UNIVERSIDADE FEDERAL DE PELOTAS

Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

Projeto, Síntese e Validação de Arquitetura do *Upsampling* para o Padrão H.264/SVC de Codificação de Vídeo Escalável

Fabiane Konrad Rediess

Pelotas, 2008

FABIANE KONRAD REDIESS

PROJETO, SÍNTESE E VALIDAÇÃO DE ARQUITETURA DO UPSAMPLING PARA O PADRÃO H.264/SVC DE CODIFICAÇÃO DE VÍDEO ESCALÁVEL

Trabalho de conclusão de curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Luciano Volcan Agostini

Pelotas, 2008

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia - UFPel

R315p Rediess, Fabiane Konrad

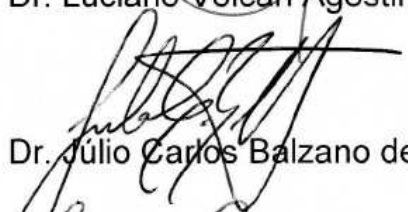
Projeto, síntese e validação de arquitetura do upsampling para o padrão H.264/SVC de codificação de vídeo escalável / Fabiane Konrad Rediess ; orientador Luciano Volcan Agostini. – Pelotas, 2008. – 67f. : il. - Monografia (Conclusão de curso). Curso de Bacharelado em Ciência da Computação. Departamento de Informática. Instituto de Física e Matemática. Universidade Federal de Pelotas. Pelotas, 2008.

1.Informática. 2.Padrão H.264/AVC. 3. H.264/SVC. 4.Escalabilidade. 5.Upsampling. I.Agostini, Luciano Volcan. II.Título.

CDD: 004.22

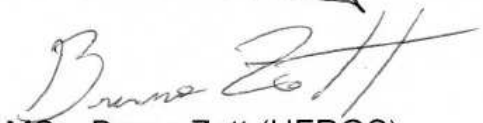
Banca examinadora:

Dr. Luciano Volcan Agostini (Presidente da Banca - UFPel)



Dr. Júlio Carlos Balzano de Mattos - (UFPel)

Dr. Leomar Soares da Rosa Júnior - (UFPel)



MSc. Bruno Zatt (UFRGS)

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, Elmira Konrad Rediess e Wilmar Rediess, por terem dado toda a base moral e por terem forjado todo o caminho para que eu chegasse até aqui. À minha mãe por ter lutado sozinha desde o falecimento do meu pai para me dar todo o suporte necessário.

Também merece agradecimento especial meu namorado, Diego Cândia Froner, por toda sua paciência e atenção dedicadas principalmente nos momentos mais difíceis. Por ser compreensivo nos momentos de esgotamento físico e mental em que o bom humor estava ausente. Pelos momentos felizes que me trouxeram paz de espírito e força para continuar. E principalmente por todo amor e carinho nesses dias que antecederam a entrega deste trabalho.

Agradeço também ao professor José Luís Güntzel, que foi quem abriu as portas para mim na iniciação científica. E que sempre foi exemplo e inspiração.

Ao meu orientador, Luciano Volcan Agostini, por estes anos de orientação na iniciação científica e também neste trabalho. Por indicar os caminhos e meios para que este trabalho pudesse ser feito e por toda a atenção dedicada.

Aos colegas de TCC, Guilherme Ribeiro Corrêa e Carolina Marques Fonseca e à ex-colega de GACI, atual mestranda da UFRGS, Thaísa Leal da Silva, pelos momentos de estudos e discussões que vivemos e também pelo apoio e disposição para ajudar que sempre demonstraram nas situações difíceis.

Resumo

REDIESS, Fabiane Konrad. **Projeto, Síntese e Validação de Arquitetura do Upsampling para o Padrão H.264/SVC de Codificação de Vídeo Escalável.** 2008. 67f. Monografia – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas

Considerando a popularização dos diferentes tipos de dispositivos que são capazes de manipular vídeos digitais, que vão desde aparelhos celulares até televisões de alta definição (HDTV), é inviável a transmissão de uma única representação de um vídeo para atender a todos estes tipos de dispositivos. Por exemplo, em aparelhos celulares, onde se tem restrições de tamanho, peso e consumo de energia, fica difícil justificar a implementação de um decodificador capaz de decodificar um vídeo com resolução HDTV e depois subamostrar essa informação para visualização no aparelho. O padrão H.264/AVC é o mais novo e mais eficiente padrão de compressão de vídeos, atingindo taxas de compressão duas vezes maiores que seus antecessores (como o MPEG-2, por exemplo). Mas a expectativa de um número cada vez maior de aplicações capazes de manipular vídeos digitais com diferentes capacidades e necessidades estimulou o desenvolvimento de uma extensão ao padrão H.264/AVC que fosse capaz de atender aos requisitos do mercado. Esta extensão é chamada H.264/SVC – *Scalable Video Coding* (WIEGAND et al., 2007). O conceito de escalabilidade está relacionado com camadas de enriquecimento, ou seja, partindo-se de uma camada base e adicionando-se camadas de enriquecimento é possível um aumento na resolução espacial e/ou na resolução temporal e/ou na qualidade do vídeo. Neste trabalho de conclusão de curso foi realizado o projeto e implementação de uma arquitetura para o filtro de *upsampling*. Este bloco é utilizado na região entre camadas na escalabilidade espacial (resolução). O filtro desenvolvido opera no contexto de um codificador/decodificador com duas camadas espaciais diádicas, cujas resoluções são QVGA para a camada base e VGA para a camada de enriquecimento. As arquiteturas foram descritas em VHDL e sintetizadas para dispositivos FPGAs das famílias Stratix II e Stratix IV (GX/E) da Altera e posteriormente foram validadas com a ferramenta ModelSim. Não foi encontrado, na literatura especializada, nenhum trabalho com o desenvolvimento de um filtro de *upsampling* para o padrão H.264

escalável. Entre os resultados de síntese obtidos para a arquitetura completa do *upsampling* chegou-se a atingir taxas de processamento entre 384 e 409 quadros por segundo com síntese direcionada para Stratix IV (GX/E). Através de estimativas para resoluções maiores, chegou-se a taxa de processamento entre 96 e 103 quadros por segundo para resolução 4VGA e para a resolução de 16VGA, a taxa ficou entre 24 e 25 quadros por segundo. Com isto, a arquitetura proposta atingiu processamento suficiente para tempo real.

Palavras-chave: Padrão H.264/AVC, H.264/SVC, Escalabilidade, *Upsampling*.

Abstract

REDIESS, Fabiane Konrad. **Design, Synthesis and Validation of an Architecture of Upsampling to the H.264/SVC Standard of Scalable Video Coding.** 2008. 67f. Monograph – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas

Considering the popularization of different types of devices that are able to handle digital videos, ranging from cell phones till high-definition televisions (HDTV), it is not possible to code and to transmit a single video stream to be used for all of these types of devices. For example, in cell phones, which are restricted in size, weight and power consumption, it is difficult to justify the implementation of a decoder that is able to decode a HDTV video resolution and then subsample it to display this information on the cell display. The H.264/AVC standard is the newest and most efficient video coding standard, reaching compression rates twice higher than the previous standards (like MPEG-2, for example). But the expectation of an increasing number of applications that are able to manipulate digital video with different capabilities and needs stimulated the development of an extension of the H.264/AVC standard. This extension is called H.264/SVC - Scalable Video Coding. The concept of scalability is related to enhancement layers, that is, there is a base layer and adding enhancement layers it is possible to increase the video quality and/or the video spatial resolution and/or the video temporal resolution. In this work, it was designed and implemented an architecture for the upsampling filter. This hardware module is used between spatial layers (resolution) in scalability. The filter was designed to work in the context of an encoder / decoder with two dyadic spatial layers, which would be QVGA resolution for the base layer and VGA resolution for the enhancement layer. The architectures were described in VHDL and synthesized targeting Stratix II and Stratix IV (GX / E) Altera FPGAs devices and they were validated with the ModelSim. We do not find, in the literature, any architecture designed to the upsampling filter of the H.264 scalable encoder. The results obtained through the synthesis of the

complete upsampling architecture shows that this architecture is able to achieved processing rates between 384 and 409 frames per second when the synthesis is targeted to Stratix IV (GX / E). The estimative for higher resolutions show processing rates between 96 and 103 frames per second for 4VGA resolution, and for the 16VGA resolution, the processing rate was between 24 and 25 frames per second. With these results, the designed architecture reached enough processing rates to process input videos at real time.

Keywords: H.264/AVC, H.264/SVC, Scalability, Upsampling.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

16VGA	*Sixteen Extended Video Graphics Array*
4VGA	*Four Extended Video Graphics Array*
AVC	*Advanced Video Coding*
Cb	*Chrominance blue*
Cr	*Chrominance red*
FPGA	*Field Programmable Gate Array*
HDTV	*High Definition Digital Television*
HSI	*Hue, Saturation, Intensity*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
ITU-T	*International Telecommunication Union - Telecommunication*
JSVM	*Joint Scalable Video Model*
JVT	*Join Video Team*
ME	*Motion Estimation*
MC	*Motion Compensation*
MCTF	*Motion Compensation Temporal Filtering*
MPEG	*Moving Picture Experts Group*

QVGA	*Quarter Vídeo Graphics Array*
RGB	*Red, Green, Blue*
SBTVD	Sistema Brasileiro de Televisão Digital
SDTV	*Standard Definition Television*
SNR	*Signal-to-noise Ratio*
SVC	*Scalable Video Coding*
VGA	*Video Graphics Array*
VHDL	*VHSIC Hardware Description Language*

SUMÁRIO

1 INTRODUÇÃO

A compressão é essencial para o sucesso das aplicações que manipulam vídeos digitais, pois um vídeo não comprimido utiliza uma quantidade de bits muito elevada para representar cada pixel da imagem. O padrão H.264/AVC (JVT, 2003) é considerado estado-da-arte em compressão de vídeo, pois ele conseguiu alcançar taxas de compressão duas vezes maiores do que os padrões anteriores, como o MPEG-2, que é atualmente usado em DVDs. No entanto, para alcançar estas taxas, o H.264/AVC tornou-se quatro vezes mais complexo computacionalmente do que os padrões anteriores (RICHARDSON, 2003).

Considerando os diferentes tipos de dispositivos que são capazes de manipular vídeos digitais, que vão desde aparelhos celulares até televisões de alta definição (HDTV), é inviável a transmissão de uma única representação de um vídeo para atender a todos estes tipos de dispositivos (SEGALL e SULLIVAN, 2007). Por exemplo, em aparelhos celulares, onde se tem restrições de tamanho, peso e consumo de energia, fica difícil justificar a implementação de um decodificador capaz de decodificar um vídeo com resolução HDTV e depois subamostrar essa informação para visualização no aparelho. Além disso, a transmissão de muitos dados que não serão aproveitados causa um desperdício do canal de recepção do aparelho.

A expectativa de um número cada vez maior de aplicações capazes de manipular vídeos digitais e com diferentes capacidades e necessidades estimulou o desenvolvimento de uma extensão ao padrão H.264/AVC que fosse capaz de atender aos requisitos do mercado. Esta extensão é chamada H.264/SVC – *Scalable Video Coding* (WIEGAND et al., 2007). O H.264/SVC provê suporte a escalabilidade, que é o suporte a múltiplos níveis de resoluções de exibição, taxa de quadros ou qualidade a partir de um único *bitstream*. O conceito de escalabilidade está relacionado com camadas de enriquecimento, ou seja, partindo-se de uma camada base e adicionando-se camadas de enriquecimento é possível um aumento na resolução da imagem e, conseqüentemente, na sua qualidade.

O SVC provê suporte a múltiplas resoluções de exibição com um único *bitstream*, isto é chamado de escalabilidade espacial. Mas o SVC também suporta escalabilidade temporal, que está relacionada com a taxa de quadros, ou seja, alguns quadros podem ser descartados; e a escalabilidade de qualidade, que é o enriquecimento na fidelidade dos quadros mantendo-se a resolução entre as camadas. O SVC também permite a combinação dessas técnicas (SEGALL e SULLIVAN, 2007).

O foco deste trabalho está em um dos blocos necessários para a ligação entre as camadas de enriquecimento na escalabilidade espacial, o bloco *upsampling*. Este bloco é responsável pela sobreamostragem das informações de codificação presentes na camada de resolução mais baixa para a camada de resolução mais alta.

Entre os objetivos gerais deste trabalho estão: (1) contribuir com as investigações relacionadas ao Sistema Brasileiro de Televisão Digital e (2) contribuir para a consolidação da pesquisa sobre compressão de vídeo junto ao Grupo de Arquiteturas e Circuitos Integrados da UFPel (GACI).

Os objetivos específicos deste trabalho são: (1) entender e dominar os conceitos de compressão de vídeo relacionados com a escalabilidade, (2) identificar as inovações implementadas pela extensão escalável do padrão H.264/AVC e (3) definir, implementar e validar uma arquitetura em hardware para o bloco *upsampling*.

A principal motivação para este trabalho diz respeito ao escopo onde o mesmo está inserido. Esta investigação é parte de um trabalho de investigação de soluções de hardware para a compressão de vídeo para as futuras gerações do Sistema Brasileiro de Televisão Digital (SBTVD), que apresenta como uma de suas principais idéias, a mobilidade e a escalabilidade, para que possa permitir o acesso a TV a partir de diferentes tipos de dispositivos.

Este trabalho de conclusão está organizado como segue: no capítulo 2 estão os conceitos básicos relacionados à compressão de vídeos digitais, assim como está apresentado o padrão H.264/AVC de compressão de vídeo. No capítulo 3 encontram-se os principais conceitos de escalabilidade e as principais inovações implementadas pela extensão escalável do padrão H.264/AVC. O funcionamento do bloco *upsampling* está descrito no capítulo 4. O capítulo 5 apresenta a arquitetura desenvolvida para o bloco alvo deste trabalho, desde sua descrição em linguagem de descrição de hardware, até sua validação. No capítulo 6, são apresentados os

resultados de síntese obtidos. Por fim, são apresentadas, no capítulo 7, as conclusões deste trabalho de conclusão de curso.

2 PRINCÍPIOS DA COMPRESSÃO DE VÍDEO

Neste capítulo serão apresentados os principais conceitos relacionados à compressão de vídeos digitais e também será apresentado resumidamente o padrão H.264/AVC de compressão de vídeo.

2.1 Introdução ao Vídeo Digital

Um vídeo digital é formado por uma seqüência de imagens estáticas, chamadas de quadros (ou *frames*). Estes quadros devem ser capturados a uma taxa entre 24 a 30 quadros por segundo para que o sistema visual humano consiga perceber a seqüência de imagens como um vídeo, ou seja, para que se tenha a idéia de um movimento contínuo.

Cada quadro de um vídeo é formado por uma matriz de pontos que são chamados de pixels. Para facilitar o processamento, os quadros são geralmente divididos em matrizes menores, que são chamadas de blocos. Nos padrões mais atuais, um bloco de tamanho 16x16 pixels é chamado de macrobloco. Cada macrobloco pode ser dividido em blocos menores de 8x16, 16x8 e 8x8 pixels e, caso o tamanho escolhido seja o de 8x8, este pode ser dividido novamente em blocos de 4x8, 8x4 e 4x4 pixels. Estes diferentes tamanhos de blocos são utilizados, pois, em geral, uma partição grande é apropriada para áreas mais homogêneas do quadro e uma partição menor tende a ser mais apropriada para áreas com muitos detalhes (AGOSTINI, 2007).

Os macroblocos podem ser de três diferentes tipos: I, P ou B. Macroblocos do mesmo tipo são agrupados em *slices*. Os macroblocos do tipo I são codificados através da codificação intra-quadro, que utiliza amostras contidas no *slice* atual, explorando a redundância espacial dentro do *slice*. Estes macroblocos também podem ser codificados através do modo I_PCM, onde são transmitidos os valores das amostras diretamente, sem nenhuma codificação.

Os macroblocos tipo P são codificados utilizando a codificação inter-quadros, que utiliza amostras contidas em um dos quadros de referência

previamente codificados. Este tipo de codificação visa explorar a redundância temporal que existe entre os quadros de uma seqüência de vídeo. Os quadros de referência são organizados em uma lista, chamada pelo padrão de lista 0 (ITU-T, 2003).

Macroblocos B também são codificados pela codificação interframe, no entanto, cada partição pode utilizar informações de até dois quadros de referência diferentes. As referências são organizadas em duas listas, chamadas de lista 0 e lista 1.

2.2 Espaço de Cores e Subamostragem

Existem várias formas de representar as cores de forma digital. O sistema usado para representar as cores é chamado de espaço de cores. A escolha de qual espaço de cores será utilizado é essencial para a eficiência da codificação.

Vários espaços de cores são utilizados para representar imagens digitais, entre eles, RGB, HSI e YCbCr (SHI e SUN, 1999). O sistema RGB é um dos mais difundidos e representa, em três matrizes distintas, as três cores primárias: vermelho, verde e azul (AGOSTINI, 2007). No sistema YCbCr, as três componentes são: luminância (Y), que são os tons de cinza da imagem, crominância azul (Cb) e crominância vermelha (Cr) (BHASKARAN e KONSTANTINIDES, 1997).

Os componentes R, G e B possuem um elevado grau de correlação, o que não é desejável do ponto de vista da compressão de vídeos, pois esta correlação dificulta o processo de codificação, resultando em uma redução na eficiência do processo. Por isso, a compressão de vídeos é aplicada para espaços de cores do tipo luminância e crominância, como o YCbCr (RICHARDSON, 2002).

Como o sistema visual humano é mais sensível a informações de luminância do que de crominância (GONZALEZ, 2003), os padrões de compressão podem explorar essa característica e aumentar a eficiência de codificação, reduzindo a taxa de amostragem dos componentes crominância em relação aos de luminância (RICHARDSON, 2002).

Existem diversas formas de relacionar os componentes YCbCr para realizar a subamostragem, entre as principais estão o formato 4:4:4, onde para cada quatro amostras de Y, existem quatro amostras de Cb e quatro de Cr; e o formato 4:2:0, onde para cada quatro amostras de Y, existem uma de Cb e uma de Cr.

Com o formato 4:2:0, apenas um quarto das informações de cada componente de crominância são considerados. O restante é descartado sem, no entanto, causar impacto visual perceptível. Isso implica em uma taxa de compressão de 50%, considerando apenas a subamostragem de cor.

Neste trabalho, é usado o espaço de cores YCbCr com subamostragem de cor no formato 4:2:0.

2.3 Redundâncias

Os vídeos digitais sem compressão necessitam de uma enorme quantidade de bits de memória para serem armazenados e de elevadas taxas de transmissão para que possam ser transmitidos. Isso implica em custos muito elevados em termos de transmissão e armazenamento, o que pode inviabilizar o desenvolvimento de produtos para esta área caso a compressão não seja utilizada. Por outro lado, é importante salientar que esses vídeos, em geral, possuem outra importante propriedade intrínseca: apresentam elevado grau de redundância. Isto significa que a maior parte dos dados necessários para representar o vídeo digitalizado se repete. O objetivo da compressão é justamente o desenvolvimento de técnicas para eliminação dessa informação redundante.

Existem, basicamente, três tipos de redundâncias:

2.3.1 Redundância Espacial

É a redundância presente na correlação existente entre os pixels espacialmente distribuídos em um mesmo quadro, também chamada de redundância intra-quadro (GHANBARI, 2003).

2.3.2 Redundância Temporal

É a redundância causada pela correlação entre os quadros próximos em um vídeo e é também chamada de redundância inter-quadros (GHANBARI, 2003). Muitas regiões dos vídeos não mudam seus valores de pixels entre quadros vizinhos, como, por exemplo, um fundo que não foi alterado de um quadro para outro. A exploração desta característica leva a elevadas taxas de compressão.

2.3.3 Redundância Entrópica

É a redundância relacionada com a forma como os símbolos codificados estão representados e não propriamente com o conteúdo da imagem. A entropia é uma medida da quantidade média de informação transmitida por símbolo do vídeo (SHI, 1999). Os codificadores, ao explorar esta redundância, têm por objetivo transmitir o máximo de informação possível por símbolo codificado e, deste modo, representar mais informações com um número menor de bits.

2.4 Padrão H.264/AVC

O padrão H.264/AVC foi desenvolvido com o objetivo principal de dobrar a taxa de compressão de seu predecessor, o padrão MPEG-2, que é atualmente utilizado em DVDs. Este objetivo foi conquistado através do uso conjunto de diversas técnicas avançadas e inovadoras de compressão e que acarretou em um significativo acréscimo na sua complexidade computacional, que é cerca de quatro vezes maior que a do MPEG-2 (RICHARDSON, 2003).

A taxa de compressão atingida pelo H.264/AVC varia de acordo com a qualidade pretendida para o vídeo codificado. Com uma taxa de 50:1, a redução na qualidade do vídeo é, em geral, pequena e pouco perceptível (ou mesmo imperceptível) ao sistema visual humano. Neste caso, o vídeo gerado possui uma qualidade equivalente a vídeos MPEG-2 usados em DVDs. Para uma primeira avaliação dos ganhos em compressão do codificador H.264/AVC em relação a vídeos não comprimidos, serão usados os exemplos de vídeos com resolução de 720x480 (SDTV) e 1920x1080 (HDTV-1080p) pixels a 30 quadros por segundo e utilizando 24 bits por pixel. Além disso, a taxa de compressão de 50:1 será tomada como referência. Para SDTV, a taxa de transmissão necessária com o uso do H.264/AVC é de cerca de 2,5 Mbps, enquanto um vídeo não comprimido usaria 249 Mbps. Apenas 10 minutos de vídeo SDTV, que ocupariam quase 19GB no vídeo não comprimido passam a ocupar apenas 380KB com o uso do H.264/AVC. Para vídeos HDTV-1080p, a taxa de transmissão necessária ao H.264/AVC fica abaixo de 30 Mbps, contra os 1,5 Gbps necessários sem o uso de compressão. A seqüência de 10 minutos passaria a ocupar 2,2MB ao invés dos 112 GB necessários ao vídeo sem compressão.

A Fig. 1 apresenta um diagrama em bloco do codificador do padrão H.264/AVC. Os principais blocos do codificador são: a predição inter-quadros que é formada pela estimação de movimento (ME) e pela compensação de movimento (MC) e que explora a redundância temporal, a predição intra-quadro que explora a redundância espacial, as transformadas e a quantização diretas, as transformadas e a quantização inversas, o filtro e a codificação de entropia.

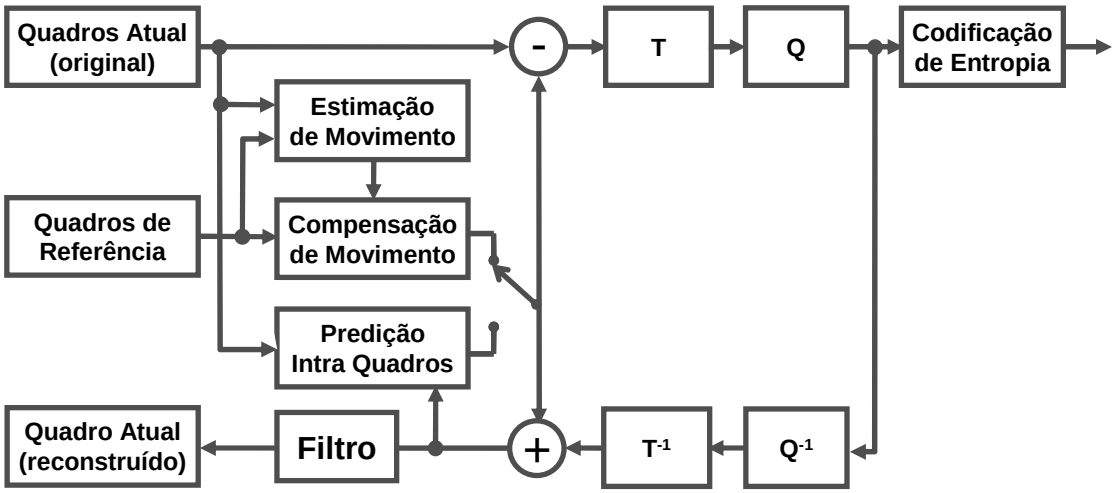


Figura 1 – Diagrama em blocos do codificador H.264/AVC

O módulo de codificação inter-quadros é responsável por reduzir a redundância temporal, através da comparação dos blocos do quadro atual com os blocos do quadro de referência. Na codificação inter-quadros, o quadro codificado é usado como quadro de referência para a codificação do próximo quadro.

O módulo de codificação intra-quadro é responsável por reduzir a redundância espacial, utilizando, para tanto, apenas a informação do quadro atual em processamento.

Após um bloco do quadro atual, alvo da codificação, ser codificado pela codificação intra-quadro ou pela codificação inter-quadros, é realizada uma subtração entre os valores do bloco original e os resultados da codificação. Esta diferença é chamada de resíduo. O resíduo, então, é enviado para os módulos responsáveis por reduzir a redundância espacial no domínio das freqüências. A primeira operação nesta direção é a transformada (módulo T na Fig. 1). O módulo T transforma a informação do domínio espacial para o domínio das freqüências. Neste domínio, a Quantização pode ser aplicada, reduzindo a redundância espacial presente nos resíduos.

Por fim, a codificação de entropia reduz a redundância entrópica, que está relacionada à forma como os dados são codificados.

O decodificador H.264/AVC é formado por um subgrupo destes blocos: compensação de movimento, predição intra-quadro, transformadas e quantização inversas, filtro e decodificação de entropia.

3 ESCALABILIDADE

Este capítulo mostrará um breve histórico da codificação de vídeo com escalabilidade, os tipos de escalabilidade e as inovações implementadas pela extensão escalável do padrão H.264/AVC.

3.1 Justificativas para o Uso da Escalabilidade

Atualmente surge um mercado com uma diversidade cada vez maior de dispositivos com variadas resoluções, que vão desde dispositivos com baixíssimas resoluções, como celulares, até dispositivos com resoluções elevadas, como as TVs de alta definição.

Para atender a este mercado, uma alternativa seria os equipamentos que usam baixa resolução e que apresentam restrições de tamanho, peso e, principalmente, consumo de energia, terem a capacidade de receber e decodificar o mesmo sinal que está sendo transmitido para os aparelhos de TV de alta definição. No entanto, isto resultaria em um grande custo computacional, ocasionando elevado consumo de energia.

Outra alternativa, seria a transmissão de dois (ou mais) sinais independentes, o chamado *simulcast* (SEGALL e SULLIVAN, 2007). Onde é transmitido um sinal que atenda as necessidades dos equipamentos de baixa resolução e outro que satisfaça as necessidades dos aparelhos de alta definição. No entanto, esta alternativa acarretaria um grande desperdiço de banda de transmissão.

A escalabilidade é uma alternativa extremamente atrativa, pois a escalabilidade trabalha com o conceito de camadas de enriquecimento, onde existe uma camada mais baixa, com menos informação e uma ou várias camadas superiores que vão adicionando informações para alcançar resoluções mais elevadas. Neste caso, existe apenas um sinal sendo transmitido e o equipamento receptor irá selecionar apenas sua camada de interesse, processando apenas as informações necessárias.

3.2 Histórico

Os padrões anteriores de compressão de vídeo (MPEG-2, por exemplo) já incluíam uma série de ferramentas de forma que os modos de escalabilidade mais importantes fossem suportados (SHWARZ, MARPE e WIEGAND, 2007). Mas estes perfis dos padrões anteriores raramente foram utilizados, devido ao fato de que, comparando com os perfis não escaláveis, o uso da escalabilidade espacial e de qualidade traziam uma significativa perda na eficiência de codificação, assim como, um aumento na complexidade do decodificador.

As atividades de padronização sobre o SVC começaram em dezembro de 2001 na 58ª reunião do MPEG através de um grupo *ad hoc*. O trabalho deste grupo e de seu sucessor resultaram em uma chamada para apresentação de propostas para uma nova atividade de padronização em outubro de 2003, a qual resultou no *Scalable Video Model* 1.0 que resumiu os conceitos promissores submetidos em março de 2004. Em janeiro de 2005, esta atividade de padronização tornou-se um item de trabalho do JVT – *Joint Vídeo Team*. (WIEN, SCHWARZ e OELBAUM, 2007). A versão final da norma foi publicada em novembro de 2007 (JVT, 2007).

3.3 Extensão Escalável do Padrão H.264/AVC

O objetivo da padronização do H.264/SVC é permitir a codificação de um *bitstream* de um vídeo de alta qualidade que contenha um ou mais subconjuntos de *bitstreams* que possam ser decodificados de forma independente com uma complexidade e qualidade similar à alcançada usando o mesmo decodificador H264/AVC existente e a mesma quantidade de dados no subconjunto do *bitstream* (SCHWARZ, MARPE e WIEGAND, 2007).

A Fig. 2 mostra uma estrutura típica de um codificador H.264/SVC com duas camadas. O *bitstream* da camada base é gerado de forma a ser compatível com o padrão não escalável H.264/AVC.

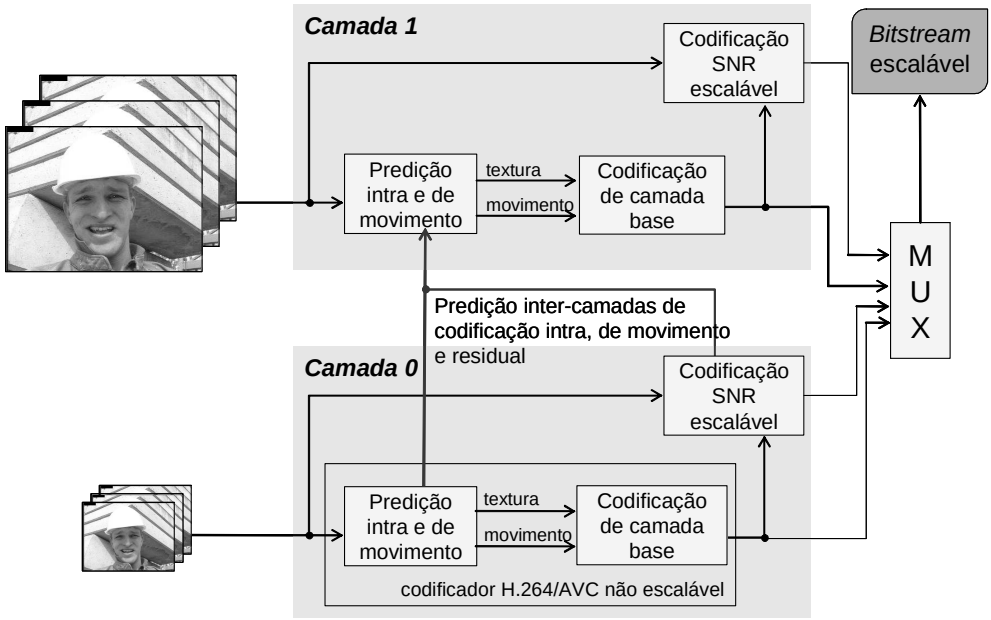


Figura 2 – Diagrama em blocos de um codificador da extensão escalável do padrão H.264/AVC

A Fig. 3 mostra uma ampliação na região entre camadas do codificador representado na Fig. 2 e mostra a localização do módulo *upsampling*. Este é o bloco alvo deste trabalho e, para ele, foi projetada a arquitetura que será apresentada nos próximos capítulos.

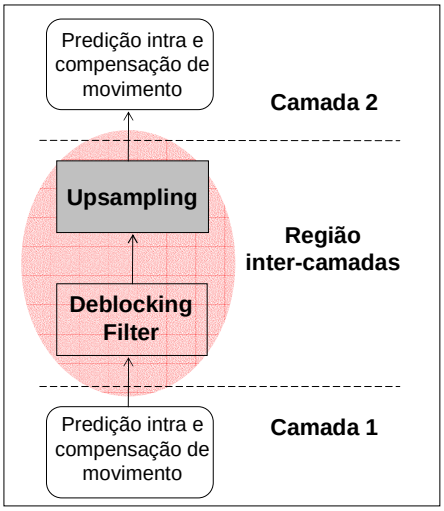


Figura 3 – Localização do *upsampling* no codificador H.264/SVC

O padrão SVC explora três tipos de escalabilidade: temporal, espacial e de qualidade, que serão detalhadas nas próximas seções.

3.3.1 Escalabilidade Temporal

A escalabilidade temporal é a técnica que permite que um único *bitstream* suporte múltiplas taxas de quadros. Ela é geralmente implementada por uma estrutura de predição temporal pré-determinada pelo padrão. No SVC, podem ser utilizados vários níveis através da estrutura de quadros tipo B hierárquica e também é utilizada a ferramenta de pré-processamento chamada MCTF – *motion-compensated temporal filtering* (compensação de movimento com filtragem temporal), para aumentar a eficiência de codificação (HUANG, et al, 2007).

3.3.2 Escalabilidade de Qualidade

A escalabilidade de qualidade é comumente chamada de escalabilidade de fidelidade ou escalabilidade SNR – *signal-to-noise ratio* (taxa sinal ruído) (SCHWARZ, MARPE e WIEGAND, 2007). Este tipo de escalabilidade poder ser considerado um caso especial em que tanto a resolução quanto a taxa de quadros se mantém entre as camadas base e de enriquecimento e apenas a qualidade do vídeo é alterada, ou seja, apenas a quantização é aplicada com constantes menores para vídeos de maior qualidade e com constantes maiores para vídeos com menor qualidade (SCHWARZ, MARPE e WIEGAND, 2007).

3.3.3 Escalabilidade Espacial

Para permitir a escalabilidade espacial, o SVC segue a abordagem tradicional de codificação multicamadas, que era também usada nos padrões anteriores como H.262|MPEG-2, H.263 e MPEG-4 (SCHWARZ, MARPE e WIEGAND, 2007). Cada camada corresponde a uma resolução espacial suportada.

Os quadros de diferentes camadas espaciais são codificados com informações de predição e parâmetros de movimento específicos daquela camada. Para melhorar a eficiência de codificação da camada de enriquecimento em relação ao *simulcast*, foram introduzidos mecanismos de predição entre camadas. Um codificador pode escolher livremente que informação da camada de referência será explorada (WIEN, SCHWARZ e OELBAUM, 2007). A seguir serão apresentados os mecanismos de predição entre camadas introduzidos pelo padrão.

3.3.4 Predição Entre Camadas (*Inter Layer Prediction*)

No SVC existem três mecanismos que podem ser utilizados na predição entre camadas: Predição de Movimento, Predição Residual e Predição Intra. A

seguir são descritas as três técnicas utilizadas. As descrições estão restritas aos casos diádicos da escalabilidade espacial que são caracterizados pela duplicação na altura e na largura da imagem (SEGALL e SULLIVAN, 2007). Esta restrição na descrição foi feita pois o foco deste trabalho está justamente na predição entre camadas para o caso diádico.

3.3.4.1 Predição de Movimento Entre Camadas (*Inter Layer Motion Prediction*)

Para utilizar a informação de movimento de uma camada mais baixa nas camadas de enriquecimento foi criado um novo tipo de macrobloco. Este novo tipo é sinalizado pelo elemento "*base mode flag*". Quando este tipo de macrobloco é utilizado, apenas um sinal residual é transmitido e nenhuma outra informação como modos de predição intra ou parâmetros de movimento é enviada.

Quando um macrobloco na camada de enriquecimento é codificado com "*base mode flag*" igual a 1 e o submacrobloco 8x8 correspondente na camada de referência (também chamado de co-localizado) tiver sido codificado com predição inter quadros, o macrobloco na camada de enriquecimento também será codificado com predição inter quadros. Neste caso, os dados de particionamento, os índices de referência (indicam qual quadro da lista foi usado como referência) e os vetores de movimento serão derivados do bloco 8x8 co-localizado da camada de referência (SCHWARZ, MARPE e WIEGAND, 2007).

Para exemplificar, quando um macrobloco é codificado usando a predição de movimento entre camadas, se o bloco 8x8 na camada de referência não estiver divido em blocos menores, o macrobloco na camada de enriquecimento também não será particionado. Caso contrário, cada partição de tamanho M x N no bloco 8x8 da camada de referência corresponderá a uma partição de tamanho 2M x 2N na camada de enriquecimento. Os índices de referência serão os mesmo utilizados na camada de referência e ambos os componentes do vetor de movimento correspondente serão derivados através de uma escala com fator 2.

3.3.4.2 Predição Residual Entre Camadas (*Inter Layer Residual Prediction*)

A predição residual entre camadas pode ser aplicada para qualquer macrobloco que tenha sido codificado com predição inter, tendo sido usado o novo tipo de macrobloco da Predição de Movimento Entre Camadas ou os tipos de macroblocos convencionais.

Devido à predição de movimento entre camadas, camadas espaciais subseqüentes provavelmente terão informações de movimento similares e, assim, os resíduos de camadas consecutivas provavelmente terão uma forte correlação (SCHWARZ, MARPE e WIEGAND, 2007).

Para o uso deste tipo de predição, um *flag* é adicionado à sintaxe do macrobloco, o "*residual prediction flag*". Quando este *flag* é igual a 1, é realizada uma operação de *upsampling* no sinal residual do bloco 8x8 correspondente na camada de referência. Este *upsampling*, que será explicado em detalhes no próximo capítulo, usa um filtro bilinear e é realizado em nível de bloco, ou seja, o filtro não ultrapassa as bordas do bloco, o que poderia gerar alguns componentes de perturbação visual. Para permitir a aplicação do filtro, é realizada uma extensão de borda no bloco.

Este sinal gerado é usado como predição para o sinal residual da camada de enriquecimento e, portanto, somente a diferença correspondente é codificada.

3.3.4.3 Predição Intra Entre Camadas (*Inter Layer Intra Prediction*)

Quando um macrobloco na camada de enriquecimento é codificado com o "*base mode flag*" igual a 1 e o submacrobloco 8x8 correspondente na camada de referência foi codificado com predição intra quadro, o sinal de predição na camada de enriquecimento será obtido através do *upsampling* do sinal intra correspondente da camada de referência (SCHWARZ, MARPE e WIEGAND, 2007). Este processo de *upsampling* é o foco deste trabalho e é responsável pela sobreamostragem das informações de codificação presentes na camada de resolução mais baixa para a camada de resolução mais alta.

4 *UPSAMPLING* NO PADRÃO H.264/SVC E INVESTIGAÇÕES SOBRE O SOFTWARE DE REFERÊNCIA

Este capítulo apresenta as principais definições do *upsampling* entre camadas do padrão H.264 escalável. Estas definições são importantes para compreender a arquitetura desenvolvida e sua complexidade.

Este capítulo também mostra as investigações desenvolvidas sobre o software de referência do padrão escalável. Esta investigação foi importante para esclarecer algumas dúvidas relativas ao padrão e também porque, através dela, foi possível gerar os dados necessários para a validação da arquitetura desenvolvida.

4.1 *Upsampling* no Padrão H.264/SVC

O método *upsampling* não está identificado como um bloco explicitamente na estrutura de codificador escalável apresentado, no entanto, ele está presente entre as camadas dos codificadores e decodificadores escaláveis. O *upsampling* é aplicado quando o modo de predição de um bloco for do tipo entre camadas e o bloco correspondente na camada de referência tiver sido codificado através de predição intra-quadro. O *upsampling* também é utilizado na predição residual entre camadas (SCHWARZ, MARPE e WIEGAND, 2007).

Esta etapa da escalabilidade é responsável por adaptar a resolução dos blocos da camada de referência à resolução da camada de enriquecimento, para que os blocos possam ser usados como predição nesta camada. Por exemplo, cada bloco de tamanho M x N no bloco 8x8 da camada de referência corresponderá a uma partição de tamanho 2M x 2N na camada de enriquecimento.

Para implementar o *upsampling* é aplicado, aos componentes de luminância da imagem, um filtro polifásico de 4 *taps* por fase. Aos elementos de crominância, é aplicado um filtro bilinear. Os filtros são aplicados primeiro na horizontal e depois na vertical. O uso de filtros diferentes para luminância e para crominância é motivado por questões de complexidade. Nos padrões anteriores, a filtragem de *upsampling* era feita com filtros bilineares para ambos os componentes, resultando em uma redução na qualidade da predição (SCHWARZ, MARPE e WIEGAND, 2007).

No SVC, o *upsampling* é formado por um conjunto de 16 filtros, onde o filtro a ser aplicado é selecionado de acordo com o fator de escala do *upsampling* e com a posição da amostra. A escolha da fase correta do filtro a ser aplicada exige um procedimento padronizado para garantir que as decisões do codificador irão afetar a decodificação de maneira correta. Este procedimento é realizado através de um cálculo baseado na posição das amostras e da escala de *upsampling*, onde os bits menos significativos do resultado determinam o índice da fase do filtro a ser usado e os bits mais significativos determinam quais amostras serão filtradas (SEGALL e SULIVAN, 2007).

Os valores dos *taps* do filtro para luminância estão apresentados na Tab. 1. Na Tab. 2 estão os valores dos *taps* para o filtro de crominância. Após a aplicação dos valores apresentados nas tabelas, é necessário realizar um deslocamento de 5 posições para a direita para normalizar o resultado.

Os valores para o filtro bilinear de crominância, apresentados na Tab. 2, também são usados para realizar o *upsampling* dos componentes tanto de luminância quanto de crominância do sinal residual (SEGALL e SULLIVAN, 2007). A única diferença entre o filtro que é aplicado ao sinal da predição residual em relação ao aplicado ao sinal da predição intra está no fato de que o filtro não ultrapassa as bordas dos blocos no caso da predição residual, o que acontece para evitar a geração de componentes de distúrbio no sinal.

A filtragem para o sinal de predição intra sempre é realizada ultrapassando as bordas dos blocos e usa as amostras dos blocos intra vizinhos. Quando os blocos vizinhos não foram codificados com predição intra-quadro, as amostras necessárias são geradas através de extensão de borda. Esta restrição se justifica, pois, para usar as amostras dos blocos vizinhos codificados com predição inter quadros, será necessário realizar a compensação de movimento naquela camada e, quando múltiplas camadas espaciais são usadas, um processo pode ter a necessidade de invocar a compensação de movimento diversas vezes, o que resultaria em uma grande complexidade. A restrição apresentada é chamada de *Constrained Inter Layer Prediction* (Predição Restrita Entre Camadas), e ela reduz a complexidade do processo, pois, desta forma, a compensação de movimento é chamada apenas uma vez em cada camada. Este processo é chamado de *Single Loop Decoding* (Decodificação em Laço Único) (SCHWARZ, MARPE e WIEGAND, 2007).

Tabela 1 – Taps do Filtro para Luminância

Índice de Fase	***Taps* do Filtro**			
	E[-1]	**E[0]**	**E[1]**	**E[2]**
0	0	32	0	0
1	-1	32	2	-1
2	-2	31	4	-1
3	-3	30	6	-1
4	-3	28	8	-1
5	-4	26	11	-1
6	-4	24	14	-2
7	-3	22	16	-3
8	-3	19	19	-3
9	-3	16	22	-3
10	-2	14	24	-4
11	-1	11	26	-4
12	-1	8	28	-3
13	-1	6	30	-3
14	-1	4	31	-2
15	-1	2	32	-1

Para o caso diádico, em que a resolução dobra nas duas dimensões de uma camada para outra, são utilizados, alternadamente, os filtros de índices 4 e 12 (SEGALL E SULLIVAN, 2007), tanto na Tab. 1 quanto na Tab. 2.

Para os elementos de luminância, os filtros são representados pelas seguintes equações:

$$S_4 = (-3.A + 28.B + 8.C - D) >> 5 \qquad (1)$$

$$S_{12} = (-A + 8.B + 28.C - 3.D) >> 5 \qquad (2)$$

As equações (1) e (2) foram extraídas da Tab. 1 e representam os filtros de luminância de índices 4 e 12. Em ambas as equações, as variáveis A, B, C e D são as amostras de entrada do filtro, considerando a ordem do posicionamento das amostras da esquerda para a direita.

Tabela 2 – *Taps* do Filtro para Crominância

Índice de Fase	***Taps* do Filtro**			
	E[-1]	**E[0]**	**E[1]**	**E[2]**
0	0	32	0	0
1	0	30	2	0
2	0	28	4	0
3	0	26	6	0
4	0	24	8	0
5	0	22	10	0
6	0	20	12	0
7	0	18	14	0
8	0	16	16	0
9	0	14	18	0
10	0	12	20	0
11	0	10	22	0
12	0	8	24	0
13	0	6	26	0
14	0	4	28	0
15	0	2	30	0

Para as informações de crominância, os filtros são representados pelas equações (3) e (4). A equação (3) foi extraída da Tab. 2 e representa o filtro de crominância de índice 4. A equação (4) também foi extraída da Tab. 2 e mostra o filtro de crominância de índice 12. As variáveis A e B representam os pixels de entrada para o filtro.

$$S_4 = (24.A + 8.B) >> 5 \qquad (3)$$

$$S_{12} = (8.A + 24.B) >> 5 \qquad (4)$$

O funcionamento da filtragem de *upsampling* para o caso diádico está representado na Fig. 4. Os círculos pretos na Fig. 4 (a) são as amostras de uma imagem na camada de referência. O resultado da filtragem horizontal está representado na Fig. 4 (b), onde os círculos representam o posicionamento das amostras originais que foram substituídas pelas amostras geradas pela filtragem com o filtro de índice 12 e os quadrados pretos são as amostras intermediárias geradas pela filtragem com o filtro de índice 4. Ao final da filtragem horizontal, tem-se a imagem com o dobro de largura. A Fig. 4 (c) apresenta o resultado do processo

de filtragem vertical, onde os círculos e os quadrados cinza correspondem ao reposicionamento do resultado da filtragem horizontal que também foram substituídos pelas amostras geradas pelo filtro 12. Os triângulos pretos são as amostras geradas pelo filtro 4.

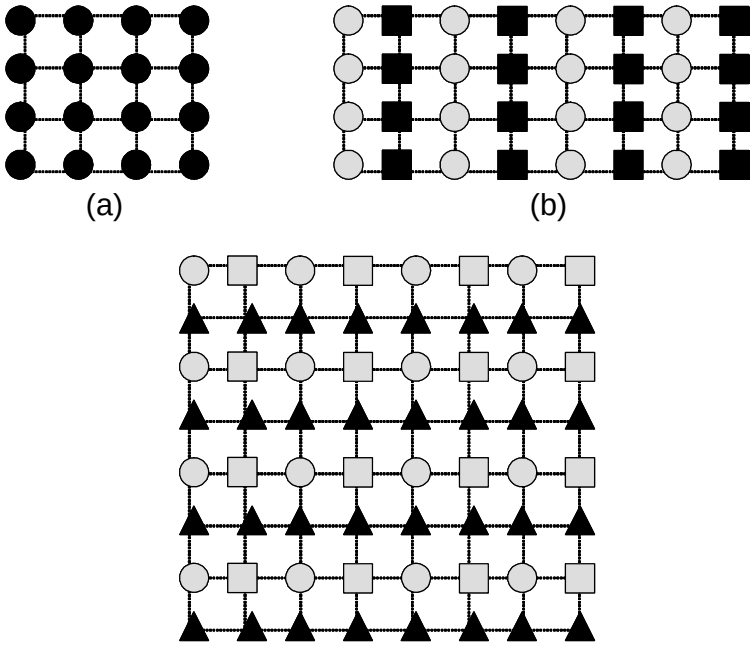


Figura 4 – *Upsampling* diádico: (a) imagem original, (b) imagem após filtragem horizontal e (c) imagem após filtragem vertical

4.2 Investigações no Software de Referência

Antes de passar diretamente à implementação da arquitetura em hardware, foi realizada uma investigação no software de referência, o JSVM – *Joint Scalable Video Model*, versão 9.12.2 (ITU-T e ISO/IEC, 2008), para verificar como o filtro é aplicado. Através do manual que acompanha a distribuição do software de referência, descobriu-se que o *upsampling* faz parte do arquivo Downconvert.cpp, onde estão implementados diversos métodos para a realização do *upsampling* e do *downsampling*. Desde o início dos estudos e da criação do padrão, diversas alternativas foram propostas tanto para o *upsampling* quanto para o *downsampling*. Estas alternativas estão disponíveis no software de referência ao se compilar a classe de forma independente, gerando o executável DownconvertStatic.exe. A classe Downconvert possui diretivas de compilação que permitem a sua compilação de forma independente, gerando o executável DownconvertStatic, ou então a compilação das rotinas necessárias para o codificidor/decodificador como um todo.

O funcionamento desta classe dentro do codificador/decodificador como um todo é diferente.

Após um estudo dos métodos através da extração e avaliação de alguns dados, foi identificado que o método *xBasicIntraUpsampling* é utilizado para realização do *upsampling* dentro do codificador/decodificador. Neste ponto, identificou-se que o método implementa a filtragem dos elementos de luminância com o conjunto de 16 filtros de 4 *taps* cada, conforme descrito no capítulo anterior deste trabalho, e não com um filtro único de 6 *taps*, como estava informado no manual da versão do software de referência. Isto aconteceu porque uma versão anterior do modelo definia que a filtragem seria feita com um filtro 6 *taps* e esta alteração foi realizada visando aumentar a eficiência da codificação. Para os elementos de crominância, é aplicado o conjunto de 16 filtros bilineares apresentados na Tab. 2.

Também foi possível avaliar, de uma forma mais ampla, como o software de referência trata a operação nos quadros e nos blocos do vídeo. O método *xBasicIntraUpsampling* recebe sua entrada de três diferentes formas: todas as informações de luminância do quadro, todas as informações de crominância azul ou todas as informações de crominância vermelha. Portanto, para o processamento completo de um quadro, são efetuadas, no total, seis chamadas ao método, pois, para cada um dos três casos mencionados, são realizadas duas chamadas, uma para a filtragem horizontal e outra para a filtragem vertical. Em cada chamada, o método processa sua entrada efetuando, primeiramente, a filtragem horizontal e, em seguida, a filtragem vertical sobre o resultado da filtragem anterior. Outra questão importante que foi observada foi a aplicação da operação de *clipping* e normalização somente após a conclusão da filtragem vertical, ou seja, quando o software de referência faz a filtragem horizontal, ele não faz a divisão por 32 presente na fórmula.

Em função destas constatações, neste trabalho decidiu-se por implementar a arquitetura seguindo esta mesma seqüência, pois apenas dessa forma seria possível ter dados reais para validar a arquitetura, através da extração de dados do software de referência.

Identificado o método que aplica o *upsampling*, foi possível extrair as entradas e saídas do mesmo para posterior comparação com os resultados gerados pela arquitetura definida neste trabalho.

A Fig. 5 apresenta um quadro de um vídeo extraído do software de referência na entrada do método *xBasicIntraUpsampling*. Este quadro está na resolução QVGA (320x240).

Figura 5 – Quadro extraído da entrada do método *xBasicIntraUpsampling*, na resolução QVGA

A Fig. 6 apresenta o quadro do vídeo extraído após a filtragem horizontal e desta forma, está com o dobro da resolução horizontal, ou seja, 640x240 pixels.

Figura 6 – Quadro extraído após a filtragem horizontal pelo método *xBasicIntraUpsampling*

A Fig. 7 está na resolução VGA (640x480) e foi extraído da saída do método, ou seja, este quadro foi gerado através do *upsampling* aplicado à Fig. 5. Na Fig. 8 está o quadro original do vídeo na resolução VGA, sem aplicação do *upsampling*.

Figura 7 – Quadro extraído da saída do método *xBasicIntraUpsampling*, na resolução VGA

Figura 8 – Quadro original na resolução VGA

5 ARQUITETURA DESENVOLVIDA PARA O UPSAMPLING

Após o estudo aprofundado dos conceitos de escalabilidade foi definido o bloco alvo da implementação desenvolvida neste trabalho. Então passou-se ao processo de definição da arquitetura para o método *upsampling*.

A arquitetura desenvolvida neste trabalho trabalha com o espaço de cores YCbCr e com subamostragem de cores 4:2:0, conforme explicado no capítulo 2. A resolução adotada para a camada base foi QVGA (320x240 pixels) e para a camada de enriquecimento VGA (640x480 pixels).

Os vídeos que foram usados tanto na extração de dados do software de referência quanto na validação da arquitetura, não foram gerados inicialmente na resolução VGA. Estes vídeos foram obtidos em TNT (2008), e estavam na resolução 4CIF (704x576 pixels). Para gerar o vídeo que pudesse ser usado de acordo com as definições adotadas para este trabalho, foi usado um executável presente no software de referência (ITU-T e ISO/IEC, 2008). Através deste executável foi gerado o vídeo na resolução VGA e a partir do vídeo na resolução VGA foi gerado o vídeo na resolução QVGA, para servir como entrada na camada base.

A Fig. 9 mostra a arquitetura para o *upsampling* completo proposta neste trabalho. Na Fig. 9 estão presentes os dois filtros de luminância, um horizontal (Filtro H Luma) outro vertical (Filtro V Luma) e os dois filtros de crominância, um horizontal (Filtro H Croma) outro vertical (Filtro V Croma). Também são apresentadas as memórias que funcionam como buffers de entrada para os filtros de luminância e crominância (MEM IN) e as memórias usadas como buffers de transposição ping-pong entre os filtros horizontais e verticais de luminância e crominância. Outro componente importante são os operadores de Clip na saída dos filtros, que são usados para colocar as saídas na mesma faixa dinâmica da entrada. Finalmente, o controle também está representado com linhas pontilhadas em vermelho. Foram omitidos os desenhos dos registradores de endereços de memória e seus respectivos multiplexadores, assim como os sinais de controle correspondentes. Esta omissão teve o objetivo de facilitar a visualização da arquitetura como um todo na figura. Cada um dos elementos da arquitetura serão explicados em maiores detalhes nas próximas seções.

Nesta arquitetura, os blocos operativos de luminância e de crominância não possuem dependências de dados e o processamento pode ser realizado em

paralelo. Os blocos de controle de luminância e de crominância foram projetados de forma independente, apenas com alguns sinais de sincronismo para sinalizar os pontos de início e fim de processamento de cada tipo de amostra.

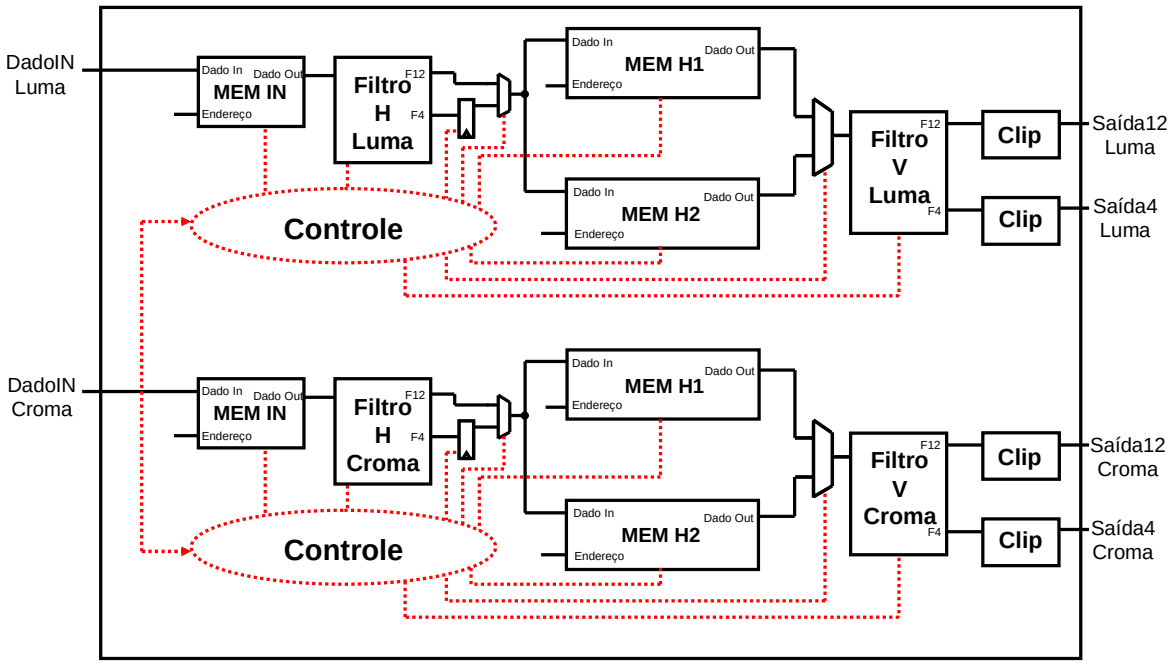


Figura 9 – Arquitetura completa do filtro de *upsampling*

5.1 Filtro de Luminância

5.1.1 Bloco Operativo de Luminância

A Fig. 10 apresenta o bloco operativo completo para a filtragem dos dados de luminância. De acordo com a definições adotadas neste trabalho, a memória que armazena o quadro de entrada, chamada de “MEM IN”, possui 76.800 palavras com largura de 8 bits. Já as memórias chamadas de “MEM H1” e “MEM H2”, que são as responsáveis por armazenar o resultado após a filtragem horizontal, possuem cada uma 153.600 palavras com largura de 15 bits. Optou-se pelo uso de duas memórias para armazenar o resultado intermediário, pois após o término da filtragem horizontal do primeiro quadro será iniciada a filtragem vertical deste quadro e, caso existisse apenas uma memória para este resultado intermediário, o filtro que faz a filtragem horizontal estaria ocioso até a conclusão da filtragem vertical. Utilizando duas memórias, enquanto é efetuada a filtragem vertical em um quadro, já é possível começar a filtragem horizontal no próximo quadro.

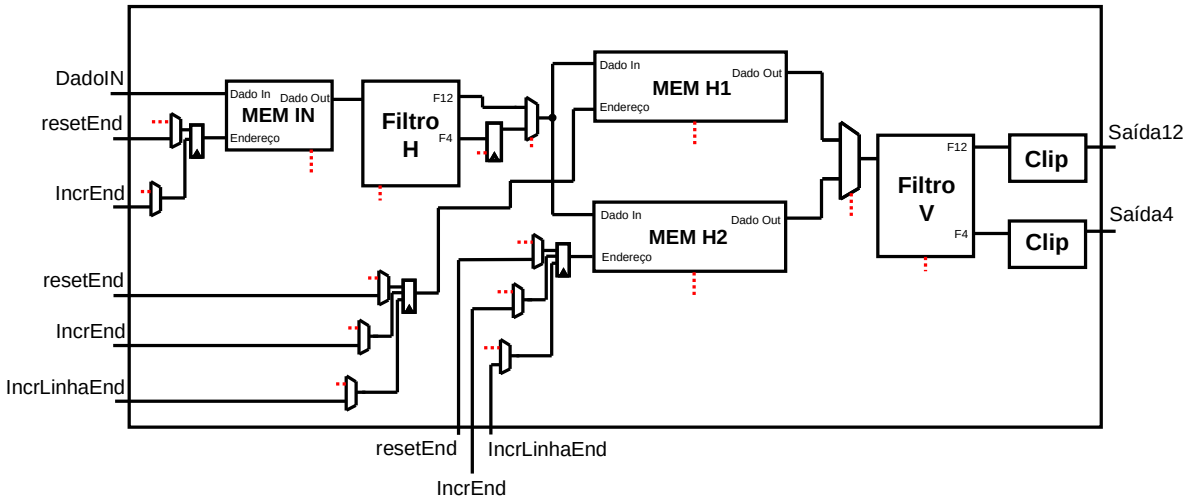


Figura 10 – Arquitetura completa do filtro de luminância

Para cada um dos módulos de memória presentes na Fig. 10 existe um registrador que armazena o endereço que será acessado. Estes registradores não possuem um dado de entrada, eles são do tipo registrador incrementador. Para a memória “MEM IN”, existe um sinal de *reset* e um sinal de incremento (que efetuará o incremento do valor armazenado). Também estão presentes mutiplexadores 2:1, pois ambos os sinais podem vir de fora da arquitetura para permitir a escrita dos dados de entrada nesta memória ou, então, do controle geral da arquitetura, quando estão sendo feitas as leituras para processamento. Para as outras memórias, além dos sinais já descritos, existe um sinal de incremento_de_linha que, ao invés de somar 1 ao valor atual do registrador, soma o valor correspondente ao deslocamento de uma linha do vídeo. Este incremento diferenciado é implementado porque para o processamento da filtragem vertical, será necessária a leitura de um elemento em cada linha da imagem intermediária, isto é, a leitura da imagem armazenada será feita coluna a coluna.

Também estão indicados, na Fig. 10, os filtros, onde o primeiro filtro é o filtro que aplica a filtragem horizontal e o segundo é o que aplica a filtragem vertical. A arquitetura interna destes componentes será apresentada em detalhes na próxima seção.

Os sinais indicados com linhas pontilhadas na Fig. 10 são os sinais provenientes do controle da arquitetura. Estes sinais também podem ser identificados por não serem entradas ou saídas da arquitetura e por não estarem conectados a outros componentes da arquitetura.

Além dos componentes já citados, foi projetado o componente *Clip*, que aparece nas saídas dos filtros da filtragem vertical. Somente após a conclusão da filtragem vertical é que o método aplica a operação de *clipping*. O *clipping* é efetuado através do deslocamento de 10 bits para a direita, o que representa os deslocamentos de 5 bits apresentados nas equações (1) e (2). Com isto, é possível ajustar o valor à faixa de valores das amostras em 8 bits. Neste trabalho, decidiu-se por implementar a arquitetura seguindo esta definição e não implementar diretamente as equações, apesar dessa forma necessitar de uma quantidade de memória bem superior para armazenar o resultado intermediário. Optou-se por ela, pois assim a arquitetura desenvolvida será 100% compatível com o software de referência e com o padrão SVC. Além disso, a validação fica facilitada, já que é possível extrair dados reais do software de referência. Foram realizados alguns testes comparativos em linguagem de programação C, que demonstraram que os resultados teriam erro de precisão, caso a divisão por 32 apresenta em (3) e (4) da próxima seção fosse aplicada imediatamente após a filtragem horizontal.

5.1.1.1 Núcleo do Filtro de Luminância

A filtragem das amostras de luminância é efetuada pela aplicação das equações apresentadas em (1) e (2). Para facilitar e otimizar a implementação em hardware, visando reduzir o consumo de recursos de hardware, essas equações foram manipuladas de forma a transformar as multiplicações, operação bastante custosa para implementação em hardware, em somas e deslocamentos. Como resultado das manipulações, foram obtidas as seguintes equações:

$$S_4 = (-2.A - A + 16.B + 8.B + 4.B + 8.C - D) >> 5 \qquad (5)$$

$$S_{12} = (-A + 8.B + 16.C + 8.C + 4.C - 2.D - D) >> 5 \qquad (6)$$

Através de manipulações algébricas na equação apresentada em (1) chegou-se à equação (5) e de manipulações semelhantes na equação (2), chegou-se à equação (6).

A Fig. 11 mostra a arquitetura definida neste trabalho para implementação da equação apresentada em (5). Nela, estão representados os deslocamentos e somas necessários e suas relações com as entradas A, B, C e D. Este filtro é formado por quatro somadores, três subtratores e cinco deslocadores.

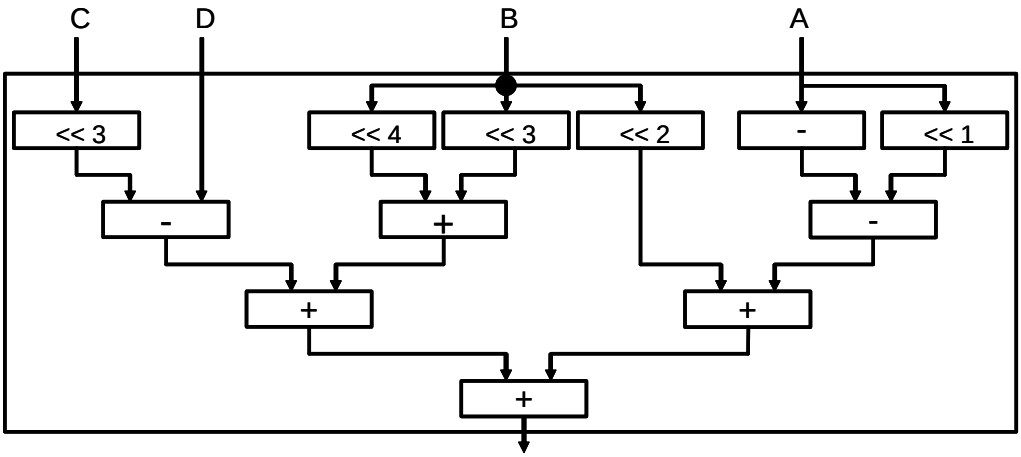


Figura 11 – Arquitetura interna do filtro interpolador de índice 4

Na Fig. 12 está representada a arquitetura do filtro de luminância de índice 12. Esta arquitetura foi gerada a partir da equação manipulada em (6) e possui uma estrutura semelhante à arquitetura do filtro 4.

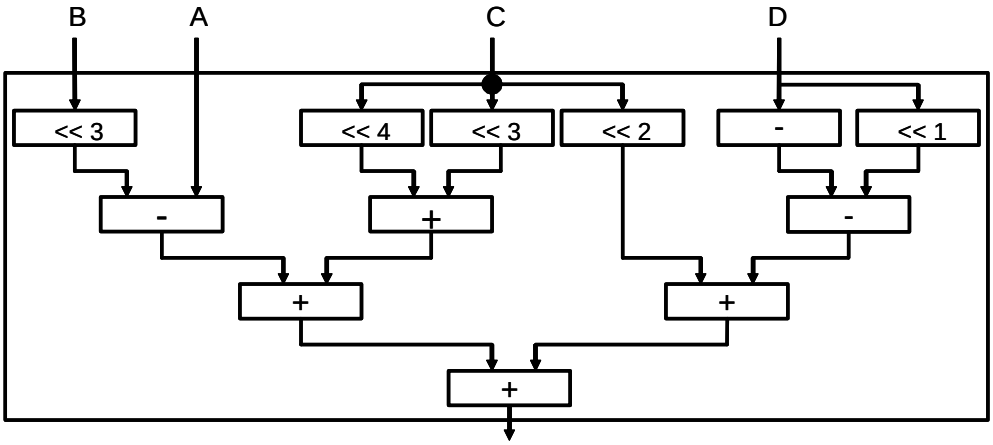


Figura 12 – Arquitetura interna do filtro interpolador de índice 12

Considerando-se que as entradas de A a D possuem uma largura de *n* bits e avaliando as operações efetuadas pela arquitetura, foi verificada a necessidade de aumento na largura de bits na saída com o objetivo de evitar o transbordo e, conseqüentemente, o erro no cálculo. Também foi feito um estudo em linguagem C, considerando todas as possibilidades de valores de entrada para se obter os maiores e menores valores resultantes. Desta forma, concluiu-se que a saída possuiria uma largura de *n*+7 bits, ou seja, resultando em um aumento de 7 bits na faixa dinâmica para cada aplicação do filtro.

Como foi verificada a possibilidade de aproveitamento dos dados de entrada entre os dois filtros, foi projetada a arquitetura presente na Fig. 13. Esta arquitetura

apresenta quatro registrados que estão ligados em seqüência, de forma que um dado de entrada fique num primeiro ciclo de relógio armazenado no registrador D e, no próximo ciclo, este dado passe para o registrador C, para dar espaço para uma nova entrada no registrador D, e a assim sucessivamente. A partir do momento que estão os quatro registradores preenchidos com dados válidos, os cálculos começam a ser efetuados pelos filtros.

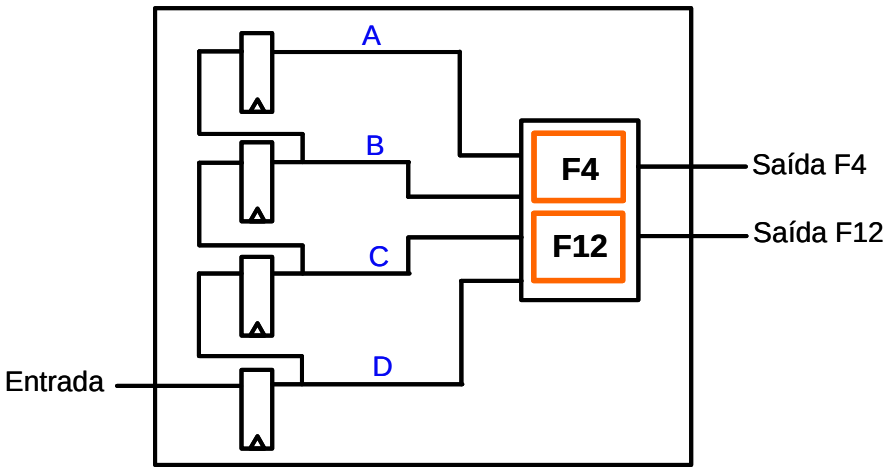


Figura 13 – Arquitetura do filtro de *upsampling* acoplando os filtros de índices 4 e 12

Sem considerar a primeira e a última amostras geradas na linha, no caso da filtragem horizontal, e coluna, no caso da filtragem vertical, foi constatado que os filtros usam as mesmas amostras e na mesma ordem para gerar as saídas. Isto é, no início do processamento de uma linha, quando os quatro registradores estiverem com dados válidos, o cálculo poderá ser executado, mas apenas o valor gerado pelo filtro 12 será armazenado como resultado válido. A partir do ciclo de relógio seguinte, um novo dado entrará no registrador D, e os dois resultados gerados pelos dois filtros serão considerados válidos, sendo que será salvo primeiro o filtro 4 e depois o filtro 12. Este processo se repete até o momento da geração da última amostra da linha ou coluna, onde será salvo apenas o resultado gerado pelo filtro 4.

5.1.2 Controle do Filtro de Luminância

Os sinais internos e sem conexão a outros componentes da arquitetura na Fig. 10 são os sinais de controle. A seguir serão descritos os sinais de controle usados na arquitetura proposta.

Para o registrador que armazena o endereço de acesso à memória “MEM IN”, existem os sinais de *reset* e incremento, bem como os sinais de seleção dos

multiplexadores. Através destes sinais é realizado o controle de quando será feita a escrita do quadro de entrada na memória e quando serão feitas as leituras para processamento. Também estão identificados os sinais de ativação da escrita nos módulos de memória *(write enable*). Para as memórias “MEM H1” e “MEM H2” existem os sinais de *reset*, incremento e incremento de linha, com os respectivos sinais de seleção dos multiplexadores. Além destes, também estão o sinal de carga no registrador que armazena o resultado do filtro 4 na primeira etapa da filtragem, o sinal de seleção do multiplexador que escolhe entre os resultados dos dois filtros qual será escrito na memória e o sinal de seleção do multiplexador que seleciona de qual dos módulos de memória, “MEM H1” ou “MEM H2”, os dados serão lidos para o processamento da filtragem vertical.

O controle da arquitetura completa do filtro de luminância foi projetado de acordo com a máquina de estados finitos apresentada na Fig. 14. Esta máquina de estados está representada por macroestados para facilitar a sua visualização e o entendimento do processo. No total, foram utilizados 13 estados para completar o processamento. Pela Fig. 14 é possível ter uma visão geral deste funcionamento. No estado 0, estado inicial, são realizadas as inicializações necessárias para começar o processamento. No estado 1, são realizadas as inicializações necessárias ao início de processamento de um novo quadro, como, por exemplo, *reset* nos registradores de endereço de memória. No estado 2 são realizadas as primeiras leituras de uma nova linha do vídeo para a filtragem horizontal, ou coluna, quando se trata da filtragem vertical. Para gerar as primeiras amostras da linha ou coluna será necessária a realização de uma extensão de borda, pois nem todas as amostras que precisam estar disponíveis ao filtro existem. Estas leituras iniciais realizadas neste estado caracterizam essa extensão de borda. No estado 3, o cálculo propriamente dito é realizado pelo filtro. Após este estado, chega-se ao estado 4, que dá início ao laço de processamento até chegar ao fim da linha ou coluna. Este laço é caracterizado por uma nova leitura na memória de entrada do filtro, cálculo e correspondente escrita na memória de resultado, quando for o caso. Ao final do processamento de uma linha ou coluna, também é necessária a realização da extensão de borda e isto é realizado na passagem do estado 4 para o estado 2. Ainda, do estado 4, pode-se partir para o estado 1, quando a linha ou coluna processada é a última do quadro. Quando for atingido o final de todo processamento, o estado seguinte é o estado 0.

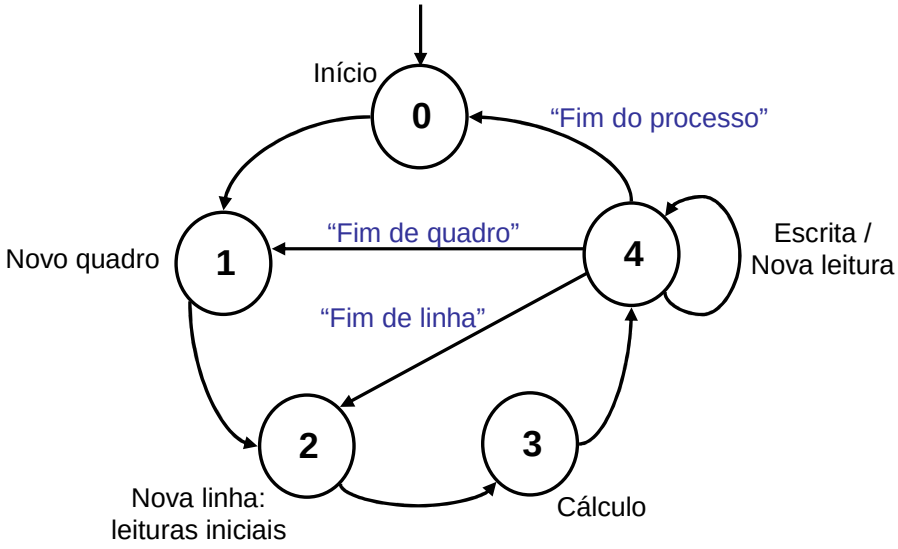


Figura 14 – Controle da arquitetura completa do filtro de luminância

Como a quantidade de ciclos que são usados para o processamento horizontal e vertical é diferente, o processamento vertical exige praticamente o dobro de ciclos do horizontal. Para permitir o controle dos dois filtros sem aumentar de forma exagerada o número de estados, foi implementada uma segunda máquina de estados, com o mesmo funcionamento da apresentada na Fig. 14. A primeira máquina de estados controla a filtragem horizontal e a segunda a filtragem vertical. As duas máquinas de estados se comunicam para que se possa controlar quando não deve ser realizada a filtragem horizontal ou vertical. A filtragem vertical não é realizada apenas enquanto não for concluída a filtragem horizontal do primeiro quadro. Já a filtragem horizontal não é realizada no intervalo entre o término da filtragem horizontal de um quadro e a conclusão da escrita do novo quadro de entrada e no intervalo entre o fim do processamento deste quadro e o término da filtragem vertical. Por exemplo, após a conclusão da filtragem horizontal do primeiro quadro, a “MEM H1” possui estes resultados armazenados, sendo possível, então, o início do processamento vertical deste quadro. Enquanto isso, a memória “MEM H2” está livre para o início do processamento horizontal do segundo quadro. Antes disto, porém, é preciso escrever o novo quadro de entrada na “MEM IN” e durante esta escrita, o processamento horizontal não pode ser realizado. Após a escrita do novo quadro de entrada, inicia-se o processamento horizontal deste quadro. Durante todo o processo de escrita do novo quadro de entrada e de filtragem horizontal deste novo quadro, o processamento da filtragem vertical do quadro anterior está sendo

executado em paralelo. Existe mais um período de tempo entre o término desta filtragem horizontal e da filtragem vertical em que o filtro horizontal ficará parado. Portanto, mesmo com inserção de um novo bloco de memória para armazenar o resultado intermediário, para permitir o processamento em paralelo, ainda existem dois momentos em que parte do hardware fica ociosa. Através dos cálculos apresentados a seguir, é possível perceber esta proporção. Para o cálculo do número de ciclos necessários às filtragens foi considerado o número dos ciclos para a geração da primeira amostra da linha ou coluna, ou seja, 4 ciclos. O mesmo número usado para geração da última amostra. Para gerar cada uma das demais amostras, é necessário 1 ciclo. Este processo é repetido para cada linha, no caso da filtragem horizontal, ou coluna, no caso da filtragem vertical.

- **Filtragem vertical:** (4+(480-2)+4)*640 = 486*640 = 311.040 ciclos
- **Escrita do quadro de entrada:** (320*240) = 76.800 ciclos
- **Filtragem horizontal:** (4+(640-2)+4)*240 = 646*240 = 155.040 ciclos
- **Escrita + filtragem horizontal:** 231.840 ciclos

Comparando a quantidade de ciclos necessários para cada uma das execuções que podem ser realizadas em paralelo, percebe-se que o filtro horizontal fica ocioso por 79.200 ciclos. No entanto, após a conclusão do processamento completo do primeiro quadro, um novo quadro é concluído a cada 311.040 ciclos, ou seja, o limitante é o processamento vertical. Caso não fosse utilizado o segundo bloco de memória para o resultado intermediário, um novo quadro só poderia ser gerado a cada 542.880 ciclos, ou seja, o correspondente a todo o processamento. Além disto, o filtro horizontal ficaria ocioso durante todo o processamento vertical mais o tempo de escrita do novo quadro de entrada, em um total 387.840 ciclos.

A Fig. 15 apresenta um diagrama de tempo que mostra a sequência de processamento das filtragens horizontal e vertical e também das escritas dos novos quadros de entrada. Em “t0” inicia-se o processamento do primeiro quadro. Este quadro é escrito na memória de entrada e, em seguida, é feita a filtragem horizontal. Em “t1”, ou seja, em t0 + 231.840 ciclos, é iniciado o processamento vertical do primeiro quadro e, ao mesmo tempo, inicia-se a escrita do novo quadro de entrada e processamento horizontal deste novo quadro. No tempo “t2”, isto é, “t1” + 231.840 ciclos, é concluída a filtragem horizontal do segundo quadro e em “t3” (“t1” + 311.040 ciclos) é concluída a filtragem vertical do primeiro quadro. Como os dados do

segundo quadro já estão prontos, pode-se iniciar a filtragem vertical deste segundo quadro.

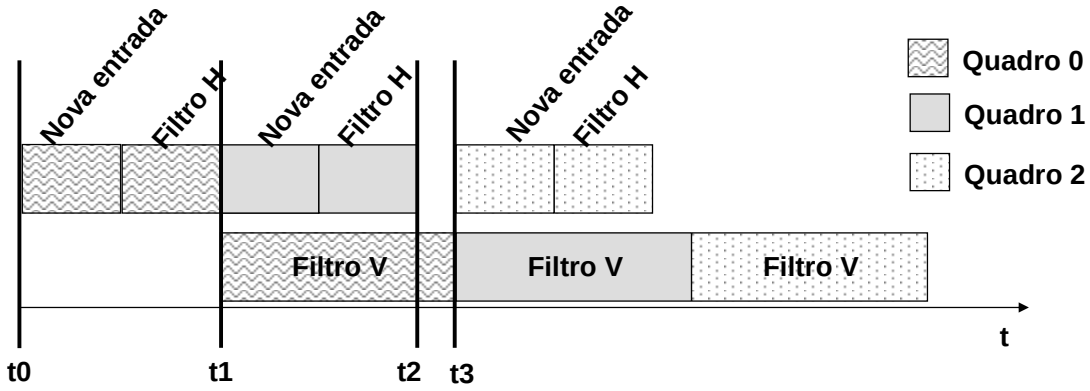


Figura 15 – Diagrama de tempo para filtragem horizontal e vertical

5.2 Filtro de Crominância

5.2.1 Bloco Operativo de Crominância

O bloco operativo do filtro de crominância apresenta a mesma estrutura da arquitetura completa do filtro de luminância, mostrada na Fig. 13. As diferenças entre as arquiteturas estão nos filtros utilizados e nos tamanhos das memórias. Os filtros usam o núcleo apresentado na Fig. 18. A memória “MEM IN” de crominância possui 19.200 palavras com largura de 8 bits e as memórias “MEM H1” e “MEM H2” possuem 38.400 palavras com largura de 14 bits.

5.2.1.1 Núcleo do Filtro de Crominância

A arquitetura para o núcleo do filtro de crominância foi definida com base na Tab. 2. Para o caso diádico, são usados os filtros de índices 4 e 12 que são representados pelas equações apresentadas em (3) e (4).

Por meio de manipulações semelhantes às aplicadas às equações de luminância, chega-se às equações (7) e (8).

$$S_4 = (16.A + 8.A + 8.B) >> 5 \qquad (7)$$

$$S_{12} = (8.A + 16.B + 8.B) >> 5 \qquad (8)$$

A arquitetura que implementa o filtro de crominância de índice 4 mostrado na equação (7) está apresentada na Fig. 16. Este filtro é formado por três deslocadores

e dois somadores. Já a Fig. 17 apresenta a arquitetura do filtro de crominância de índice 12, que resultou da equação (8). Esta arquitetura é também constituída de três deslocadores e dois somadores.

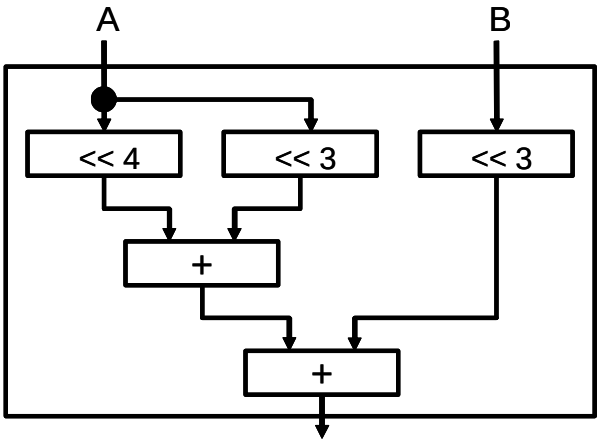


Figura 16 – Filtro de crominância de índice 4

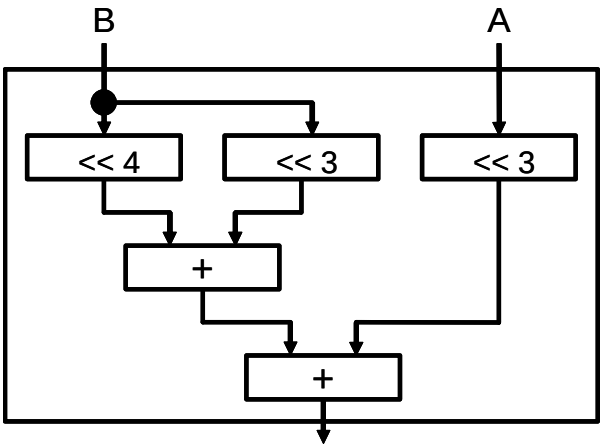


Figura 17 – Filtro de crominância de índice 12

Na Fig. 18 está apresentada a arquitetura correspondente ao núcleo completo do filtro de crominância. O projeto desta arquitetura foi feito baseado na arquitetura que foi desenvolvida para luminância, tendo um funcionamento bem semelhante. Além dos dois filtros apresentados na Fig. 16 e na Fig. 17, ela possui dois registradores ligados em seqüência, que armazenam as amostras que servirão para os cálculos efetuados pelos filtros. Considerando que a entrada possui uma largura de *n* bits, as saídas corresponderão a *n+6* bits.

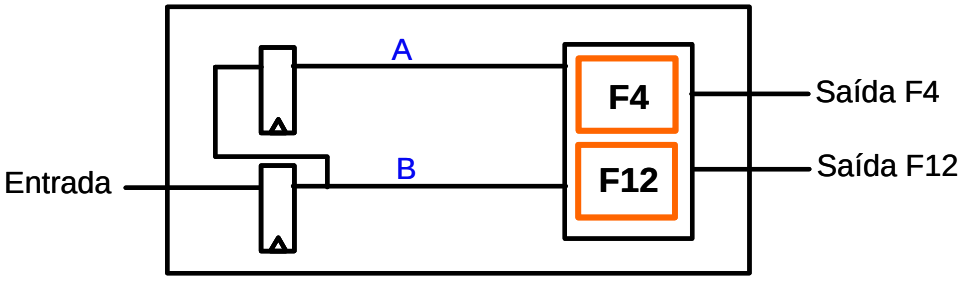


Figura 18 – Núcleo do filtro de crominância

5.2.2 Controle do Filtro de Crominância

A máquina de estados do controle do filtro de crominância também segue a mesma estrutura do controle do filtro de luminância, apresentado na Fig. 14.

Neste projeto, a arquitetura completa do filtro de crominância possui capacidade para processar apenas um dos tipos de crominâncias (Cb ou Cr) por vez. Esta decisão foi tomada para que não fosse necessário replicar o hardware para este processamento. Porém, torna-se requisito para esta arquitetura que a mesma seja capaz de processar as crominâncias Cb e Cr em seqüência durante o tempo em que as amostras de luminância são processadas.

Para o processamento das amostras de crominância Cb, é realizada a escrita do quadro Cb de entrada, em seguida, é realizada a filtragem horizontal e, posteriormente, a filtragem vertical. No momento em que está concluída a filtragem horizontal da crominância Cb, pode-se dar início ao processamento das amostras de crominância Cr. A escrita do quadro de entrada e a filtragem horizontal das amostras de Cr são realizadas em paralelo com a filtragem vertical de Cb. Somente após a conclusão da filtragem de Cb é que é possível começar a filtragem vertical das amostras de Cr. Desta forma, o tempo necessário à conclusão de toda a filtragem de crominância é constituído pela escrita do quadro de entrada, pela filtragem horizontal e por duas vezes a filtragem vertical.

De acordo com os cálculos apresentados abaixo, o tempo necessário para todo o processamento das amostras de crominância é de 212.720 ciclos. Tempo este inferior ao necessário para toda a filtragem de luminância, que é de 311.040 ciclos. Para cálculo do número de ciclos necessários às filtragens foi considerado o número do ciclos para a geração da primeira amostra da linha ou coluna, ou seja, 2 ciclos. O mesmo número usado para geração da última amostra. Para gerar cada

uma das demais amostras, é necessário 1 ciclo. Este processo é repetido para cada linha, no caso da filtragem horizontal, ou coluna, no caso da filtragem vertical.

- **Filtragem vertical:** (2+(240-2)+2)*320 = 242*320 = 77.440 ciclos
- **Escrita do quadro de entrada:** (160*120) = 19.200 ciclos
- **Filtragem horizontal:** (2+(320-2)+2)*120 = 322*120 = 38.640 ciclos

Em função dos dados apresentados, é possível concluir que a implementação de apenas um bloco operativo para processar as crominâncias Cb e Cr em seqüência é viável, já que não afeta o desempenho do filtro de *upsampling* completo pois o caminho crítico deste módulo segue residindo no módulo que realiza a filtragem de luminância. Além disso, se duas arquiteturas independentes fossem desenvolvidas para crominância, ambos os blocos operativos ficariam um tempo considerável ociosos.

5.3 Validação das Arquiteturas

As arquiteturas apresentadas nas seções anteriores foram descritas em linguagem de descrição de hardware VHDL (Ashenden, 1998). As arquitetura para os filtros completos de luminância e de crominância foram projetadas, implementadas e completamente validadas. No entanto, a arquitetura do upsampling completo foi projetada e implementada, mas sua validação não foi concluída.

Em um primeiro momento foi implementado o núcleo do filtro de luminância. Logo em seguida, os demais blocos envolvidos na filtragem dos elementos de luminância, isto é, os blocos de memória, registradores comuns e incrementadores e os multiplexadores. Cada um destes blocos, ao ser descrito em VHDL, foi sintetizado utilizando a ferramenta Quartus II (ALTERA, 2008) e testado através das formas de onda do próprio Quartus II, para que fosse avaliado se o funcionamento estava de acordo com o esperado. Nesta avaliação, buscou-se testar alguns casos médios e os casos extremos. No caso do filtro, por exemplo, foram testados alguns valores reais, extraídos do software de referência, e os valores que geravam os resultados com maiores e menores amplitudes. Estes valores de entrada foram extraídos de um programa desenvolvido em linguagem C, que testava todos os valores possíveis dentro da faixa de valores de 8 bits e, então, mostrava os resultados com maior e menor amplitude que poderiam ser gerados pelo filtro.

Posteriormente, passou-se à implementação do filtro de luminância completo. No entanto, como o funcionamento dos dois núcleos dos filtros (horizontal e vertical) é idêntico e os componentes relacionados são bem semelhantes, optou-se por implementar apenas a estrutura envolvida ao redor do primeiro filtro, o filtro H e então validar esta arquitetura. Assim, com esta estrutura validada, seria facilitada a implementação e validação do restante da arquitetura, consistindo esta da inclusão do segundo bloco de memória intermediária, do Filtro V e do Clipping. Esta escolha se justifica por reduzir a área investigada, facilitando a busca por eventuais erros nas descrições.

A arquitetura descrita inicialmente foi a que está apresentada na Fig. 20. Para sua validação foi desenvolvido um arquivo de *test bench*. O arquivo de *test bench* é uma descrição em VHDL que instancia a arquitetura alvo e é responsável pela inserção de estímulos de entrada e pela leitura dos resultados desta arquitetura. O processo de validação foi realizado através do software ModelSim (MENTOR, 2008) em sua versão ModelSim-Altera 6.1g.

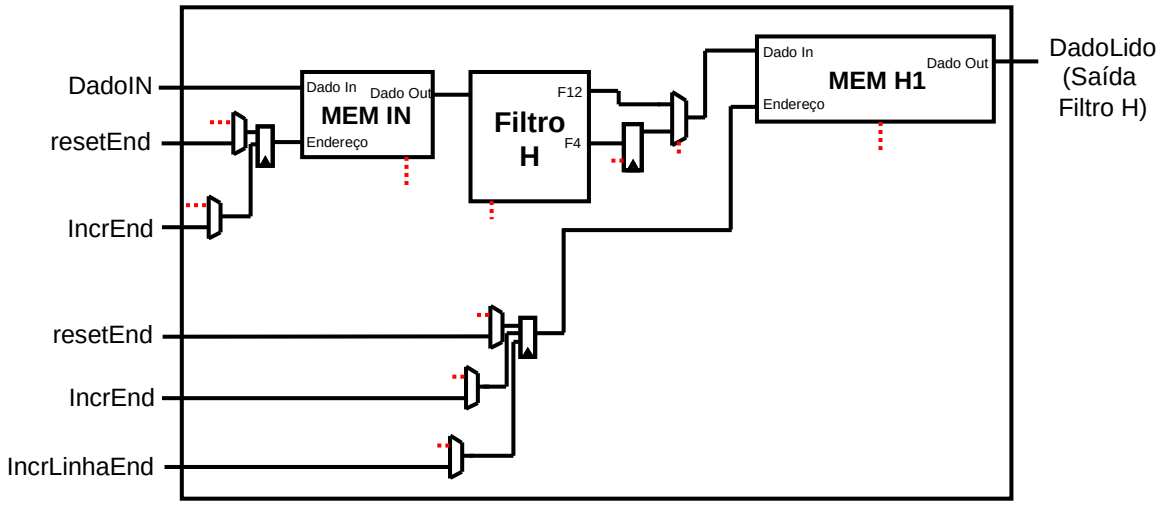


Figura 20 – Estrutura para validação inicial

Os estímulos de entrada utilizados pelo arquivo de *test bench* foram extraídos do software de referência. Nesta validação inicial, foram extraídos os quadros de entrada e os dados resultantes após a aplicação da filtragem horizontal.

O *test bench* lê primeiramente os dados do quadro de entrada que foram extraídos do software de referência e escreve estes dados na memória “MEM IN”. Durante este procedimento, os sinais de controle para o registrador de endereço de

memória são entradas externas à arquitetura e são gerados pelo *test bench*. Assim que a escrita é concluída, os sinais de controle do registrador de memória passam para o controle da arquitetura e o *test bench* ativa o sinal de início da arquitetura, que faz com que o processamento comece. Durante todo o processamento, o *test bench* não tem mais nenhum controle sobre o que está acontecendo e só resta a ele aguardar pelo sinal da arquitetura que indica que a filtragem horizontal do quadro foi finalizada. Para finalizar, o *test bench* deve ler os dados que estão escritos na memória “MEM H1” e escrevê-los em arquivo para que possa ser feita a comparação com os resultados gerados pelo software de referência.

Os dados foram comparados e após alguns ajustes na descrição da arquitetura proposta, chegou-se aos resultados esperados.

O próximo passo então seria a descrição e validação da arquitetura completa. Para a descrição da arquitetura completa, foram incluídos o segundo módulo de memória intermediária, “MEM H2”; o registrador de endereço e sinais de controle correspondentes; o filtro vertical e os componentes de clipping.

O controle implementado para a arquitetura que foi inicialmente validada, o filtro horizontal, controla as leituras efetuadas na memória de entrada, os momentos em que deverão ser realizadas as extensões de bordas e os momentos em que os dados válidos foram calculados pelo filtro e, então, faz a escrita destes dados na memória intermediária. Para o restante da arquitetura, ou seja, o filtro vertical, o controle funciona da mesma forma, os dados são lidos da memória intermediária, controlando-se os momentos em que deve ser realizada a extensão de borda e, posteriormente, a escrita na saída. Então decidiu-se implementar uma segunda máquina de estados finitos como um segundo processo no VHDL dentro do mesmo módulo de controle, mantendo-se uma comunicação entre as duas máquinas para se identificar o início e o fim do processamento de cada uma delas.

Para a validação desta arquitetura completa, também foi projetado um arquivo de *test bench* que escreve os quadros de entrada na memória “MEM IN” e escreve os dados de saída da arquitetura, os quais são obtidos após a aplicação do clipping. Como, neste ponto, já se sabe que os dados que estarão escritos na memória intermediária estão corretos, a área de busca por eventuais problemas fica reduzida. Este *test bench* foi desenvolvido com alguns processos que funcionam em paralelo, para permitir a simulação da forma como a arquitetura foi projetada para funcionar em hardware, implementando todos os paralelismos projetados.

Um processo escreve os quadros de entrada na memória de entrada, outro dispara o processamento do filtro horizontal, outro dispara o processamento do filtro vertical e outro processo faz a retirada dos dados de saída da arquitetura. Todos estes processos devem manter comunicação para permitir o perfeito sincronismo.

Comparando-se os dados gerados pela arquitetura que foram extraídos pelo *test bench* com os dados extraídos do software de referência, foi possível verificar que a arquitetura respondeu como esperado.

As arquiteturas da parte de crominância também foram descritas em VHDL, assim como as de luminância.

A descrição VHDL do núcleo do filtro foi realizada de forma semelhante à utilizada para descrever o filtro de luminância. O núcleo foi testado inicialmente dentro da ferramenta Quartus II, através das formas de onda. Após esta avaliação inicial do funcionamento, os blocos que formam a arquitetura de crominância foram conectados para validação da arquitetura completa.

A arquitetura de crominância foi desenvolvida seguindo a especificação de Segall e Sullivan (2007). Neste artigo, está descrito que, do mesmo modo que para as informações de luminância, é aplicado um conjunto de 16 filtros. Além disso, também está definido que, para o caso abordado neste trabalho, o caso diádico, os filtros utilizados são os filtros de índices 4 e 12 alternadamente. No entanto, ao se extrair os dados do software de referência para a realização da validação, foram constatadas diferenças entre os valores esperados e os gerados. Uma investigação no software de referência mostrou que, para crominância, os filtros que estão sendo utilizados são os filtros de índices 2 e 10.

Como o software de referência está em constante desenvolvimento e em diversos aspectos permite o uso de ferramentas não padronizadas pelo SVC e como o artigo de Segall e Sullivan (2007) está em desacordo com os resultados do software de referência no que diz respeito à filtragem de crominância, optou-se, neste trabalho, por considerar as informações presentes em Segall e Sullivan (2007) para implementar a arquitetura do filtro de crominância, por considerar este artigo mais confiável, já que é escrito pelo coordenador do JVT (Gary Sullivan). Além disso, as definições do artigo de Segall e Sullivan (2007) sobre a filtragem das amostras de luminância para o caso diádico estão 100% de acordo com o software de referência, reforçando a confiabilidade das informações que ali estão apresentadas.

Desta forma, não foi possível validar a arquitetura com dados do software de referência. Então, foram gerados alguns dados para validação da arquitetura através de um software implementado em C, que efetua o mesmo processamento executado pelo software de referência, apenas fixando-se os valores dos índices dos filtros que serão utilizados.

Para o processo de validação, também foi desenvolvido um *test bench* que utiliza as informações geradas pelo software descrito acima para injetar os sinais de entrada e retirar os dados gerados. O processo de validação foi realizado com o software ModelSim. Os dados gerados pela arquitetura foram comparados com os dados gerados pelo software e observou-se que a arquitetura gerou o resultado esperado.

Cabe salientar que, caso seja identificado, futuramente, que o procedimento descrito em Segall e Sullivan (2007) para processamento das informações de crominância para o caso diádico não esteja de acordo com o proposto pela norma, pequenas alterações arquiteturais no núcleo do filtro de crominância serão suficientes para deixá-la adaptada à norma.

6 RESULTADOS

Neste capítulo serão apresentados os resultados de síntese que foram obtidos para as arquiteturas que foram implementadas. As arquiteturas foram sintetizadas para FPGAs da Altera (ALTERA, 2008) através da ferramenta Quartus II também da Altera.

6.1 Resultados de Síntese

As arquiteturas apresentadas para os núcleos dos filtros de luminância e crominância foram sintetizados para FPGAs das famílias Stratix II e Stratix IV (GX/E).

Na Tab. 3 estão os resultados obtidos para dispositivo EP2S15F484C3 da família Stratix II. O núcleo do filtro de luminância atingiu a freqüência de 154,66 MHz e o de crominância atingiu 414,08 MHz.

Tabela 3 – Resultados de síntese dos núcleos dos filtros para Stratix II

	Luminância	**Crominância**
Freqüência (MHz)	154,66	414,08
ALUTs	154	42
Registradores Lógicos Dedicados	66	40

Família Stratix II, dispositivo: EP2S15F484C3

Também foram gerados os resultados de síntese dos núcleos para um FPGA da família Stratix IV (GX/E). Como não foi possível sintetizar a arquitetura completa do *upsampling* (luminância+crominância) para nenhum dispositivo da família Stratix II, estes dados foram gerados para permitir a comparação entre o núcleo e a arquitetura completa. Os resultados de síntese para o Stratix IV estão apresentados na Tab. 4.

A ferramenta Quartus II ainda não consegue gerar resultados de análise de *timing* totalmente confiáveis para os dispositivos da família Stratix IV, pois os modelos de *timing* ainda estão em desenvolvimento. O primeiro resultado de freqüência apresentado na Tab. 4 provém do modelo chamado “Slow 900mV 85C

Model” e o segundo resultado do modelo chamado “Slow 900mV 0C Model”. Para o primeiro modelo, os resultados de freqüência foram inferiores aos encontrados para o dispositivos Stratix II.

Tabela 4 – Resultados de síntese dos núcleos dos filtros para Stratix IV (GX/E)

	Luminância	Crominância
Freqüência Modelo 1 (MHz)	151,42	381,68
Freqüência Modelo 2 (MHz)	161,39	406,01
ALUTs	154	42
Registradores Lógicos Dedicados	66	40

Família Stratix IV, dispositivo: EP45SGX530HH35C3

Na Tab.5 estão os resultados de síntese para as arquiteturas completas tanto de luminância quanto de crominância. Esta síntese foi direcionada para um FPGA da família Stratix IV. Conforme mencionado anteriormente, não foi possível realizar a síntese da arquitetura completa de luminância para nenhum dispositivo da família Stratix II e esta impossibilidade foi por causa da memória interna do FPGA que a arquitetura necessita para ser sintetizada. Apesar da arquitetura usar um total de bits de memória inferior às capacidades dos dispositivos, a memória interna do FPGA está organizada em blocos e a forma como a memória foi implementada na arquitetura fez com que a ferramenta de síntese necessitasse de mais blocos de memória do que os blocos disponíveis em um dispositivo Stratix II.

Tabela 5 – Resultados de síntese das arquiteturas completas para Stratix IV (GX/E)

	Luminância	Crominância	Upsampling Completo
Freqüência Modelo 1 (MHz)	137,61	190,99	119,5
Freqüência Modelo 2 (MHz)	146,76	202,59	127,42
ALUTs	1.267	759	2.024
Registradores Lógicos Dedicados	577	454	1032
Bits de Memória	5.222.400	1.288.800	6.451.200

Família Stratix IV, dispositivo: EP45SGX530HH35C3

Para a implementação da arquitetura completa de crominância, seria possível a síntese para um dispositivo da família Stratix II. No entanto, para melhor comparação entre as arquiteturas, esta foi sintetizada para o mesmo dispositivo da família Stratix IV para o qual foi implementada a arquitetura de luminância.

6.2 Taxas de Processamento

Após realizada a síntese das arquitetura e conhecendo a quantidade de ciclos necessários para que as arquiteturas consigam efetuar o processamento das amostras, foi possível calcular a taxa de processamento das arquiteturas desenvolvidas. Nas tabelas a seguir estão descritos os resultados para as arquiteturas desenvolvidas, onde a resolução da camada de enriquecimento é VGA (640x480). Também foi estimado o desempenho da arquitetura, considerando-se as mesmas freqüências de operação, para as resoluções de 4VGA (1280x960) e 16VGA (2560x1920) na camada de enriquecimento. Todas as estimativas consideram a existência de apenas duas camadas espaciais e a resolução citada é a resolução da camada de enriquecimento. A camada base destas estimativas respeita o caso diádico.

As Tabs. 6 e 7 apresentam a taxa de processamento, em quadros por segundo, considerando apenas os núcleos dos filtros. Este cálculo considera a capacidade do filtro de gerar duas amostras a cada ciclo, excetuando-se as amostras das bordas dos quadros. Para o filtro de crominância, o cálculo já considera que serão processadas as amostras de crominância Cb e, na seqüência, as amostras de crominância Cr.

Na Tab. 6 estão os dados gerados para os núcleos dos filtros através dos resultados obtidos com a síntese para o dispositivo Stratix II. Considerando que a etapa de processamento das amostras de luminância é a etapa limitante da freqüência de operação do filtro completo, a arquitetura desenvolvida consegue gerar 978 quadros VGA por segundo. Na estimativa para 16VGA, pode se chegar a 62 quadros por segundo.

Tabela 6 – Taxa de processamento dos núcleos dos filtros para Stratix II

	Luminância	**Crominância**
Quadros VGA	978	5.260
Quadros 4VGA	*248	*1.331
Quadros 16VGA	*62	*334

**estimativa*

A Tab. 7 apresenta os resultados obtidos para os núcleos dos filtros para o dispositivo Stratix IV (GX/E). Estão representados os cálculos para os dois modelos

de *timing* que a ferramenta utilizou para gerar os resultados de freqüência. Neste dispositivo, a arquitetura desenvolvida para o filtro de luminância processa até 957 quadros VGA por segundo no pior caso e para a estimativa com a resolução 16VGA a arquitetura é capaz de processar 61 quadros por segundo.

Tabela 7 – Taxa de processamento dos núcleos dos filtros para Stratix IV (GX/E)

	Slow 900mV 85C Model		**Slow 900mV 0C Model**	
	Luminância	**Crominância**	**Luminância**	**Crominância**
Quadros VGA	957	4.848	1.020	5.157
Quadros 4VGA	*242	*1.227	*258	*1.305
Quadros 16VGA	*61	*308	*65	*328

**estimativa*

Nas Tab. 8 e Tab. 9 estão representados os resultados gerados para as arquiteturas completas dos filtros de luminância e crominância, com a síntese direcionada para o dispositivo Stratix IV (GX/E). Na Tab. 8 estão os resultados que foram gerados pelo primeiro modelo de *timing* e na Tab. 9 estão os resultados para o segundo modelo de *timing*. Foi calculada a taxa de processamento para a arquitetura desenvolvida na resolução VGA e, também, foram estimadas as taxas de processamento para as resoluções maiores. A arquitetura do filtro completo de luminância desenvolvida neste trabalho tem o pior desempenho e mesmo assim atinge, no pior caso, uma taxa de processamento de 442 quadros VGA por segundo e na estimativa para resolução 16VGA chega-se a 27 quadros por segundo. Considerando a arquitetura completa que engloba o filtro de luminância e de crominância é possível chegar a uma taxa de 384 quadros por segundo para a arquitetura proposta neste trabalho. Estimando a taxa de processamento para a resolução 16VGA, chega-se a 24 quadros por segundo.

Tabela 8 – Taxa de processamento para as arquiteturas completas para o primeiro modelo de timing do Stratix IV (GX/E)

	Slow 900mV 85C Model		
	Luminância	**Crominância**	***Upsampling* Completo**
Quadros VGA	442	1.233	384
Quadros 4VGA	*111	*309	*96
Quadros 16VGA	*27	*77	*24

Tabela 9 – Taxa de processamento para as arquiteturas completas para o segundo modelos de timing do Stratix IV (GX/E)

	Slow 900mV 0C Model		
	Luminância	**Crominância**	***Upsampling* Completo**
Quadros VGA	471	1.308	409
Quadros 4VGA	*118	*328	*103
Quadros 16VGA	*29	*82	*25

Nas arquiteturas definidas e implementadas neste trabalho, em que são utilizadas duas camadas espaciais (QVGA para a camada base e VGA na camada de enriquecimento) conseguiu-se uma taxa de processamento que atinge tempo real.

Para as estimativas realizadas para as resoluções maiores, em que as resoluções citadas são as resoluções das camadas de enriquecimento, apenas para para a síntese da arquitetura completa do filtro de luminância e para o *upsampling* completo, chegou-se a um processamento inferior a 30 quadros por segundo. No entanto, em nenhum caso a taxa estimada foi inferior a 24 quadros por segundo que é a taxa mínima exigida para tempo real.

7 CONCLUSÕES

Neste trabalho de conclusão de curso foi realizado um estudo sobre a codificação de vídeo escalável com foco na extensão escalável do padrão H.264/AVC. Após a identificação dos mecanismos necessários à implementação da escalabilidade, passou-se a um estudo aprofundado do bloco escolhido como alvo deste trabalho, o *upsampling*. O estudo do *upsampling* partiu de artigos recentes que apresentam o seu funcionamento chegando a detalhes de implementação investigados no software de referência.

O bloco de *upsampling* é utilizado na região entre camadas na escalabilidade espacial. Neste trabalho, foi definido que o filtro a ser desenvolvido trabalharia no contexto de um codificador/decodificador com duas camadas espaciais diádicas, cujas resoluções seriam QVGA para a camada base e VGA para a camada de enriquecimento.

O passo seguinte deste trabalho foi a definição e projeto de uma arquitetura para implementar a filtragem de *upsampling*. Inicialmente foram definidas as arquiteturas para os núcleos dos filtros de luminância e crominância. Posteriormente, foi projeta a arquitetura com toda a estrutura de memórias e demais componentes necessário ao processamento completo da filtragem. Esta implementação baseou-se na seqüência de processamento do software de referência.

As arquiteturas foram descritas em VHDL e sintetizadas para dispositivos FPGAs das famílias Stratix II e Stratix IV (GX/E) da Altera. A idéia inicial foi direcionar a síntese para um Stratix II, no entanto, a quantidade de blocos de memória interna do dispositivo não foi suficiente para acomodar os blocos de memória da arquitetura. Por isso, a síntese da arquitetura completa teve que ser direcionada para um dispositivo Stratix IV (GX/E), mesmo que os resultados de timing que são gerados pela ferramenta Quartus II para estes dispositivos ainda não sejam definitivos.

As arquiteturas desenvolvidas foram validadas através da ferramenta ModelSim. Os dados que serviram de base para a validação da arquitetura de luminância foram retirados do software de referência. Para a validação da

arquitetura de crominância, os dados retirados do software de referência não estavam de acordo com a descrição do funcionamento de Segall e Sullivan (2007) e, então, os dados foram gerados por um software desenvolvido para tal fim.

Entre os resultados de síntese obtidos para a arquitetura completa do *upsampling* chegou-se a uma freqüência de operação entre 119,5 MHz e 127,42 MHz para os dois modelos de timing que são utilizados pela ferramenta para a síntese direcionada para dispositivos Stratix IV (GX/E). Com estas freqüências, foi possível atingir taxas de processamento entre 384 e 409 quadros por segundo.

Também foi realizada uma estimativa de taxas de processamento para resoluções maiores, sempre respeitando o uso de duas camadas espaciais e o caso diádico. Para a resolução da camada de enriquecimento 4VGA, a taxa de processamento ficou entre 96 e 103 quadros por segundo para dos dois modelos de timing. Para a resolução de 16VGA, a taxa ficou entre 24 e 25 quadros por segundo.

Não foi possível comparar este trabalho com outros trabalhos publicados na literatura, pois não foram encontrados outros trabalhos sobre o desenvolvimento em hardware do filtro de *upsampling* da extensão escalável do padrão H.264/AVC. Deste modo, este trabalho tem um importante grau de ineditismo.

Como trabalhos futuros estão previstos a validação da arquitetura completa do *upsampling* proposta neste trabalho e a prototipação desta arquitetura. Também como trabalho futuro está a busca de alternativas mais eficientes para o uso e gerenciamento de memória, para tentar evitar problemas como o encontrado neste trabalho em relação à síntese em alguns dispositivos.

Outras investigações futuras são a implementação do *upsampling* em mais de duas camadas espaciais e a implementação do *upsampling* para os casos não diádicos.

Referências

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. Tese de Doutorado–Universidade Federal do Rio Grande do Sul. II. PPGC, Porto Alegre, Brasil-RS, 2007.

Altera Corporation. “Altera: The Programmable Solutions Company”. Disponível em: www.altera.com. Acesso em: agosto, 2008.

Ashenden, Peter. **The Student’s Guide to VHDL**. San Francisco: Morgan Kaufmann, 1998.

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

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

Gonzalez, R.; Woods, R. **Processamento de Imagens Digitais**. São Paulo: Edgard Blücher, 2003.

Huang, Hsiang-Chun; Peng, Wen-Hsiao; Chiang, Tihao; Hang, Hsueh-Ming. **Advances in the Scalable Amendment of H.264/AVC**. Communications Magazine, IEEE. Vol. 45, No. 1, pp. 68-76, Jan. 2007.

ITU-T e ISO/IEC JTC1. “**Joint Scalable Video Model JSVM-9.12.2**”. Abril, 2008.

JVT Editors (T. Wiegand, G. Sullivan, A. Luthra), **Draft ITU-T Recommendation and final draft international standard of joint video specification** (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC), 2003.

JVT Editors (T. Wiegand, G. Sullivan, A. Luthra), **Draft ITU-T Recommendation and final draft international standard of joint video specification** (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC), 2007.

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

Richardson, I. E. G. **H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia**, John Wiley & Sons Publishers, USA, 2003.

Schwarz, H.; Marpe, D.; Wiegand, T. **Overview of the Scalable Video Coding Extension of the H.264/AVC Standard**. IEEE Transactions on Circuits and

Systems for Video Technology, Vol. 17, No. 9, pp. 1103-1120, Sept. 2007, invited paper.

Segall, C.A.; Sullivan, G.J. **Spatial Scalability Within the H.264/AVC Scalable Video Coding Extension.** IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp. 1121-1135. Set. 2007.

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

TNT - Institut fuer Informationsverarbeitung - Institute of Signal Processing, Universitaet Hannover. "**Test Sequences**". Disponível em: ftp.tnt.uni-hannover.de/pub/svc/testsequences/. Acesso em: junho, 2008.

Wiegand, T.; Sullivan, G.; Reichel, J.; Schwarz, H.; Wien, M. **Joint Draft 11 of SVC Amendment**, Joint Video Team, Doc. JVT-X201, Jul. 2007.

Wien, M.; Schwarz, H.; Oelbaum, T. **Performance Analysis of SVC**. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp.1194-1203, Sept. 2007, invited paper.

APÊNDICE A – Publicações e prêmios durante a graduação

A.1 Prêmios recebidos durante a graduação

1º Lugar na Área de Ciência Exatas e da Terra, XVI Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2007.

Prêmio Jovem Pesquisador 1º Lugar na Área de Engenharias, XV Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2006.

A.2 Trabalhos publicados durante a graduação

A.2.1 Artigos completos publicados em periódicos

ROSA, Leandro; PORTO, Marcelo; REDIESS, Fabiane; PETRY, Rafael; SUSIN, Altamiro; BAMPI, Sergio, AGOSTINI, Luciano. **Avaliação Algorítmica para a Estimação de Movimento na Compressão de Vídeos Digitais**. Hífen, Uruguaiana, 2007.

MILANI, Cleber; REDIESS, Fabiane; VORTMANN, César; PATZER, Gabriel; MATTOS, Marcelo; OLIVEIRA, Lucas Ferrari de. **Estudo do Desenvolvimento Multiplataforma de uma Ferramenta para Processamento e Visualização Interativa de Imagens de Ressonância Magnética**. Logos (Canoas), 2007.

REDIESS, Fabiane; SILVA, André; VORTMANN, João; GUNTZEL, José; BAMPI, Sergio; AGOSTINI, Luciano. **Projeto de Hardware para a Compensação de Movimento do Padrão H.264/AVC de Compressão de Vídeo**. Hífen (Uruguaiana), v. 30, p. 1-8, 2006.

SILVA, Thaísa; VORTMANN, João; REDIESS, Fabiane; GUNTZEL, José; BAMPI, Sergio; AGOSTINI, Luciano. **Codificador de Entropia Segundo o Perfil Baseline do padrão H.264/AVC de Compressão de Vídeo**. Hífen (Uruguaiana), v. 30, p. 1-8, 2006.

A.2.2 Trabalhos completos em anais de eventos

SAMPAIO, Felipe; REDIESS, Fabiane; FONSECA, Carolina Marques; SUSIN, Altamiro; BAMPI, Sérgio; AGOSTINI, Luciano. **Architectural Design for Forward and Inverse Transforms of H.264/AVC Standard Focusing in the Intra Frame Coder**. In: VLSI-SoC 2008 - The 16th IFIP/IEEE International Conference on Very

Large Scale Integration, 2008, Ilha de Rhodes. VLSI-SoC 2008 - The 16th IFIP/IEEE International Conference on Very Large Scale Integration. Piscataway: IEEE, 2008.

VORTMANN, João; PETRY, Rafael; CORRÊA, Guilherme; REDIESS, Fabiane; AGOSTINI, Luciano; CAVALHEIRO, Gerson. **Estimação de Movimento com Multiprogramação Leve**. In: WSCAD-SSC 2008 - Workshop em Sistemas Computacionais de Alto Desempenho, 2008, Campo Grande. WSCAD-SSC 2008.

CORRÊA, Guilherme; REDIESS, Fabiane; AGOSTINI, Luciano; CAVALHEIRO, Gerson. **Aplicação de Técnicas de Processamento Paralelo no Algoritmo Full Search de Estimação de Movimento**. In: 8ª Escola Regional de Alto Desempenho (ERAD 2008), 2008, Santa Cruz do Sul. Anais da 8ª Escola Regional de Alto Desempenho. Porto Alegre, 2008. p. 205-208.

SAMPAIO, Felipe; FONSECA, Carolina; REDIESS, Fabiane; BAMPI, Sérgio; AGOSTINI, Luciano. **Desenvolvimento de uma Arquitetura Paralela para a Hadamard 2-D 2x2 do Padrão H.264/AVC**. In: Iberchip 2008 - XIV Workshop IBERCHIP, 2008, Puebla - México.

REDIESS, Fabiane; SAMPAIO, Felipe Martin; FONSECA, Carolina; BAMPI, Sérgio; SUSIN, Altamiro; AGOSTINI, Luciano. **Architectural Design for Forward and Inverse 4x4 Transforms of H.264/AVC Standard Focusing in the Intra Frame Coder**. In: SIM 2008 - XXIII Simpósio Sul de Microeletrônica, 2008, Bento Gonçalves. XXIII Simpósio Sul de Microeletrônica. Porto Alegre, 2008. p. 123-126.

SAMPAIO, Felipe; REDIESS, Fabiane; FONSECA, Carolina; BAMPI, Sérgio; SUSIN, Altamiro; AGOSTINI, Luciano. **Architecture Design and Prototyping of a Fully Parallel H.264/AVC 2x2 Hadamard Transform Targeting Real Time in Very High Resolution Videos**. In: SIM 2008 - XXIII Simpósio Sul de Microeletrônica, 2008, Bento Gonçalves. XXIII Simpósio Sul de Microeletrônica. Porto Alegre, 2008. p. 131-134.

PORTO, Marcelo; BAMPI, Sergio; SUSIN, Altamiro; ROSA, Leandro; REDIESS, Fabiane; PETRY, Rafael; AGOSTINI, Luciano. **Investigation of Motion Estimation Algorithms Targeting High Resolution Digital Vídeo Compression**. XIII Brazilian Symposium on Multimedia and the Web. Gramado. 2007

ROSA, Leandro; REDIESS, Fabiane; PETRY, Rafael; PORTO, Marcelo; AGOSTINI, Luciano; BAMPI, Sérgio. **Motion Estimation Algorithms Evaluation**. In: XXII SIMPÓSIO SUL DE MICROELETRÔNICA, 2007, Porto Alegre. 2007.

REDIESS, Fabiane; GUNTZEL, José; BAMPI, Sergio; AGOSTINI, Luciano. **Desenvolvimento Arquitetural Para a Compensação de Movimento do Padrão H.264/AVC**. In: XIII Taller IBERCHIP, 2007, Lima. XIII Taller IBERCHIP, 2007.

REDIESS, Fabiane; SILVA, André; GUNTZEL, José; AGOSTINI, Luciano. **A Motion Compensation Architecture for Video Compression**. In: XXI SIM - South Symposium on Microeletronics, 2006, Porto Alegre. South Symposium on Microeletronics. Porto Alegre : SBC, 2006. p. 65-68.

A.2.3 Resumos expandidos em anais de eventos

REDIESS, Fabiane; GUNTZEL, José. **Projeto e Implementação de Processadores RISC em VHDL**. In: XIV CIC - Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2005, Pelotas.

A.2.4 Resumos simples em anais de eventos

REDIESS, Fabiane; AGOSTINI, Luciano. **Projeto de Hardware para o Upsampling da Codificação de Vídeo Escalável do Padrão H.264/AVC**. In: XVII Congresso de Iniciação Científica – UFPel, 2008, Pelotas.

REDIESS, Fabiane; CORRÊA, Guilherme; AGOSTINI, Luciano. **Desenvolvimento de Hardware para o Filtro Redutor de Efeito de Bloco Inter-Camadas do Padrão H.264 Escalável de Compressão de Vídeo com Foco nas Futuras Gerações do Sistema Brasileiro de TV Digital**. In: XVII Congresso de Iniciação Científica – UFPel, 2008, Pelotas.

SAMPAIO, Felipe; FONSECA, Carolina; REDIESS, Fabiane; AGOSTINI, Luciano. **Desenvolvimento de um Protótipo em FPGAs Altera da Arquitetura Serial da Transformada Hadamard 2-D 2x2 do Padrão H.264/AVC com Foco em Vídeos de Alta Resolução**. In: CIC 2007 - XVI Congresso de Iniciação Científica - UFPel, 2007, Pelotas.

REDIESS, Fabiane; PETRY, Rafael; ROSA, Leandro; AGOSTINI, Luciano. **Avaliação de Algoritmos para a Estimação de Movimento com Blocos de 8x8 Pixels na Compressão de Vídeos Digitais com Foco no Padrão Brasileiro de Televisão Digital**. In: XVI Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2007, Pelotas.

PETRY, Rafael; REDIESS, Fabiane; ROSA, Leandro; AGOSTINI, Luciano. **Estimação de Movimento na Compressão de Vídeos Digitais: uma Análise**

Quantitativa Sobre os Impactos do Uso de Blocos com 4x4 Pixels In: XVI Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2007, Pelotas.

ROSA, Leandro; REDIESS, Fabiane; PETRY, Rafael; AGOSTINI, Luciano. **Uma Análise Quantitativa de Algoritmos para a Estimação de Movimento com Diversos Tamanhos de Bloco com Foco em Vídeos Digitais de Alta Resolução**. In: XVI Congresso de Iniciação Científica da Universidade Federal de Pelotas, 2007, Pelotas.

CORREA, Marcelo; FRANCK, Helen; GOMES, Bianca; REDIESS, Fabiane; GUNTZEL, José; OLIVEIRA, Lucas. **Implementação em Hardware do Algoritmo Histograma 2D: Funções Média X e Média Y**. In: XV Congresso de Iniciação Científica da UFPel, 2006, Pelotas.

REDIESS, Fabiane; GUNTZEL, José; AGOSTINI, Luciano. **Desenvolvimento de Hardware para a Compensação de Movimento do Padrão H.264/AVC com Foco na Compressão de Vídeo do Sistema Brasileiro de Televisão Digital**. In: XV Congresso de Iniciação Científica da UFPel, 2006, Pelotas.