocos nos quais ela é dividida para codificação, e então é armazenada e exibida. As imagens armazenadas são usadas como referência para predição das próximas a serem decodificadas, não necessariamente na mesma ordem em que são exibidas.

O DPB (*Decoded Picture Buffer*) é o banco que armazena as imagens que já foram decodificas dentro de uma janela deslizante, ou seja, as últimas N imagens, incluindo as informações dos vetores de movimento usadas na predição temporal e estrutura de codificação da mesma, campo ou quadro. O número de imagens do DPB depende da resolução, pois ele é limitado a 12.288kB para a transmissão *full-seg* e 891kB para *one-seg*.

As imagens nele contidas podem ser usadas ou não na construção das listas de referência dependendo da marcação que elas recebem através de seus parâmetros. Essas marcações também podem ser alteradas por instruções específicas recebidas juntamente com o cabeçalho de *slices*.

Valores associados a cada imagem para posterior ordenamento nas listas são também armazenados no DPB, como o *frame_num* dos cabeçalhos dos seus slices e o Picture Order Count, que deve ser calculado para cada imagem.

5 GERENCIAMENTO DE LISTAS

Assim que os as informações de uma imagem são disponibilizadas pelo *parser*, é calculado o seu POC (*Picture Order Count*), e para cada imagem com predição temporal é necessária uma nova inicialização das listas de imagens de referência, lista 0 e lista 1, seguindo o fluxograma mostrado na figura 16. Depois de feita a decodificação, a imagem resultante é armazenada no DPB e marcada como "usada para referência" ou "não usada para referência", dependendo dos parâmetros passados pelo *parser*. São também mantidos os dados dos vetores de movimento de cada macrobloco para predição dos vetores das próximas imagens.

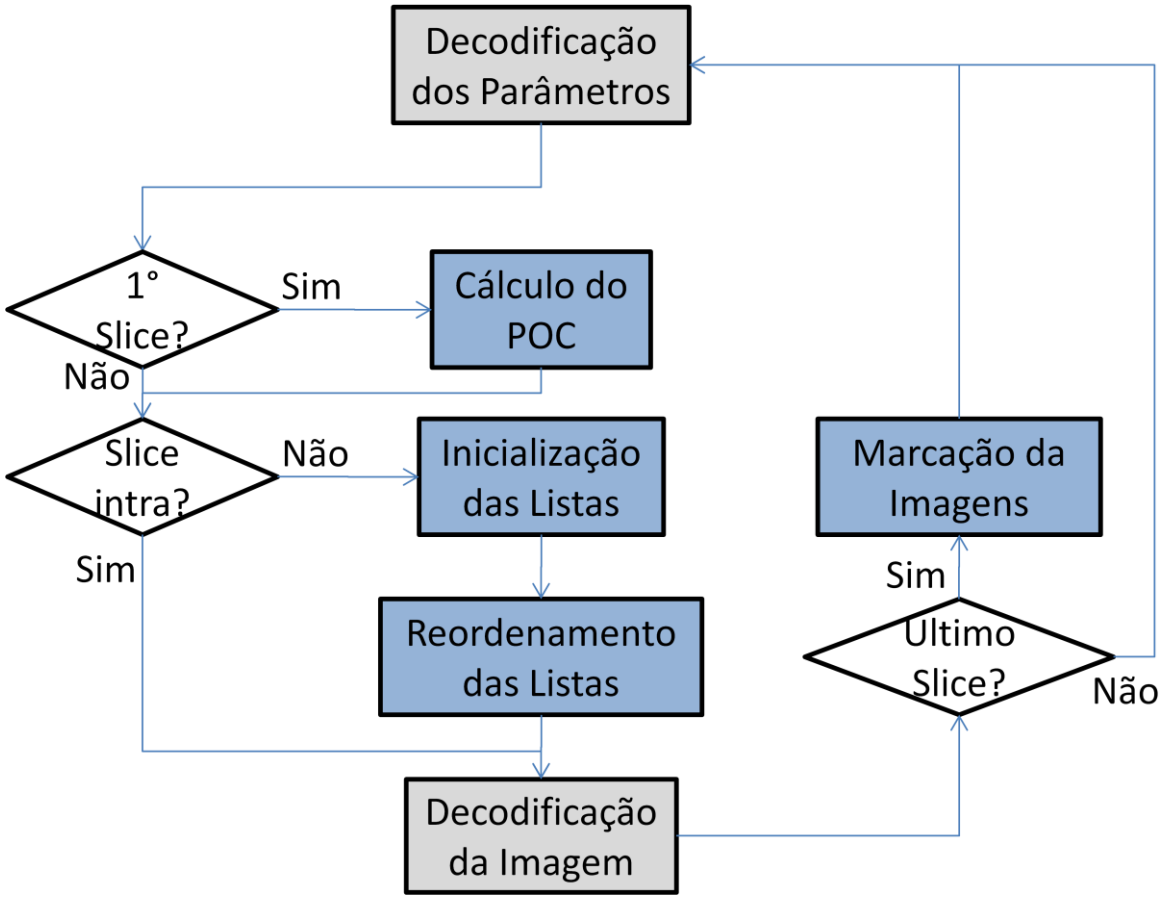


Figura 16 – Fluxograma do uso das listas durante a decodificação

A inicialização das listas, por sua vez, começa com a atribuição dos valores de FrameNum e PicNum, para cada quadro ou campo, respectivamente, marcado como referência no DPB, listando-os segundo algum critério para em seguida atribuir-lhes índices, ora para cada campo individual, ora para cada par completo. Essa atribuição depende de como a imagem que está sendo decodificada é estruturada quanto ao entrelaçamento. As imagens são assim organizadas em 1 ou 2 listas, dependendo do tipo da decodificação atual, P ou B, respectivamente.

As derivações de FrameNum e PicNum são simples, sendo o FrameNum das imagens obtido por uma relação entre o *frame_num* da imagem atual e da armazenada, e o PicNum, que é diferente para cada campo de um par entrelaçado, tendo seu valor igual ao dobro ou o dobro mais 1 do FrameNum, dependendo da paridade do campo.

Após a inicialização das listas, é possível haver um reordenamento customizado, através de parâmetros enviados pelo codificador junto com o cabeçalho do slice. Essas mudanças na ordem das imagens nas listas têm o objetivo de diminuir o tamanho dos índices utilizados nas referências, privilegiando imagens mais antigas, do ponto de vista da decodificação, e que não seriam naturalmente colocadas em índices próximos da imagem atual, mas assim mesmo são mais usadas pelos seus macroblocos.

As imagens podem ser marcadas como referências de longo prazo, ou *long term reference*, quando sua utilização é feita repetidamente em trechos relativamente distantes na linha do tempo da seqüência de vídeo. Desse modo, não se exclui a imagem do DPB mesmo após ela estar fora da janela deslizante de N imagens armazenadas, como ocorreria com uma de curto prazo, ou *short term reference*. Ainda podem ser feitas alterações nas marcações das imagens através das Operações de Gerenciamento de Memória.

5.1 Cálculo do POC

O POC (*Picture Ordem Count*) é um contador de imagens utilizado pelo decodificador como critério de ordenamento das listas de referência para imagens com predição bidirecional, os *slices* B. Seu valor não é explicitamente transmitido com o vídeo, e deve ser calculado para cada imagem.

Esse cálculo pode ser feito de 3 maneiras diferentes, dependendo de como é especificado nos parâmetros da seqüência de vídeo, através do parâmetro de seqüência *pic_order_cnt_type*. Dependendo de seu valor, são necessárias diferentes entradas para o cálculo, de acordo com a tabela 1, a título de exemplificação.

POC tipo 0	**POC tipo 1**	**POC tipo 2**
field_pic_flag	field_pic_flag	field_pic_flag
bottom_field_flag	bottom_field_flag	bottom_field_flag
prevPicOrderCntMsb	frame_num	frame_num
prevPicOrderCntLsb	num_ref_frames_in_pic_order_cnt_cycle	num_ref_frames_in_pic_order_cnt_cycle
pic_order_cnt_lsb	prevFrameNumOffset	prevFrameNumOffset
delta_pic_order_cnt_bottom	delta_pic_order_cnt[0]	MaxFrameNum
MaxPicOrderCntLsb	delta_pic_order_cnt[1]	prevFrameNum
	offset_for_ref_frame	
	offset_for_non_ref_pic	
	offset_for_top_to_bottom_field	

Tabela 1 – Parâmetros utilizados no calculo do POC

A saída do processo é um valor de POC para cada campo, *top* POC para o campo par e *bottom* POC para o ímpar, mesmo quando a imagem é um quadro, pois cada campo pertencente ao quadro pode ser referenciado separadamente no caso da imagem a ser decodificada ser um campo. Os detalhes do cálculo podem ser encontrados na norma ITU[4].

5.2 Inicialização das Listas

Cada vez que a predição temporal for utilizada, ou seja, em imagens tipo P ou B, a lista deve ser inicializada. Nesse ponto, as imagens previamente decodificadas que estão marcadas como “usadas para referência” são classificadas segundo um critério, dependendo do tipo de entrelaçamento da imagem a ser decodificada, formando uma ou duas listas, cada uma com ordenação diferente, conforme a norma [4].

Quando a imagem que vai ser reconstruída é do tipo P e um quadro, o ordenamento das listas é baseado no PicNum das imagens de curto prazo, que deriva do FrameNum, em ordem decrescente, e cada par de campos completo é referenciado de uma vez só, como se fosse um único quadro. Campos sem par não são listados. Após todas de curto prazo, as imagens de longo prazo recebem os índices conforme seus valores de LongTermPicNum.

Quando a imagem que será decodificada é um campo do tipo P, o ordenamento é feito pelo FrameNum das imagens de curto prazo, diretamente, mas agora incluindo todo campo que estiver marcado no DPB como usado para referência, pareado ou não. Os quadros são decompostos em um par de campos, e cada um ganha um índice diferente na lista, sendo sempre o primeiro o de paridade oposta da imagem atual, alternando a cada índice da lista. Por exemplo, se a imagem atual é um campo ímpar, ou *bottom field*, o primeiro índice da lista será um campo par, ou *top field*, o seguinte um ímpar, *bottom*, depois par novamente e assim por diante

As imagens do tipo B tem suas listas ordenadas através do *Picture Order Count*, sendo a diferença entre a lista 0 e a lista 1 que a primeira tem seus índices distribuídos primeiramente às imagens de curto prazo com POC menor que da imagem atual, descendentemente, e somente após essas as de POC maior que a atual, em ordem crescente. A lista 1 tem a formação com essas etapas na ordem inversa, e as imagens de longo prazo seguem em ambos os casos.

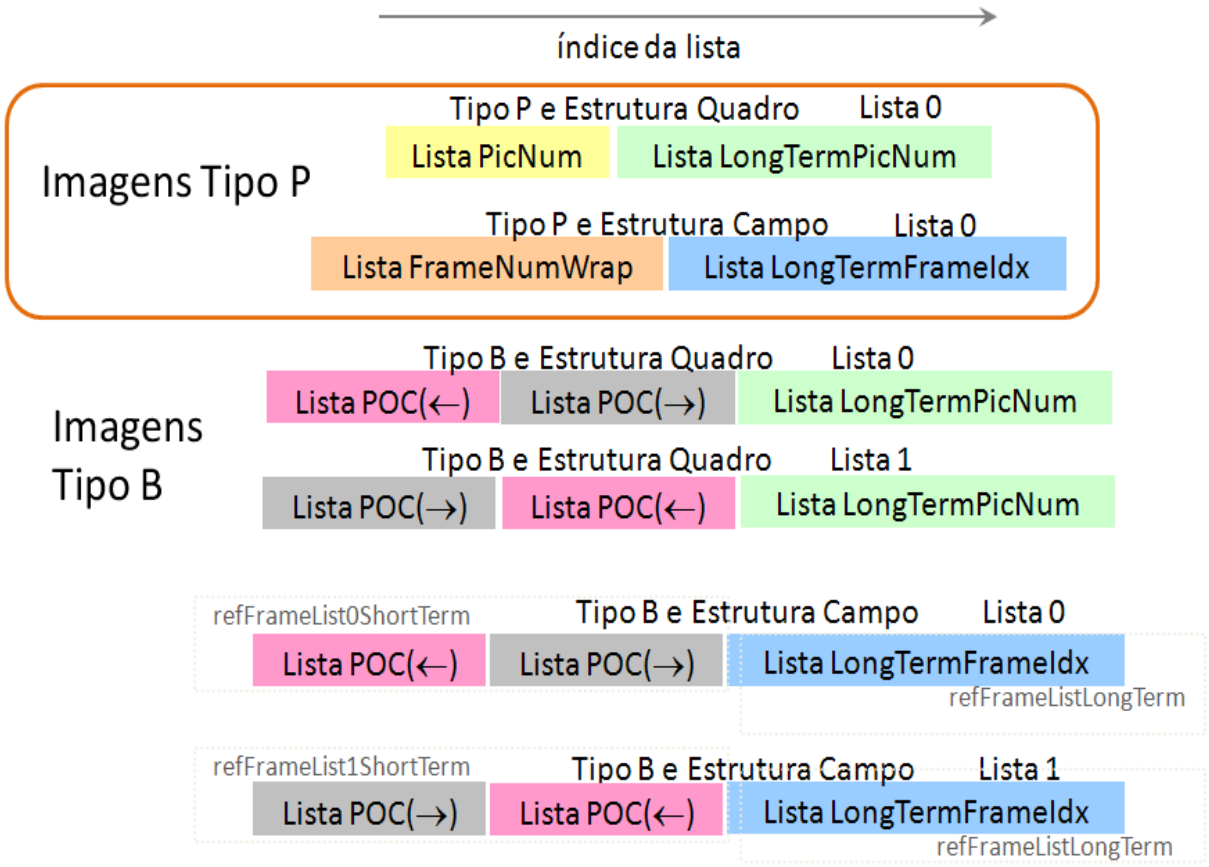


Figura 17 - Formação das listas conforme o tipo de slice e a estrutura de entrelaçamento

5.3 Reordenamento

O reordenamento é o processo onde uma imagem tem sua posição na lista 0 ou na lista 1 alterada conforme for indicado pelo codificador através de parâmetros transmitidos explicitamente. O reordenamento de uma listax acontece ao menos uma vez quando a *flag* contida no cabeçalho da imagem a ser decodificada *ref_pic_list_reordering_flag_lx* é verdadeira para a lista x.

Este reordenamento pode ser aplicado tanto às listas de curto prazo quando as de longo prazo, o que é indicado pelo parâmetro *reordering_of_pic_nums_idc*, que acompanha o *flag*, sendo 0 ou 1 para curto prazo e 2 para longo prazo, e ele pode ser executado ciclicamente para várias imagens da lista. O processo é interrompido pela sinalização do valor *reordering_of_pic_nums_idc*=3.

O processo de reordenar consiste em procurar na lista a imagem a ser reposicionada (indicada por PicNum em referência de curto prazo por LongTermPicNum em referência de longo prazo), e reposicionar a imagem na lista através de um índice definido.

Para o reordenamento de curto prazo (*short-term*) é passado o parâmetro *abs_diff_pic_num_minus1*, e para longo prazo, é passado o elemento sintático *long_term_pic_num*, utilizados para a obtenção de PicNum e LongTermPicNum a ser buscado na lista através de um processo descrito na norma [4].

As operações de reordenamento são aplicadas às listas após a sua inicialização, antes da sua utilização e só são válidas durante a decodificação da imagem corrente. Abaixo, na figura 18, pode-se ver um exemplo de reordenamento, onde a imagem indicada pelo PicNum é colocada no índice, exprexo por refIdx, que então é incrementado para a colocação da próxima imagem com o próximo PicNum indicado.

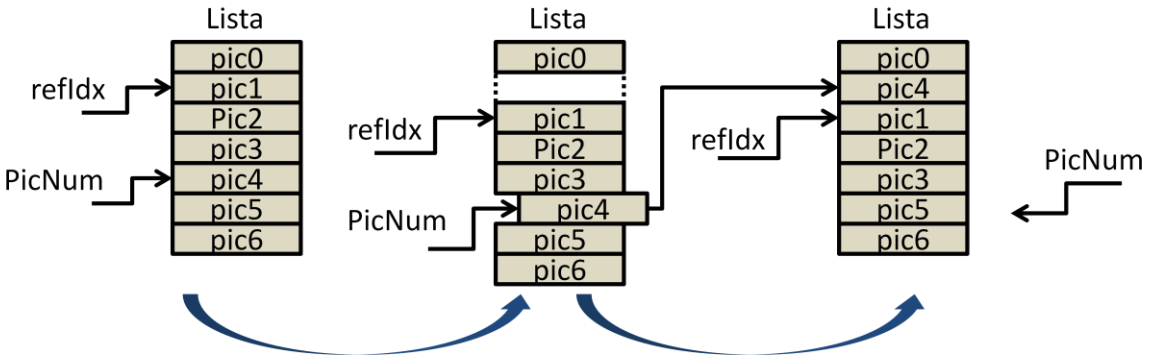


Figura 18 - Processo de reordenamento

5.4 Marcação das Imagens e o DPB

No decodificador em hardware em questão no projeto, as informações visuais das imagens são armazenadas em uma RAM externa, devido ao tamanho, e são acessadas através de endereços que referem ao início de cada uma. Já as informações de construção da imagem podem ser armazenadas localmente em estruturas de registradores associadas a cada endereço de imagem na memória.

No momento da marcação, a imagem atual, após totalmente decodificada, é definida como usada ou não na construção das próximas listas, e se vai ser como de curto ou longo prazo. Também podem ser aplicadas às imagens já existentes na memória operações de controle de gerenciamento de memória, ou *Memory Management Control Operations* (MMCO), que alteram permanentemente os atributos de ordenamento das imagens no DPB, como a sua marcação de "usada para referência" ou o seu índice de referência de longo prazo.

MMCO	Operação	Argumento
0	Finaliza o processo de Controle Adaptativo de Memória (*Adaptive Memory Control*).	-
1	Marca uma imagem de referência de curto prazo como "não usada para referência".	difference_of_pic_nums_minus1
2	Marca uma imagem de referência de longo prazo como "não usada para referência".	long_term_pic_num
3	Atribui um LongTermFrameIdx para uma imagem de referência de curto prazo, tornando-a de longo prazo.	difference_of_pic_nums_minus1 e long_term_frame_idx
4	Todas as imagens com LongTermFrameIdx maiores que o especificado são marcadas como "não usadas para referência".	max_long_term_frame_idx_plus1
5	Marca todas as imagens como "não usadas para referência".	-
6	Atribui um LongTermFrameIdx para a imagem corrente.	long_term_frame_idx

Tabela 2 – Operações de controle do gerenciamento de memória

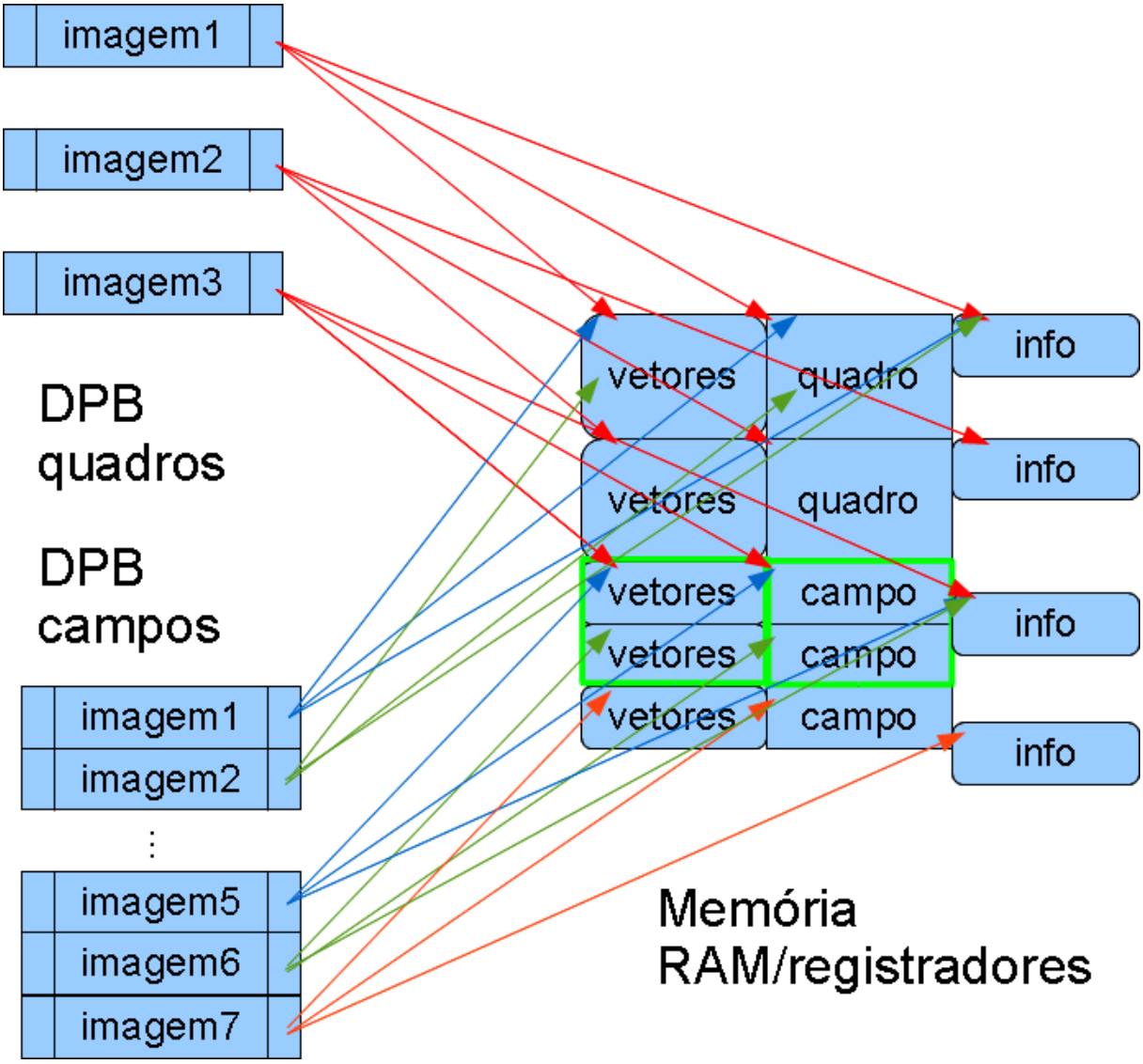


Figura 19 - Estrutura do DPB proposta

Todos essas operações listadas na tabela 2 são também descritas na norma [4] com detalhes, indicando o uso de seus argumentos.

A figura 19 mostra a implementação por *software* desenvolvida para o DPB, com uma estrutura dupla para campos e quadros, facilitando o chaveamento entre as estruturas de campo e quadro. Para implementação em hardware, o DPB foi unificado novamente para fins de economia de área em chip, sendo que sua estrutura foi minimizada para atender apenas o escopo deste trabalho, como será explicado mais adiante.

6 MÓDULO PROPOSTO

A proposta inicial é um método de gerenciamento de listas baseado na manutenção contínua de listas primárias, com as quais se trabalha aplicando operações simples, dependendo do tipo da imagem que está sendo decodificada, método descrito em [5]. Assim, pode-se agilizar e simplificar bastante a inicialização das listas, pois as listas estão sempre parcialmente inicializadas, e o processo é adaptado ao formato de entrelaçamento das imagens atual e a referenciada através de operações de descarte e alternância de campos.

As listas primárias consistem em listas com as imagens marcadas como referência, cada uma ordenada de acordo com um critério de ordenamento diferente, com a composição das quais é possível montar a lista 0 e a lista 1. Os critérios são o número do quadro (*frame_num*), o POC e o índice de longo prazo (LongTermFrameIdx).

6.1 Inserção de Imagens

As novas imagens são inseridas nestas listas na posição adequada ao fim de suas decodificações, restando apenas as operações de campo e quadro para serem aplicadas no momento de sua utilização. Os critérios de formação das listas primárias são mostrados na tabela 3.

Lista primária	**Critério de formação**	**Direção do ordenamento**
FNd	frame_num	Descendente
POCa	POC	Ascendente
LTa	LongTermFrameIdx	Ascendente

Tabela 3 – Critérios de formação das listas primárias

Assim, a cada nova imagem decodificada, seus valores de *frame_num*, POC e LongTermFrameIdx são confrontados com os das imagens nas listas primarias até encontrar a

posição onde ela deve ser inserida. Com a imagem já inserida nas listas primárias, é possível evitar vários ciclos de varreduras no DPB para montar cada uma das listas, a cada novo *slice* com predição temporal.

6.2 Fluxo de Operações

Para conformar o processo de armazenamento e gerenciamento de referências às variações no entrelaçamento das imagens, foi criada uma estrutura no *software* para gerenciar os dados individuais de cada uma delas e de suas representações progressivas e entrelaçadas, como foi explicado anteriormente. Cada quadro possui uma correspondência como um par de campos complementares, e os campos já pareados são agrupados para terem uma correspondência em quadro. É necessário manter a coerência entre as formas diferentes de representação das imagens, e também não se pode ignorar totalmente o formato original das mesmas, pois essa informação continua repercutindo em outros processos da decodificação, que não são abordados neste trabalho, como a compensação de movimento (MC).

Esses dados, individuais para cada imagem, seja quadro completo ou apenas um campo, par (*top*) ou ípar (*bottom*), são armazenados nos registros do DBP listados a seguir, de forma a associar os campos que formam um mesmo par, indicado por poucos atributos em comum.

Atributos comuns ao par de campos	**Atributos individuais**
Formato de entrelaçamento da decodificação	Existe no DPB
frame_num	Usado para referência de curto prazo
long_term_frame_idx	Usado para referência de longo prazo
Informações de macoblocos individuais	*Picture Order Count*

Tabela 4 – Atributos armazenados em cada registro do DPB no *software*

Para a implementação em *hardware*, esses dados foram mesclados em um único tipo de registro, como par de campos, mesmo para imagens originalmente quadros. que contem a seguinte listagem de atributos:

- Campo *top* existente no DPB
- Campo *bottom* existente no DPB
- Campo *top* usado como referência de curto prazo
- Campo *bottom* usado como referência de longo prazo
- Campo *top* usado como referência de longo prazo
- Campo *bottom* usado como referência de longo prazo
- *Top* POC
- *Bottom* POC
- Formato de entrelaçamento da decodificação
- frame_num
- long_term_frame_idx
- =>ref_idx_lx de cada macrobloco
- =>Vetores de movimento de cada macrobloco
- =>MbFieldDecodingFlag de cada macrobloco

Os itens marcados com => representam sub-estruturas onde são guardados valores referentes a cada um dos macroblocos da imagem. Como pode-se ver, é abrigado um par de campos ou um quadro sob a mesma estrutura de dados, que são implementados como registradores.

Essa estrutura, que foi inicialmente testada no *software*, o PRH.264, foi então adaptada para uma representação em *hardware* somente na parte relevante para as listas de referência. Os outros dados não implementados são utilizados pelo MC para inferir algumas informações necessárias pra decodificação de macroblocos individuais, e não serão tratados neste trabalho, necessitando de uma implementação futura por parte do módulo de MC ou modificação deste mesmo módulo.

6.3 Componentes Desenvolvidos

Para o decodificador em hardware do projeto, foram implementados alguns blocos operacionais que participam da formação e gerenciamento das listas de referência. O primeiro

componente que foi necessário desenvolver foi um bloco para o cálculo do *Picture Order Count*, que ainda não avia sido implementado até o momento, e que foi planejado para integrar o *parser* do decodificador.

O bloco faz uso de vários sinais internos do parser para calcular os valores de POC para a imagem sendo ela quadro ou campo. Ele é descrito em VHDL, com uma máquina de estados que executa as operações intermediarias até obter o valor final de POC para ambos os campos de cada quadro, devendo ser iniciada ao término da decodificação do cabeçalho do primeiro *slice* de cada imagem.

Em seguida, foram desenvolvidos dois pequenos sub-módulos, na figura 20, do decodificador de *slice header* do *parser* que não haviam sido implementados, mas previstos [9], responsáveis pela decodificação dos elementos sintáticos, ou parâmetros, relativos ao reordenamento de listas e as operações de controle de gerenciamento de memória. Suas saídas devem ser dirigidas a uma FIFO, uma fila de dados, para que as instruções nos elementos sintáticos sejam encaminhadas aos blocos desenvolvidos responsáveis por essas operações.

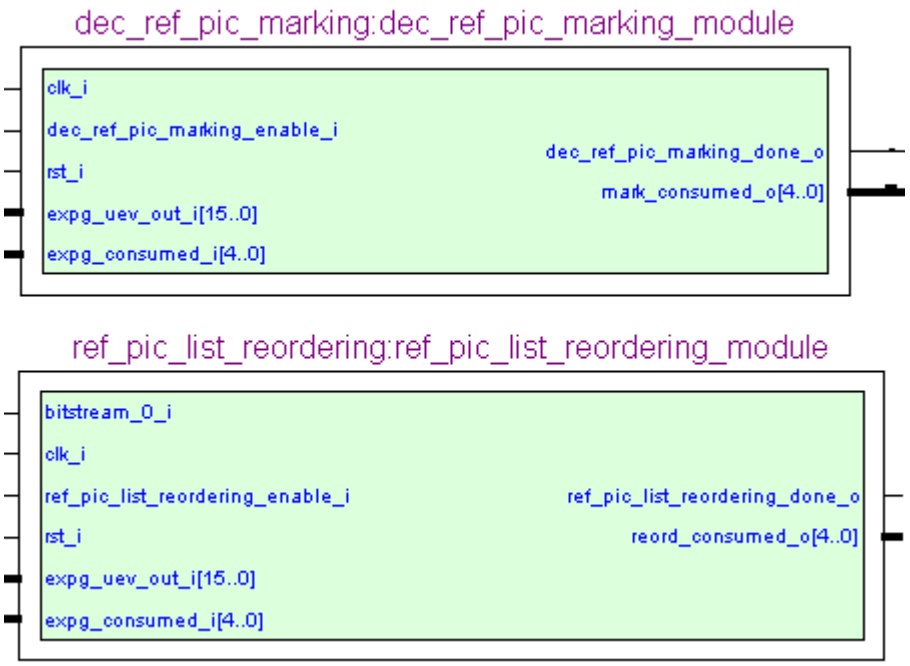


Figura 20 - Sub-módulos do parser desenvolvidos neste trabalho

Esses blocos, que integram o módulo de gerenciamento de listas, aparecem na figura 21, junto aos demais, sendo o *Reorder Unit* responsável pelo reordenamento e o *Mark Unit* responsável pelas operações MMCO, além da marcação ordinária das novas imagens no DPB ao término da decodificação. Os outros blocos são o *Insert Unit*, responsável pela inserção de novas imagens nas listas primárias, o banco de registradores das listas primárias e o banco de registradores do DPB.

Cada banco de registradores, seja das listas primárias como do DPB, possui 4 camadas, selecionadas por uma de suas entradas, usadas para diferentes tipos de registros, indexados por outra entrada, como visto na figura 22.

Nas listas primárias, cada camada contém um índice no DPB para um diferente critério de ordenamento, indexados pela ordem da lista, junto de sinais de validade do registro por campo, par e ímpar, sendo esses critérios os da tabela 3: *frame_num*, POC e LongTermFrameIdx. No DPB, há as mesmas camadas, desta vez guardando o valor do respectivo atributo da imagem apontada na memória pelo endereço armazenado na quarta camada.

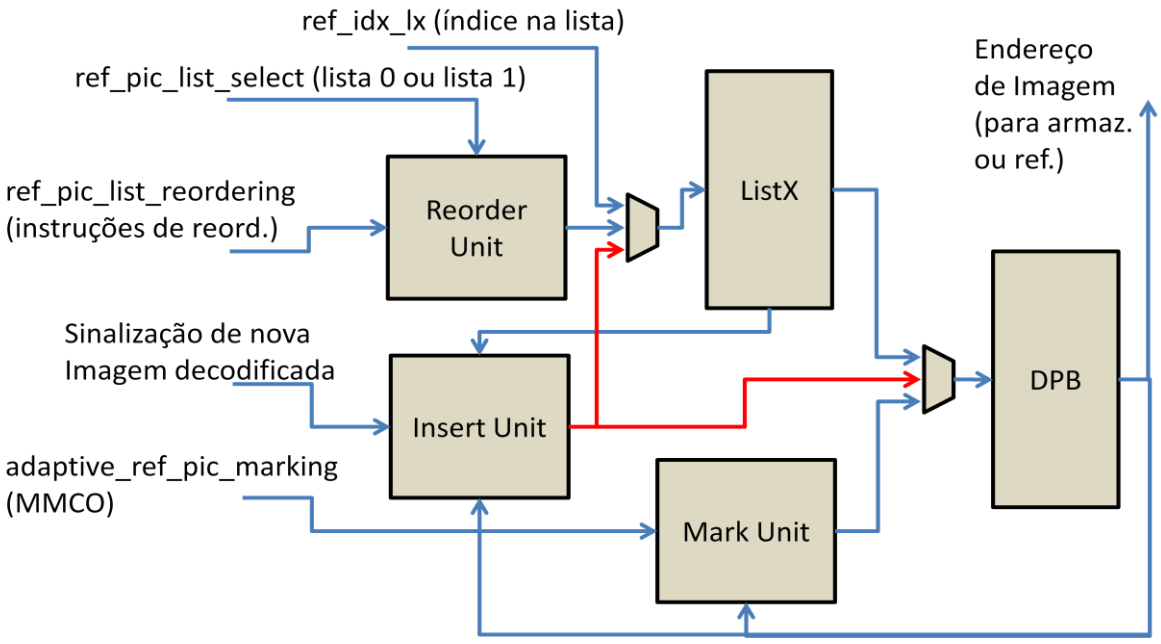


Figura 21 - Diagrama de blocos do módulo

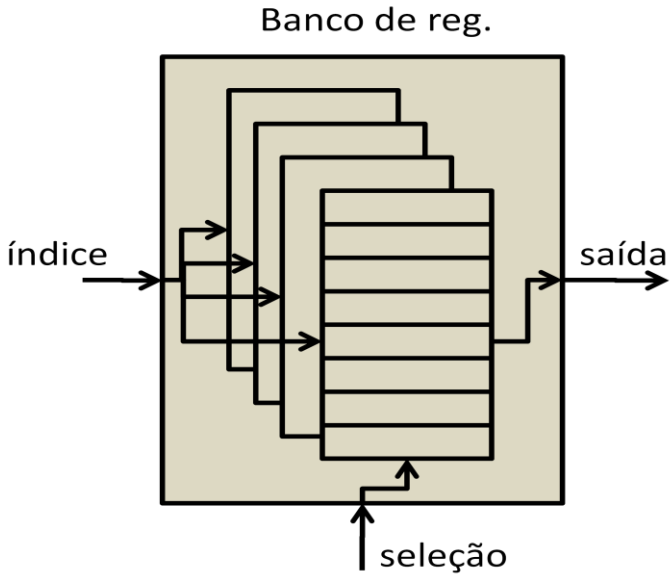


Figura 22 - Banco de registradores do módulo

O bloco de inserção, *Insert Unit*, é responsável também por procurar um espaço vazio no DPB para uma nova imagem que será decodificada, e fornecer seu endereço na memória para armazenamento da informação visual, assim como tem o papel de controlar os multiplexadores que dão acesso aos índices nos dois bancos de registradores. Com esse controle, ela pode assumir e varrer o DPB, checando a existência e a marcação de cada entrada. Quando encontra um endereço para ser usado, sinaliza e aguarda a instrução de inserção nas listas, que deve ser dada ao término da decodificação da imagem. Nesse momento ele procura a posição em cada lista primária que a imagem deve ser inserida, substitui a que esta lá deslocando todo resto da lista, posição a posição.

A unidade de marcação, *Mark Unit*, interpreta os elementos sintáticos esperados de uma FIFO que são operações de controle do gerenciamento de memória, e varre o DPB até achar a imagem com o frame_num, POC ou LongTermFrameIdx que deve ser marcada, executando a marcação. Ela está subordinada ao multiplexador comandado pela unidade de inserção, e espera a sua vez de usar o DPB.

A unidade de reordenamento implementada, *Reorder Unit*, considera os valores passados pela FIFO para buscar na lista e inserir o índice, deslocando os índices seguintes, estando subordinada também à unidade de inserção para poder ter acesso às listas primárias.

7 RESULTADOS

Testes demonstram que a implementação funciona, como um deles visto abaixo, na figura 23. Pode-se ver as formas de onda dos sinais da unidade de inserção, inicialmente buscando um endereço livre no DPB e posteriormente realocando as posições numa das listas primárias. A flecha superior mostra o sinal de requerimento de índice no DPB, a outra mostra o sinal de escrita na lista primária. Essa simulação foi feita com entradas arbitrárias, e não necessariamente reflete o ambiente em que o módulo será inserido, mas é um indicativo de que ele cumpre o proposto no trabalho.

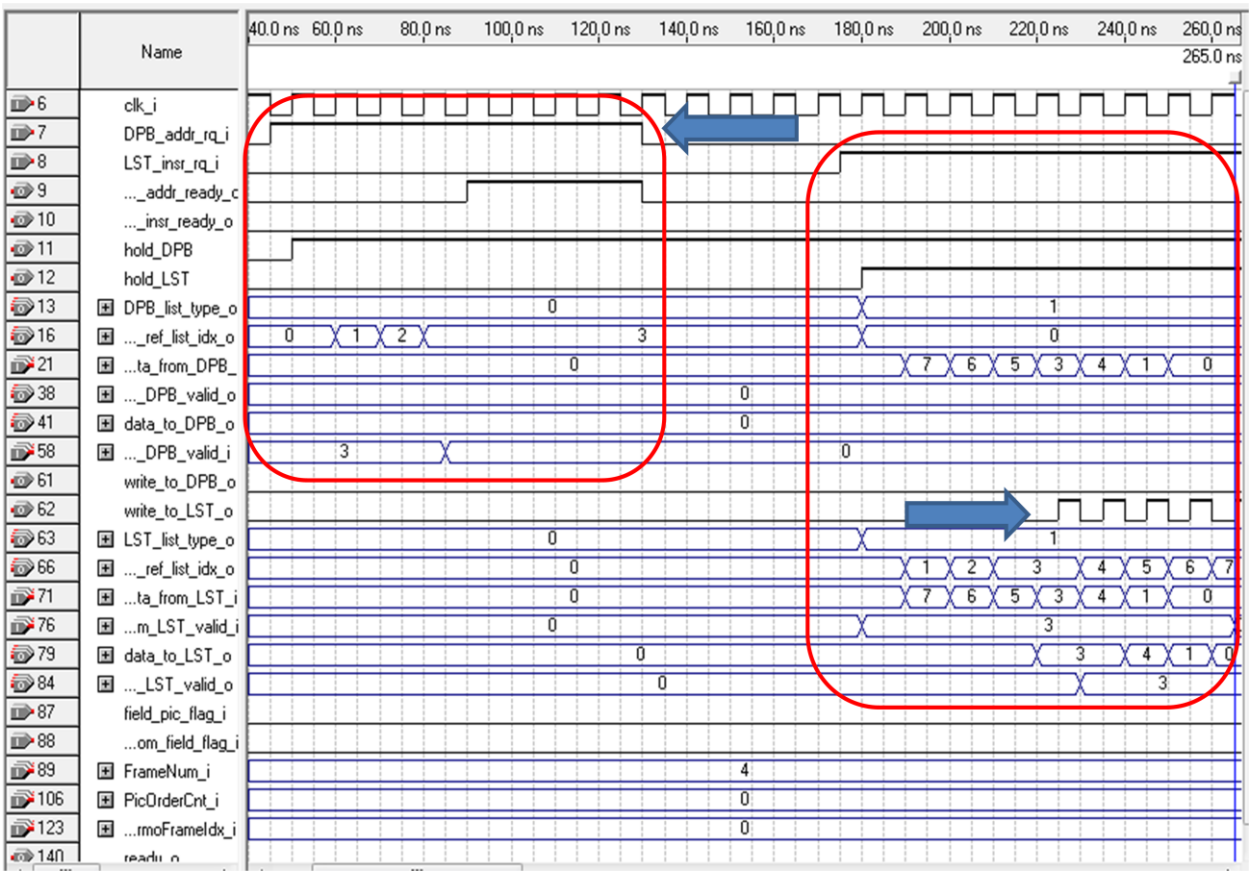


Figura 23 - Simulação da unidade de inserção

Os sub-módulos do decodificador de cabeçalho de *slice* já foram testados junto ao *parser*, e se comportaram como esperado, decodificando os novos elementos sintáticos de um *bitstream* real de uma emissora, como mostrado nas figuras 24.

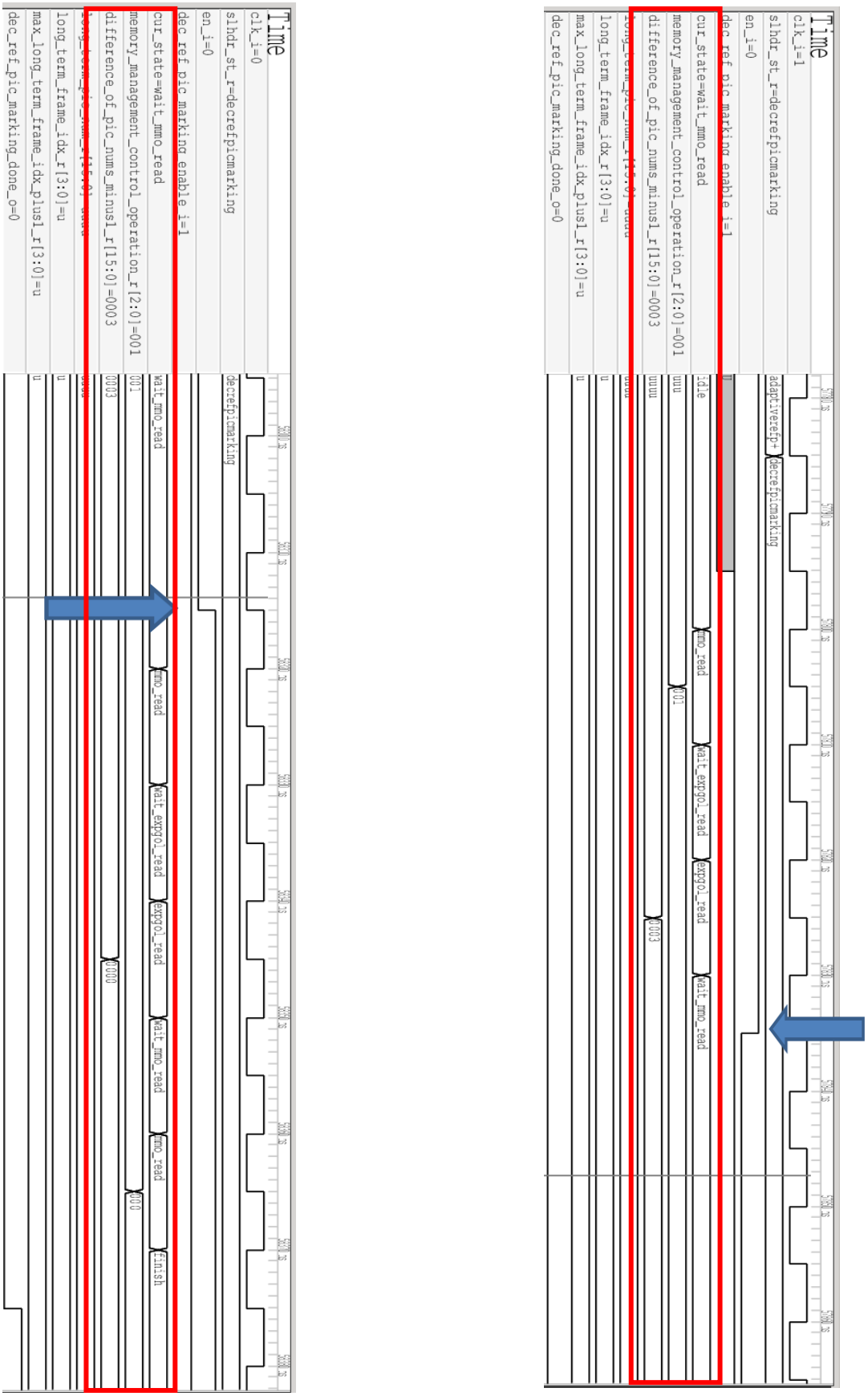


Figura 24 - Formas de onda do sub-módulo de MMCO integrado ao parser

Nessas formas de onda pode-se ver, na superior, o submódulo iniciando seu funcionamento ao receber a entrada de habilitração, e então passando pelos estados necessários, destacados nos retângulos, para obter os valores da MMCO que deve ser executada e seu argumento, em sequência até terminar as operações de controle do gerenciamento da memória. O outro sub-módulo tem funcionamento bastante semelhante, porem obtendo o tipo de reordenamento e o argumento usado para o cálculo do índice, ambos são entradas do processo de reordenamento executado pelo bloco referente. Os diagramas de estados desses dois submódulos são mostrados nas figuras 25 e 26. Ambas as máquinas podem ser paradas no caso do *bitstream* ter de ser parado pela FIFO de entrada estar vazia, aguardando a retomada da chegada dos dados, o que ocorre na figura 24 onde a seta marca, na forma de onda superior. O andamento é retomado sem erros na forma de onda inferior, assim que o *bitstream* é liberado para continuar sua leitura, igualmente marcado.

A tabela 5 mostra os dados de síntese desses sub-módulos decodificadores de MMCOs e instruções de reordenamento, quando ao número de funções lógicas, número de registradores e número de pinos.

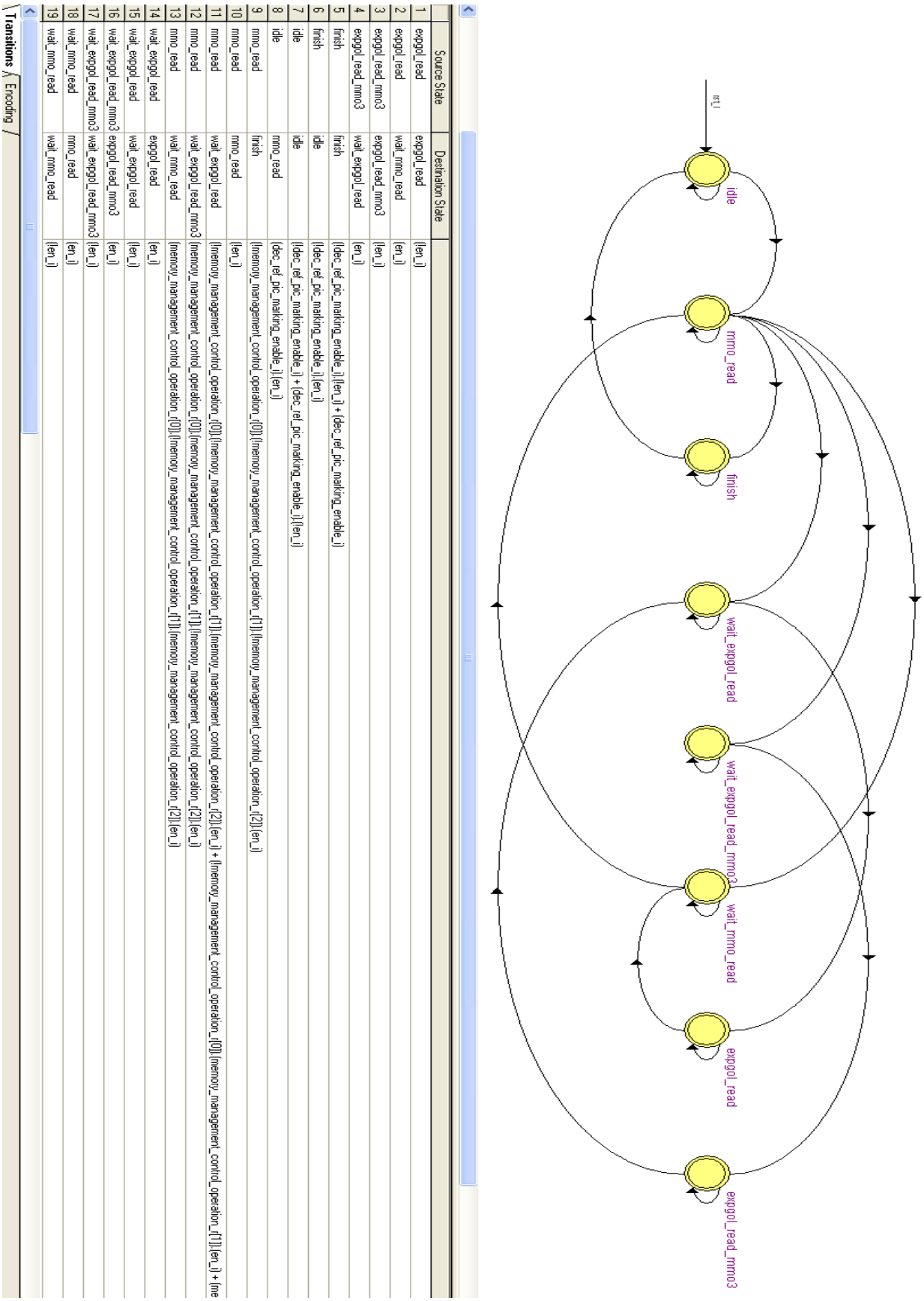
	Decodificador MMCO	Decodificador de Reord.
Funções lógicas	23	18
Registradores	52	43
Pinos mapeados	74	67

Tabela 5 – Dados de síntese dos sub-módulos

Como era esperado de componentes simples e pequenos, que tem somente a função de extrair e guiar os elementos sintáticos até as unidades que os utilizam. Já o bloco que calcula o POC, também a ser integrado no *parser*, se mostra bastante mais complexo, assim como os demais, como ilustrado na tabela 6. Os bancos são notavelmente formados por registradores.

	POC *Unit*	*Insert Unit*	*ListX / DPB*
Funções lógicas	709	160	332
Registradores	218	45	384
Pinos mapeados	217	125	19

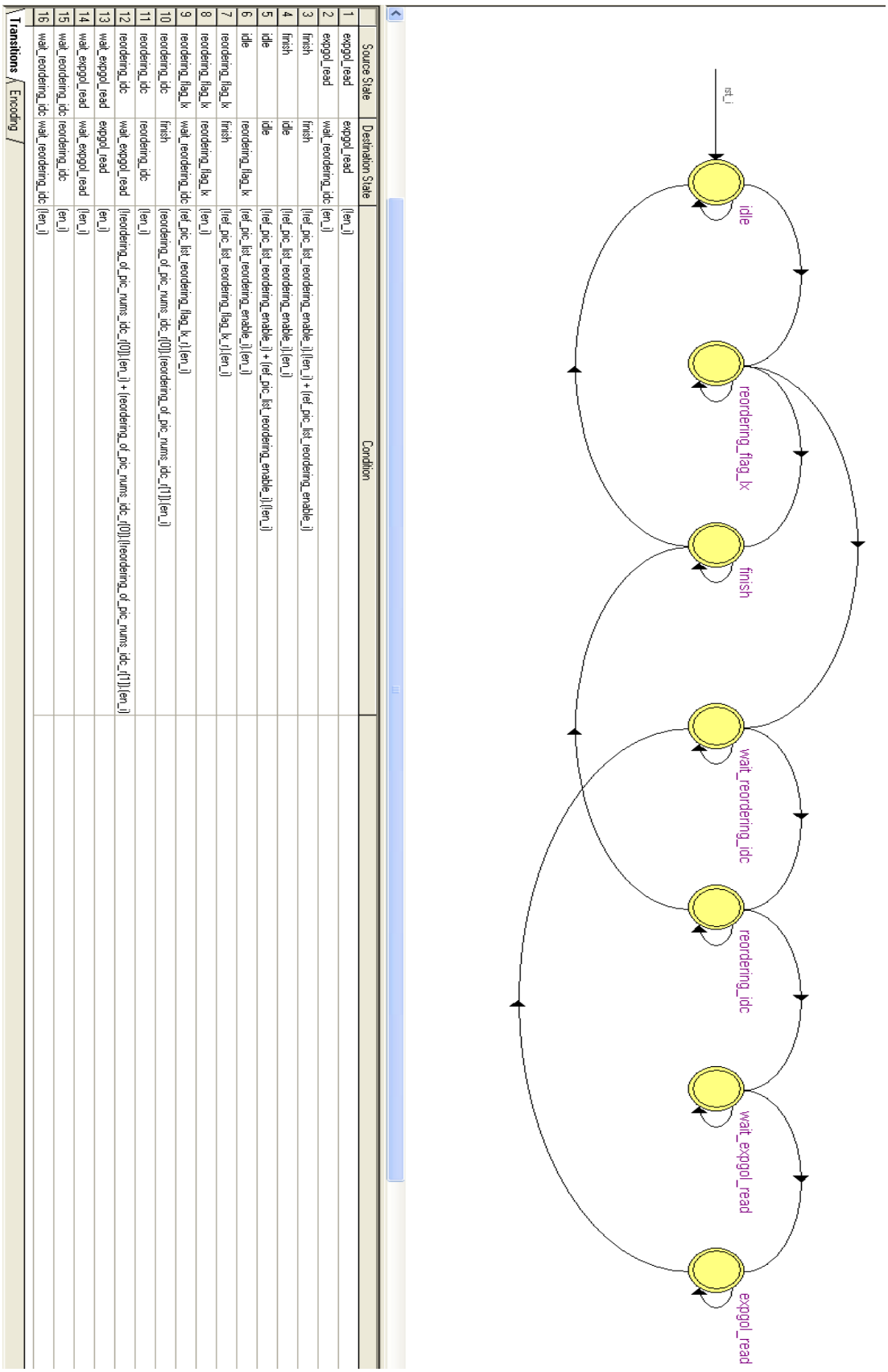
Tabela 6 – Dados de síntese dos blocos de POC, de inserção e bancos de registradores



	Source State	Destination State	
1	expgol_read	expgol_read	!len_i)
2	expgol_read	wait_mmo_read	(en_i)
3	expgol_read_mmo3	expgol_read_mmo3	!len_i)
4	expgol_read_mmo3	wait_expgol_read	(en_i)
5	finish	finish	!(dec_ref_pic_marking_enable_i).!(en_i) + (dec_ref_pic_marking_enable_i)
6	finish	idle	!(dec_ref_pic_marking_enable_i).(en_i)
7	idle	idle	!(dec_ref_pic_marking_enable_i) + (dec_ref_pic_marking_enable_i).!(en_i)
8	idle	mmo_read	(dec_ref_pic_marking_enable_i).(en_i)
9	mmo_read	finish	!(memory_management_control_operation_r[0]).!(memory_management_control_operation_r[1]).!(memory_management_control_operation_r[2]).(en_i)
10	mmo_read	mmo_read	!(en_i)
11	mmo_read	wait_expgol_read	!(memory_management_control_operation_r[0]).!(memory_management_control_operation_r[1]).!(memory_management_control_operation_r[2]).(en_i) + !(memory_management_control_operation_r[0]).!(memory_management_control_operation_r[1]).(en_i) + !(me
12	mmo_read	wait_expgol_read_mmo3	!(memory_management_control_operation_r[0]).!(memory_management_control_operation_r[1]).!(memory_management_control_operation_r[2]).(en_i)
13	mmo_read	wait_mmo_read	!(memory_management_control_operation_r[0]).!(memory_management_control_operation_r[1]).!(memory_management_control_operation_r[2]).(en_i)
14	wait_expgol_read	expgol_read	(en_i)
15	wait_expgol_read	wait_expgol_read	!(en_i)
16	wait_expgol_read_mmo3	expgol_read_mmo3	(en_i)
17	wait_expgol_read_mmo3	wait_expgol_read_mmo3	!(en_i)
18	wait_mmo_read	mmo_read	(en_i)
19	wait_mmo_read	wait_mmo_read	!(en_i)

Transitions | Encoding

Figura 25 - Máquina de estados do decodificador de MMCOs



	Source State	Destination State	Condition
1	expgol_read	expgol_read	!(en_i)
2	expgol_read	wait_reordering_idc	(en_i)
3	finish	finish	!(ref_pic_list_reordering_enable_i)|(!(en_i) + (ref_pic_list_reordering_enable_i)
4	finish	idle	!(ref_pic_list_reordering_enable_i)|(en_i)
5	idle	idle	!(ref_pic_list_reordering_enable_i) + (ref_pic_list_reordering_enable_i)|(!(en_i)
6	idle	reordering_flag_lk	(ref_pic_list_reordering_enable_i)|(en_i)
7	reordering_flag_lk	finish	!(ref_pic_list_reordering_flag_lk_i)|(en_i)
8	reordering_flag_lk	reordering_flag_lk	!(en_i)
9	reordering_flag_lk	wait_reordering_idc	(ref_pic_list_reordering_flag_lk_i)|(en_i)
10	reordering_idc	finish	(reordering_of_pic_nums_idc_[0])|(reordering_of_pic_nums_idc_[1])|(en_i)
11	reordering_idc	reordering_idc	!(en_i)
12	reordering_idc	wait_expgol_read	(!(reordering_of_pic_nums_idc_[0])|(en_i) + (reordering_of_pic_nums_idc_[0])|(!(reordering_of_pic_nums_idc_[1])|(en_i)
13	wait_expgol_read	expgol_read	(en_i)
14	wait_expgol_read	wait_expgol_read	!(en_i)
15	wait_reordering_idc	reordering_idc	(en_i)
16	wait_reordering_idc	wait_reordering_idc	!(en_i)

Figura 26 - Máquina de estados do decodificador de instruções de reordenamento

8 CONCLUSÃO

Com este trabalho foi possível, entender os processos envolvidos na codificação e decodificação de vídeo no padrão H.264, em especial no que diz respeito à predição temporal. Estes estudos serão apresentados ao laboratório, LaPSI, e acrescentados ao decodificador, ao longo do seu tempo de desenvolvimento que ainda se estende no momento da entrega deste documento. As descrições em VHDL podem e devem ser utilizadas futuramente para compor o decodificador, seguindo a etapa de integração dos módulos do projeto com os ajustes necessários.

Como experiência pessoal, o trabalho proporcionou uma experiência de projeto, assim como foi todo o tempo de trabalho de pesquisa no LaPSI. Foram exercitados os conhecimentos adquiridos ao longo do curso, principalmente no uso de ferramentas de simulação de sistemas digitais.

9 TRABALHOS FUTUROS

Como principal trabalho futuro tem-se a integração do módulo e os blocos desenvolvidos ao decodificador, a fim de testar, fazer os eventuais ajustes e validar o módulo. Com a ajuda do *software*, implementando todas as funcionalidades exigidas pelo padrão H.264 [4] e pela norma brasileira [3], será possível comparar saídas dos processos intermediários e avaliar se todo o módulo funciona como é esperado para diversos casos de decodificação de vídeos oriundos das transmissões das emissoras.

Somente após testes completos integrados ao sistema, com entradas extraídas de *bitstreams* reais será possível afirmar a completa operação das listas de referência em toda abrangência da norma [4], pois se trata de um processo bastante complexo e estocástico.

Outro objetivo futuro é desenvolver um acesso ao DPB por parte da saída de vídeo do sistema, pois após a decodificação das imagens, elas devem ser exibidas, marcadas como tal e possivelmente descartadas, se não forem mais usadas como referência.

BIBLIOGRAFIA

[1] HDTV Subjective Quality of H.264 vs. MPEG-2, with and without Packet Loss. Margaret H. Pinson, Stephen Wolf, and Gregory Cermak.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE T. CSVT, 13(7), pp. 560-576, (2003).

[3] Norma Brasileira ABNT 15602-1:2007 Televisão Digital Terrestre – Codificação de Vídeo, Áudio e Multiplexação. Parte 1: Codificação de Vídeo. Rio de Janeiro, Brasil, 2007.

[4] ITU-T Recommendation H.264 – Advanced video coding for generic audiovisual services, Video Coding Experts Group, Mar. 2005.

[5] Updating Strategy Based Architecture for Reference Picture Management in H.264/AVC. Kai Luo, Dongxiao Li, Member, IEEE, Lianghao Wang, Ming Zhang Institute of Information and Communication Engineering, Zhejiang University, Hangzhou 310027, P. R. China.

[6] SBTVD, R. H. Wiki Rede H.264 SBTVD. Julho, 2010.
http://www.lapsi.eletro.ufrgs.br/h264/wiki

[7] M. A. Lorencetti, W. T. Staehler, and A. A. Susin, "Reference C software H.264/AVC decoder for hardware debug and validation," in XXIII South Symposium on Microelectronics (SBC, ed.), (Bento Gonçalves - RS), pp. 127–130, 2008.

[8] I. E. G. Richardson, “H.264 and MPEG-4 Video Compression: Video Coding for the Next-generation Multimedia”, John Wiley and Sons, England, 2003.

[9] LORENCETTI, M. A. Parser em vhdl para decodificador de video h.264 para sbtvd. Projeto de Diplomação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.