UNIVERSIDADE FEDERAL DE PELOTAS

Bacharelado em Ciência da Computação

Trabalho de Conclusão de Curso

Filtro Redutor do Efeito de Bloco Intercamadas para o Padrão H.264/SVC Realizado com Dispositivo FPGA

Guilherme Ribeiro Corrêa

Pelotas, 2009

GUILHERME RIBEIRO CORRÊA

FILTRO REDUTOR DO EFEITO DE BLOCO INTERCAMADAS PARA O PADRÃO H.264/SVC REALIZADO COM DISPOSITIVO FPGA

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 (Universidade Federal de Pelotas)
Co-orientador: Prof. Dr. Luís A. da Silva Cruz (Universidade de Coimbra, Portugal)

Pelotas, 2009

Banca examinadora:

_____

_____

_____

_____

AGRADECIMENTOS

Antes de tudo, agradeço aos meus pais, Élida e Vitório, todo o amor, empenho, dedicação e trabalho ao longo dos últimos vinte e um anos para que eu e minha irmã pudéssemos crescer com saúde e educação. Agradeço também por terem me dado um grande presente chamado Amanda, a pessoa que eu mais amo no mundo.

Aos meus avós, Edith e Chico, agradeço o carinho e o incentivo que recebi durante a minha vida acadêmica. À minha avó Sali, agradeço igualmente o carinho e todas as lições de história, de geografia, de política, de música e de vida que me foram ensinadas.

Aos meus grandes amigos Stuart, Filipe, Dany e Paula, agradeço pelos jantares que me ajudam, há mais de sete anos, a descansar depois de árduas semanas de estudo. Obrigado pela amizade de vocês e por serem sempre divertidos e felizes.

Ao meu amigo Elvio, obrigado por estar sempre disposto a ir ao NYIP depois (ou ao invés) de estudar, por me obrigar a fazer exercícios físicos e por me ouvir reclamar da vida quando um semestre ou um namoro está chegando ao fim.

Ao meu amigo e ex-professor Güntzel, meu primeiro orientador, agradeço a oportunidade que me apresentou em 2005, quando fui convidado a ingressar no GACI. Obrigado pelas lições de responsabilidade, dedicação e, obviamente, microeletrônica, que recebi durante a graduação. Obrigado também por ser um grande exemplo para mim.

Ao professor Luciano Agostini, meu segundo orientador, agradeço o empenho que teve em garantir o crescimento do GACI e de todos gacistas. Obrigado por todos os conselhos (sempre úteis e práticos) que me foram dados e pela orientação.

Obrigado a todos os gacistas pelo tempo que trabalhamos juntos em um ambiente super agradável. Obrigado a todos os que um dia prepararam o café no GACI. Obrigado à Fabiane, à Carol, à Thaísa e ao Felipe pelas discussões que antecederam este trabalho.

Agradeço ao professor Luís Cruz, meu orientador em Portugal, pela ótima hospitalidade no Instituto de Telecomunicações e por sempre ser extremamente atencioso e dedicado à minha orientação.

Aos amigos que fiz em Portugal, agradeço pelo companheirismo e por todos os momentos que passamos juntos durante os seis meses em Coimbra. Agradeço, especialmente, ao Gregory por estar sempre disposto a me ouvir, incentivar e aconselhar nos momentos mais difíceis do intercâmbio. As suas palavras otimistas foram, sem dúvida, essenciais para a conclusão deste trabalho.

Agradeço, por fim, à UFPel, ao CNPq e ao Santander, que fomentaram as bolsas de trabalho, iniciação científica e mobilidade internacional que recebi durante a gradução. Obrigado também aos tios Márcio, Valério e Fafá, que complementaram o fomento do Santander, e a todos aqueles que sempre torceram por mim.

Resumo

CORRÊA, Guilherme Ribeiro. **Filtro Redutor do Efeito de Bloco Intercamadas para o Padrão H.264/SVC Realizado com Dispositivo FPGA.** 2009. 70f. Monografia – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas.

Com o aumento do número de dispositivos capazes de manipular vídeos digitais, surge um problema relacionado à transmissão e à representação de tais vídeos em equipamentos com características distintas, como resolução e poder de armazenamento. Uma solução para este problema seria a implementação de um decodificador capaz de redimensionar o vídeo para as capacidades do dispositivo. Esta solução, contudo, poderia aumentar consideravelmente o custo do dispositivo e o seu consumo de energia. Outra solução sugere a transmissão de duas ou mais codificações do mesmo vídeo, uma para cada tipo de dispositivo. Isto, entretanto, acarretaria um grande desperdício de banda, visto que a mesma informação seria transmitida mais de uma vez. A escalabilidade é a proposta que mais se adequa à solução do problema de transmissão, já que apenas um *bitstream* é gerado por vídeo e cada dispositivo pode selecionar apenas a informação do seu interesse. Para que isso seja possível, uma estrutura de camadas de codificação/decodificação é utilizada. Assim, a partir das informações de uma camada base e adicionando-se informações de camadas de enriquecimento, é possível um aumento na resolução espacial, temporal ou de qualidade do vídeo.

No padrão H.264/AVC, a escalabilidade é suportada através da extensão H.264/SVC – *Scalable Vídeo Coding*. O foco deste trabalho está no Filtro Redutor de Efeito de Bloco Intercamadas, um módulo utilizado pela Predição Intra Intercamadas do H.264/SVC, quando a escalabilidade utilizada é do tipo espacial. O funcionamento do filtro é similar ao do Filtro Redutor de Efeito de Bloco proposto no H.264/AVC, que tem por finalidade suavizar artefatos em forma de blocos inseridos à imagem devido a um elevado passo de quantização durante a codificação.

Este trabalho propõe uma nova ordem de filtragem, diferente da sugerida pelo padrão H.264/AVC, mas que, ainda assim, obedece as suas restrições. Propõe-se uma filtragem em nível de amostras, ao invés de blocos, o que possibilita uma

melhor exploração do paralelismo da arquitetura. Desta forma, a arquitetura desenvolvida é capaz de realizar quatro filtragens concorrentemente, diminuindo significativamente o número de ciclos utilizados para a filtragem de um macrobloco completo, quando comparada com trabalhos equivalentes.
As arquiteturas foram descritas em VHDL e sintetizadas para dispositivo FPGA da família Stratix III da Altera. Os resultados da análise de *timing* mostraram que a arquitetura é capaz de processar até 623 quadros por segundo para uma resolução alta (HDTV – 1920x1080 pixels), um resultado que satisfaz com folga os requerimentos impostos pelo padrão H.264/AVC (24 a 30 quadros por segundo).

Palavras-chave: Padrão H.264/AVC, H.264/SVC, Escalabilidade, Filtro Redutor de Efeito de Bloco.

Abstract

CORRÊA, Guilherme Ribeiro. **Interlayer Deblocking Filter for H.264/SVC Extension Implemented in FPGA.** 2009. 70f. Monograph – Curso de Bacharelado em Ciência da Computação. Universidade Federal de Pelotas.

With the increase of the number of devices able to handle digital videos, a problem related to the transmission and the displaying of these videos by devices with different characteristics has arisen. The first thought solution for this problem was the implementation of decoders which could resample the videos for the device's capacity. However, this solution can increase significantly the device's cost and power consumption. Another solution suggests the transmission of two or more codes for the same video, one for each type of device. Even so, it would require a large waste of frequency range, since the same information would be transmitted more than once. This way, the scalability is the solution which fits better to the transmission problem, since just one bitstream is generated for each video and each device selects just the information of its interest. To make it possible, a layer-based structure for coding and decoding is used. Thus, from the information of a base layer and by adding information of enhancement layers, it is possible to increase the spatial, temporal or quality resolution of the video.

In the H.264/AVC standard, the scalability is provided by the H.264/SVC – Scalable Video Coding – extension. The focus of this project is an Interlayer Deblocking Filter, a module used by the Interlayer Intra Prediction of H.264/SVC when the scalability type used is spatial. The operation of this filter is similar to the Deblocking Filter proposed in the H.264/AVC standard, which is used to smooth block-shaped artifacts inserted in the image due to a high quantization step during the coding process.

This work proposes a new filtering order, different from the one suggested in the H.264/AVC, but, even so, following its requirements. A filtering order in sample level is proposed, instead of block level, enabling a better exploration of the architecture's parallelism. This way, the developed architecture can perform four concurrent filterings, decreasing significantly the number of cycles used for the filtering of a complete macroblock.

The architecture was described in VHDL and synthesized for an Altera's Stratix III FPGA device. The timing analysis results showed that the architecture is able to

process until 623 frames per second with a high resolution (HDTV – 1920x1080 pixels). This result satisfies and is over the requirements imposed by the H.264/AVC (from 24 to 30 frames per second).

Keywords: H.264/AVC standard, H.264/SVC, scalability, Deblocking Filter.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

bS	*Boundary Strength*
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*
LOP	*Line of Pixels*
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*
RGB	*Red, Green, Blue*

SBTVD	Sistema Brasileiro de Televisão Digital
SDTV	*Standard Definition Television*
SNR	*Signal-to-noise Ratio*
SVC	*Scalable Video Coding*
VHDL	*VHSIC Hardware Description Language*

SUMÁRIO

1 INTRODUÇÃO

Para que seja possível a manipulação de um vídeo digital em dispositivos eletrônicos, o mesmo precisa passar por um processo que reduza a quantidade de bits utilizados na sua representação. Tal processo, chamado de compressão de vídeo, vem sendo amplamente investigado na indústria e na academia ao longo dos últimos anos. Diversos padrões foram desenvolvidos, mas o H.264/AVC (JVT, 2003) é considerado o estado-da-arte em compressão de vídeo, pois atingiu o seu objetivo de dobrar a taxa de compressão com relação aos padrões anteriores. Contudo, para atingir tal objetivo, a complexidade do codificador H.264/AVC é muito maior que a de padrões anteriores, como o MPEG-2 e o MPEG-4 Parte 2 (RICHARDSON, 2003).

Com o crescente avanço na tecnologia de codificação de vídeo e com o rápido desenvolvimento de infra-estruturas de rede, armazenamento e poder computacional, o número e a variedade de dispositivos que suportam aplicações de vídeo aumentaram significativamente. Desta forma, o processo de decodificação de cada dispositivo, especialmente os de menores capacidades, pode se mostrar bastante oneroso em alguns aspectos como consumo de energia, já que estes equipamentos precisariam implementar funções de adaptação do vídeo transmitido para a sua capacidade de visualização (SEGALL e SULLIVAN, 2007). Por exemplo, não seria justificável projetar um dispositivo de baixa resolução e com a capacidade de decodificar e subamostrar vídeos de alta resolução. Além de aumentar o custo da arquitetura, isto também aumentaria consideravelmente o consumo de energia e a potência dissipada pelo dispositivo ao ponto de exceder os limites que determinam a resolução do seu *display*. Além disso, o envio de detalhes de alta resolução não exibidos pelo *display* do dispositivo representaria um desperdício do canal de envio de dados.

A fim de solucionar este problema, pesquisadores têm desenvolvido padrões com suporte a escalabilidade (SCHWARZ, MARPE e WIEGAND, 2007) (WIEGAND et al., 2007) (SEGALL e SULLIVAN, 2007). O conceito de escalabilidade permite a cada dispositivo selecionar partes de um único *bitstream* do vídeo codificado para recepção a fim de adaptá-lo às necessidades do usuário final ou às diferentes

capacidades que os receptores apresentam. Assim, ao invés de criar uma multiplicidade de *bitstreams*, um para cada combinação de parâmetros, apenas um *bitstream* é gerado por vídeo e o dispositivo decodificador seleciona apenas as partes correspondentes à sua capacidade.

Existem, basicamente, três tipos de escalabilidade: temporal, espacial e de qualidade. As escalabilidades temporal, espacial e de qualidade descrevem casos em que subconjuntos do *bitstream* representam o conteúdo fonte com um tamanho de imagem reduzido (resolução espacial) ou com uma taxa de quadros reduzida (resolução temporal) ou com uma qualidade visual da imagem reduzida, respectivamente. Na escalabilidade de qualidade, entretanto, o sub-*bitstream* contém a mesma resolução espacial e temporal que o *bitstream* completo, mas com menor fidelidade. Combinações entre as técnicas também podem ser realizadas (SEGALL e SULLIVAN, 2007).

O padrão H.264/AVC tem suporte à escalabilidade através da extensão SVC (*Scalable Video Coding*) (WIEGAND et al., 2007). O trabalho apresentado nesta monografia está centrado na escalabilidade espacial do padrão H.264/AVC, também chamado, neste caso, de H.264/SVC. Este tipo de escalabilidade é atingido através de uma estrutura de múltiplas camadas de codificação. Em cada camada, a mesma imagem é disposta em resoluções diferentes, sendo que a primeira camada possui a menor resolução. Cada uma das resoluções seguintes é codificada pela sua respectiva camada, a qual apenas adiciona informações complementares ao *bitstream* gerado pela camada inferior ao invés de gerar um novo *bitstream* completo.

O foco deste trabalho está no Filtro Redutor de Efeito de Bloco existente entre as camadas de enriquecimento da escalabilidade espacial. Este filtro é usado para suavizar o efeito de bloco dos dados da camada inferior antes que estes sejam utilizados pela camada superior. O efeito de bloco ocorre quando, por efeito de um elevado passo de quantização, os pixels que compõem unidades chamadas blocos possuem valores muito diferentes dos pixels de blocos vizinhos. Assim, os limites de tais unidades podem ser percebidos em uma imagem, diminuindo a sua qualidade subjetiva.

O objetivo deste trabalho consiste na realização de uma arquitetura de um Filtro Redutor de Efeito de Bloco focado nos conceitos de escalabilidade espacial da

extensão SVC para o padrão H.264/AVC e integra-se nas pesquisas realizadas no Grupo de Arquiteturas e Circuitos Integrados (GACI), da UFPel, que vem trabalhando com temas diversos na área de compressão de vídeo com o padrão H.264/AVC (AGOSTINI, 2007), e também nos trabalhos de investigação do Instituto de Telecomunicações (IT), da Universidade de Coimbra.

Este trabalho também irá contribuir com o desenvolvimento do hardware a ser utilizado pelo Sistema Brasileiro de Televisão Digital (SBTVD) em suas gerações futuras, já que o sistema utilizado atualmente ainda não suporta nenhum tipo escalabilidade. No SBTVD, há um particular interesse pela escalabilidade espacial, já que esta seria uma maneira de viabilizar uma convivência harmônica de receptores SDTV (*Standard-Definition Television* - Televisão de Definição Padrão) e HDTV (*High-Definition Television* - Televisão de Alta Definição) (TOME, 2007), cujas resoluções estão definidas no padrão atual.

Esta monografia está organizada da seguinte forma: o capítulo 2 trata de conceitos básicos pertinentes à codificação de vídeos digitais e ao codificador H.264/AVC. Os conceitos de escalabilidade e as inovações implementadas pela extensão escalável do padrão H.264/AVC são apresentadas no capítulo 3. O capítulo 4 apresenta o funcionamento detalhado dos módulos que compõem o Filtro Redutor de Efeito de Bloco e descreve outros trabalhos encontrados na literatura acerca do filtro. O capítulo 5 apresenta a arquitetura desenvolvida para o módulo alvo deste trabalho, a metodologia da sua descrição em VHDL e os resultados de síntese obtidos. No capítulo 6, são apresentadas as conclusões deste trabalho e no Apêndice A são listadas as publicações do aluno durante a graduação.

2 COMPRESSÃO DE VÍDEO

Este capítulo apresenta conceitos básicos relacionados à compressão de vídeo que são necessários para o entendimento dos capítulos seguintes e da arquitetura desenvolvida neste trabalho. Além disso, apresenta-se o funcionamento geral do codificador H.264/AVC.

2.1 Introdução à Compressão de Vídeo Digital

Vídeos são compostos por uma série de imagens estáticas chamadas quadros (ou *frames*) que, quando reproduzidas em sequência, podem induzir a percepção de movimento. Cada quadro é formado por um conjunto de macroblocos que, por sua vez, são formados por conjuntos de 4x4 blocos. Um bloco, no padrão H.264/AVC, é uma matriz de 4x4 pixels. Um macrobloco, portanto, é uma matriz de ou 16x16 pixels. Cada uma das quatro linhas de um bloco é chamada de LOP (*Line of Pixels* – Linha de Pixels).

O número de imagens reproduzidas por segundo, designado por taxa temporal ou, em inglês, *frame rate*, determina o quão suave serão percebidos os movimentos em uma cena. Cada imagem estática é formada por uma matriz de blocos, os quais são formados por uma matriz de pontos chamados pixels. Assim, dependo da resolução do vídeo, a quantidade de dados manipulados pode ser extremamente alta, dificultando o seu armazenamento e a transmissão de informações. Por exemplo, para transmitir-se um vídeo de resolução de 720 x 480 pixels a 30 quadros por segundo (usado em televisão digital com definição normal), utilizando 24 bits por pixel e sem utilizar compressão, a taxa de transmissão necessária seria de aproximadamente 249 milhões de bits por segundo (249 Mbps). Em termos de armazenamento, seriam necessários 19 bilhões de bytes (19 GB) para uma sequência de 10 minutos deste vídeo.

A compressão de um vídeo é, portanto, essencial para a viabilidade do funcionamento de aplicações que possam manipulá-lo. Para que a compressão possa ser realizada, as técnicas existentes valem-se de uma propriedade importante

dos vídeos digitais: o elevado grau de redundância. Assim, grande parte da enorme quantidade de dados existentes em uma sequência de imagens pode ser descartada, possibilitando a representação do vídeo digital com uma quantidade de bits muito menor do que a original.

2.2 Redundância de Dados

Em um vídeo digital, redundante é todo dado que não possui informação relevante para a representação de uma imagem. Existem três tipos de redundância que podem ser exploradas na compressão de vídeos: redundância espacial, redundância temporal e redundância entrópica.

2.2.1 Redundância Espacial

Também chamada de “redundância intraquadro” ou “redundância interpixel”, é decorrente da correlação entre pixels existentes dentro de um mesmo quadro (GHANBARI, 2003). Se a correlação é percebida no domínio espacial, ou seja, se os valores de alguns pixels em um mesmo quadro são semelhantes, a redundância pode ser reduzida através de técnicas de “codificação intraquadro”, presentes na maioria dos padrões de codificação de vídeo atuais. Caso a correlação seja percebida no domínio das frequências, a redundância espacial é reduzida através de uma operação chamada de quantização, conforme explicado na seção 2.5 deste trabalho.

2.2.2 Redundância Temporal

Também chamada de “redundância interquadros” ou “redundância interquadro”, é decorrente da correlação entre quadros temporalmente vizinhos (GHANBARI, 2003). Vários casos de redundância temporal podem ocorrer. Por exemplo, os valores de um bloco de pixels pode não mudar de um quadro para outro ou podem variar pouco. Em outros casos, o bloco de pixels pode aparecer simplesmente deslocado em relação à posição que ocupava no quadro anterior.

2.2.3 Redundância Entrópica

A entropia relaciona a quantidade média de informação transmitida por símbolo do vídeo. Quanto maior a probabilidade de um símbolo ocorrer, menor é a quantidade de informação nova transmitida por este símbolo. Assim, os

codificadores buscam transmitir o máximo de informação possível por símbolo codificado, ou seja, representar mais informações com um número menor de bits sem apresentar perdas.

2.3 Espaços de Cores

O sistema de visão do ser humano é composto por elementos chamados bastonetes e cones. Os primeiros são sensíveis à intensidade luminosa, enquanto que os cones são sensíveis às cores primárias. Assim, todas as cores são vistas como combinações das três cores primárias (vermelho, verde e azul). Milhares de cores distintas podem ser percebidas a partir de combinações de intensidades das cores primárias, mas apenas duas dúzias de tons de cinza são perceptíveis, os quais indicam variações na intensidade luminosa da imagem (GONZALES, 2003).

Espaços de cores são usados para codificar digitalmente pixels dentro de quadros. Diversos espaços, tais como RGB, HSI e YCbCr (SHI e SUN, 1999) vêm sendo utilizado para representar imagens digitais. O RGB, um espaço de cores simples e bastante utilizado, divide os pixels em três componentes: R (*red* – vermelho), G (*green* – verde) e B (*blue* – azul). O YCbCr também é composto por três componentes. O componente de luminância, Y, representa o brilho da imagem, enquanto que os dois componentes de crominância, Cb e Cr, representam o valor de desvio de uma cor a partir do cinza em direção ao azul e ao vermelho, respectivamente (BHASKARAN e KONSTANTINIDES, 1997).

Como o sistema visual percebe melhor as mudanças na intensidade do brilho em uma imagem do que as mudanças na intensidade da cor, o sistema YCbCr foi desenvolvido buscando tratar a informação de cor completamente separada da informação de brilho. Assim, os dois tipos de informação podem ser manipulados de forma diferenciada pelos codificadores de imagens estáticas e de vídeos. No H.264/AVC, por exemplo, o sistema YCbCr representa os componentes de crominância usando menos amostras em relação aos componentes de luminância (RICHARDSON, 2002).

2.4 Subamostragem em YCbCr

A eficiência da codificação aumenta significativamente quando a subamostragem de cor é utilizada, pois parte dos dados é simplesmente descartada sem causar qualquer impacto visual perceptível ao olho humano. Para tanto, existem diversas configurações da relação de subamostragem espacial das componentes de crominância em relação à componente de luminância. Os formatos mais comuns são o 4:4:4, o 4:2:2 e o 4:2:0.

No formato 4:4:4, para cada quatro amostras de luminância (Y), existem quatro amostras de crominância azul (Cb) e quatro amostras de crominância vermelha (Cr). Neste caso, nenhuma subamostragem é aplicada, pois a resolução dos três componentes é a mesma. No segundo caso, 4:2:2, existem duas amostras de crominância azul e duas amostras de crominância vermelha para cada quatro amostras de luminância. Já no formato 4:2:0, para cada quatro amostras de luminância, existe apenas uma amostra de cada tipo de crominância. O formato deveria, portanto, ser chamado 4:1:1. Contudo, por motivos históricos, esta não é a nomenclatura utilizada (RICHARDSON, 2003).

Considerando um vídeo codificado no formato 4:2:0, por exemplo, já que cada componente de crominância possui apenas um quarto das amostras do componente de luminância, pode-se dizer que a taxa de compressão é de 50% em relação a um vídeo no formato 4:4:4, utilizando apenas a subamostragem. Neste trabalho, o formato adotado é o 4:2:0, por ser o mais amplamente utilizado e por ser perfeitamente adequado à percepção do sistema visual humano (GONZALES, 2003). A escolha do formato adotado, contudo, não afeta de forma significativa o projeto da estrutura do Filtro Redutor de Efeito de Bloco, foco deste trabalho.

2.5 Codificador de Vídeo H.264/AVC

Os codificadores de vídeo fazem uso abundante das técnicas de redução de redundância, para assim conseguir reduzir o volume de dados que representam um sinal de vídeo. O codificador de vídeo do padrão H.264/AVC é composto por vários módulos que, direta ou indiretamente, têm como finalidade reduzir a redundância do sinal de vídeo, conforme indicado na Fig. 1 (AGOSTINI, 2007): estimação de movimento (*Motion Estimation* – ME), compensação de movimento (*Motion*

Compensation – MC), predição intraquadro, transformadas diretas (T) e inversas (T^{-1}), quantização direta (Q) e inversa (Q^{-1}), codificação de entropia e filtro.

Dois ou mais quadros são utilizados simultaneamente durante a codificação, sendo um o quadro atual e os outros os quadros de referência, que já foram anteriormente processados pelo codificador.

Em cada bloco do quadro atual, a codificação interquadros ou a codificação intraquadro é aplicada, conforme ilustra a chave seletora entre os dois modos de codificação na Fig. 1.

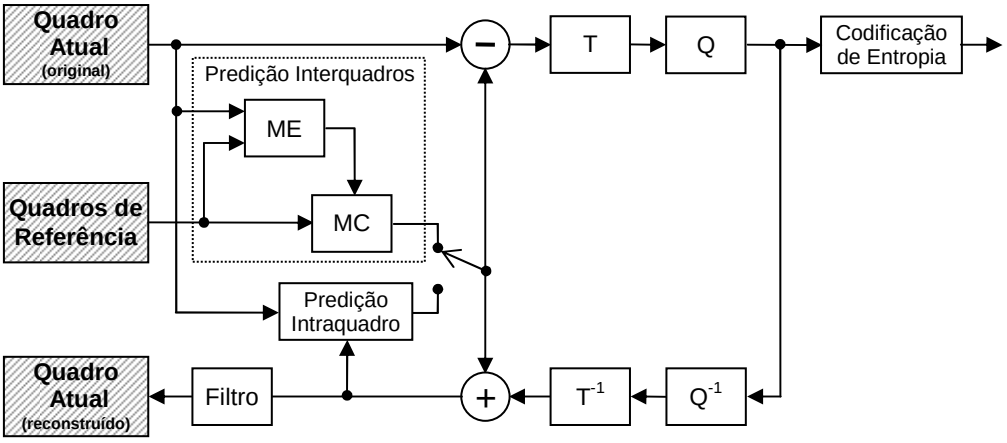

Figura 1 – Diagrama em blocos do codificador H.264/AVC (AGOSTINI, 2007)

A função do módulo de predição intraquadro é diminuir a redundância espacial. Assim, este módulo reutiliza informações já codificadas do quadro atual para realizar a predição do bloco que está sendo processado.

O módulo de predição interquadros é formado pela estimação de movimento (ME) e pela compensação de movimento (MC). Ele é responsável pela redução da redundância temporal, visto que opera realizando comparações entre blocos do quadro atual e dos quadros de referência. Neste modo, o bloco do quadro de referência com maior semelhança com o bloco em questão no quadro atual é escolhido e usado na codificação.

Após a codificação interquadros ou intraquadro, uma subtração entre o bloco codificado e o bloco original é realizada, dando origem a um conjunto de dados chamado de resíduo de predição. O resíduo é enviado, então, para os módulos responsáveis por reduzir a redundância espacial no domínio das frequências. Esta operação é realizada, primeiramente, pelo módulo de transformadas (T), que

converte a informação do domínio espacial para o domínio das frequências. Após isso, o módulo de quantização (Q) é aplicado com a finalidade de reduzir a redundância espacial presente nos resíduos.

Por fim, a redundância entrópica é reduzida através da codificação de entropia. Este tipo de codificação baseia-se na forma como os dados são codificados e na probabilidade de ocorrência dos símbolos (RICHARDSON, 2002). Explicações mais profundas sobre este tipo de codificação não são relevantes para este trabalho.

Como o processo de codificação gera perdas de informação, o quadro codificado seria diferente do quadro original após a decodificação. Assim, as referências utilizadas pelo codificador (quadro original) e pelo decodificador (quadro codificado reconstruído) seriam diferentes e esta diferença nas referências do codificador e do decodificador causariam distorções graves no vídeo, gerando grandes perdas de qualidade visual. Para resolver este problema, foi necessária a inserção, no codificador, de parte do decodificador. Desta forma, o quadro original é descartado após a sua codificação e o quadro reconstruído é armazenado e usado como referência para a codificação do próximo quadro.

Este processo de reconstrução do quadro no codificador é apresentado no *loop* da Fig. 1 que se estabelece entre o módulo Q^{-1} e o módulo Filtro. A operação inversa de quantização é aplicada e, após isso, a transformada inversa gera os resíduos reconstruídos. O resultado da predição intraquadro ou da predição interquadros é somado aos resíduos e o conteúdo resultante dessa adição passa pelo Filtro Redutor de Efeito de Bloco (módulo Filtro na Fig. 1). Após isso, a imagem pode ser armazenada para ser utilizada pela codificação interquadros do próximo quadro ou pode ser diretamente utilizado pela codificação intraquadro dos próximos blocos do quadro atual.

O padrão H.264/AVC normaliza a utilização do Filtro Redutor de Efeito de Bloco. Este filtro era opcional na maioria dos demais padrões de compressão de vídeo, mas passou a ser obrigatório no H.264/AVC. O Filtro Redutor de Efeito de Bloco é o foco deste trabalho e será apresentado detalhadamente no capítulo 4 desta monografia.

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 Introdução à Escalabilidade

Atualmente, surge um mercado com uma diversidade cada vez maior de dispositivos capazes de manipular vídeos com variadas resoluções, desde dispositivos com telas pequenas, como telefones móveis, até dispositivos com resoluções elevadas, como as TVs de alta definição.

Para fornecer serviços de difusão de vídeo a este mercado, uma alternativa possível 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 solução, chamada *simulcast* (SEGALL e SULLIVAN, 2007), consiste na transmissão de dois (ou mais) sinais independentes, onde são transmitidos sinais que atendam às necessidades dos equipamentos de baixa resolução e dos aparelhos de alta definição. No entanto, esta alternativa acarreta um grande desperdiço de banda de transmissão.

A escalabilidade é uma alternativa extremamente atrativa, pois propõe a existência de uma camada que representa o sinal de vídeo com uma resolução espacial, temporal e/ou de qualidade mais baixa, com menos informações, e uma ou várias camadas superiores que apenas adicionam informações à camada inferior para alcançar resolução, qualidade ou taxa de amostragem mais elevadas. Deste modo, existe apenas um sinal sendo transmitido e o equipamento receptor seleciona apenas a camada de seu 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 para que os modos de escalabilidade mais importantes fossem suportados (SCHWARZ, MARPE e WIEGAND, 2007). Contudo, 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, além de um aumento considerável 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 resultou 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 escalável 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.

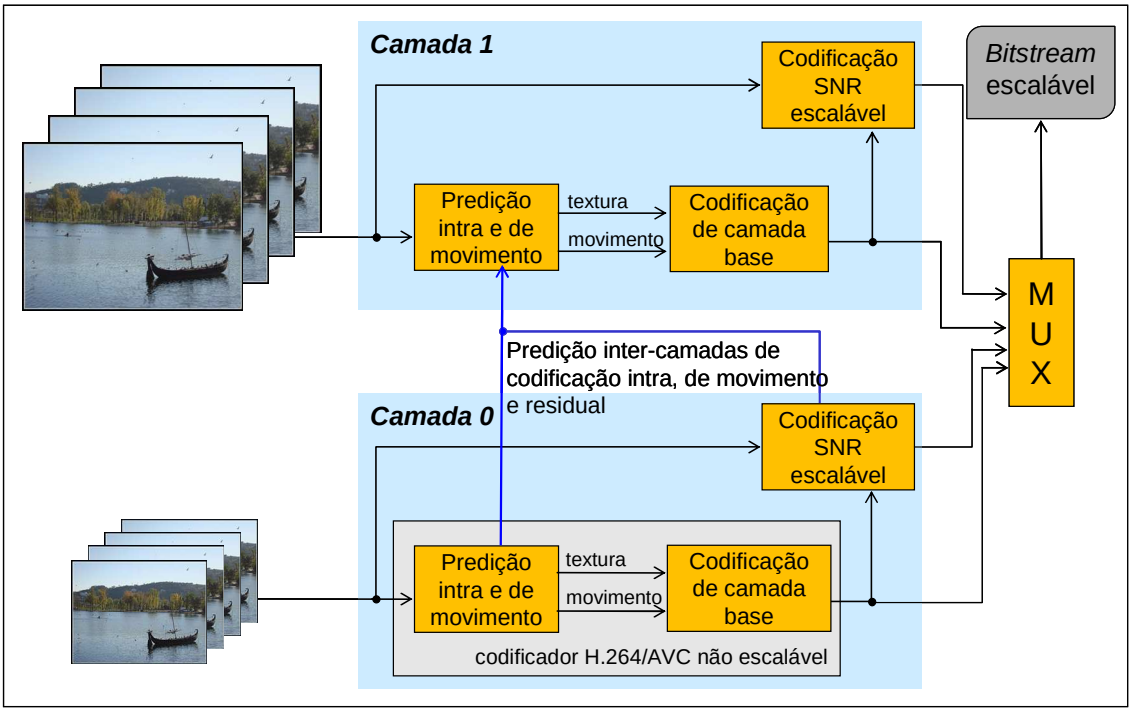

Figura 2 – Diagrama em blocos de um codificador da extensão escalável do padrão H.264/AVC

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 amostragem 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 hierárquica de quadros tipo B (que utilizam, como referência, tanto quadros anteriores quanto quadros futuros).

3.3.2 Escalabilidade de Qualidade

A escalabilidade de qualidade é comumente chamada de escalabilidade de fidelidade ou escalabilidade SNR (*Signal-to-Noise Ratio*) (SCHWARZ, MARPE e WIEGAND, 2007). Este tipo de escalabilidade pode ser considerado um caso especial em que a 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 intercamadas. Isso possibilita ao codificador escolher livremente que informação da camada de referência será explorada (WIEN, SCHWARZ e OELBAUM, 2007). A seguir serão apresentados os mecanismos de predição intercamadas introduzidos pelo padrão.

3.3.4 Predição Intercamadas

No SVC existem três mecanismos que podem ser utilizados na predição intercamadas: Predição de Movimento, Predição Residual e Predição Intra. As descrições, presentes nos próximos parágrafos, estão restritas ao caso diádico de escalabilidade espacial, caracterizado pela duplicação na altura e na largura da imagem de uma camada para a outra (SEGALL e SULLIVAN, 2007).

3.3.4.1 Predição de Movimento Intercamadas

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 interquadros, o macrobloco na camada de enriquecimento também será codificado com predição interquadros. Neste caso, os dados de particionamento, os índices de referência (indicam qual lista de quadros[1] foi usada como referência) e os vetores de

[1] O padrão H.264/AVC define duas listas contendo os quadros de referência. Alista 0 contém o quadro passado mais próximo, seguido de quaisquer outros quadros passados, seguidos de quaisquer outros quadros futuros. A

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 Intercamadas, 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 Intercamadas

A Predição Residual Intercamadas 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 Intercamadas ou os tipos de macroblocos convencionais.

Devido à Predição de Movimento Intercamadas, camadas espaciais subsequentes 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, uma *flag* (chamada *residual prediction flag*) é adicionada à sintaxe do macrobloco. Quando esta *flag* é igual a 1, é realizada uma operação de *upsampling* no sinal residual do bloco 8x8 correspondente na camada de referência. O sinal gerado pelo *upsampling* é, então, usado como predição para o sinal residual da camada de enriquecimento e, assim, somente a diferença correspondente é codificada.

3.3.4.3 Predição Intra Intercamadas

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 intraquadro, o sinal de predição na camada de enriquecimento é obtido através do *upsampling* do sinal intra correspondente da

---

lista 1 contém o quadro futuro mais próximo, seguido de quaisquer quadros futuros, seguidos de quaisquer outros quadros passados.

camada de referência (SCHWARZ, MARPE e WIEGAND, 2007). Contudo, antes da realização do *upsampling* neste caso, é necessária a aplicação de um Filtro Redutor de Efeito de Bloco Intercamadas, conforme ilustra a Fig. 3. Tal filtro é aplicado para que quaisquer artefatos gerados na imagem devido a um elevado passo de quantização seja suavizado antes da informação da camada de referência ser utilizada pela camada de enriquecimento.

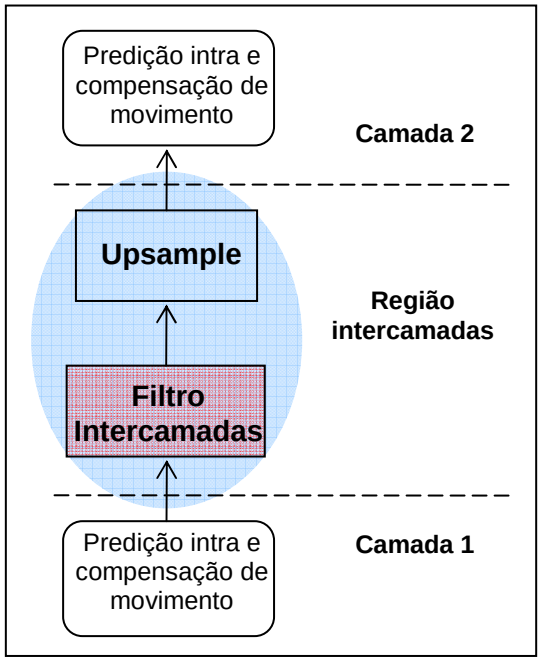

Figura 3 – Predição Intra Intercamadas, com destaque para o Filtro Redutor de Efeito de Bloco Intercamadas.

O Filtro Redutor de Efeito de Bloco Intercamadas é o foco deste trabalho e será explicado com mais profundidade no capítulo 4 desta monografia.

Quando uma imagem da camada de referência é filtrada, somente os blocos que foram codificados em modo intra são processados. Isto acontece devido ao conceito de *single loop* do SVC, o qual define que a compensação de movimento nunca é realizada em uma camada de referência (SCHWARZ, MARPE e WIEGAND, 2007). Na camada de enriquecimento, os blocos são, portanto, preditos após a sua filtragem e sobreamostragem (*upsampling*) na camada de referência. Caso não haja outra camada de enriquecimento, a compensação de movimento pode ser realizada e os resíduos de blocos que sofreram codificação interquadros são adicionados à imagem. Por fim, o Filtro Redutor de Efeito de Bloco não escalável é aplicado para gerar a imagem final.

Em termos de algoritmo, o processo de Predição Intra Intercamadas ocorre da seguinte forma:

1) Decodificação de informações de modo de codificação e de resíduos da camada de referência;
2) Realização de predição intra na camada de referência e adição de resíduos;
3) Filtragem da camada de referência reconstruída (filtro intercamadas). Contudo, todos os blocos codificados através de codificação interquadros não estão disponíveis e, portanto, não têm as suas bordas filtradas;
4) Decodificação das informações de modo de codificação e resíduos da camada de enriquecimento;
5) Para todo bloco I_BL[2], realização de predição das informações filtradas da camada de referência;
6) Compensação de movimento na camada de enriquecimento;
7) Adição de resíduos na camada de enriquecimento;
8) Filtragem da camada de enriquecimento (filtro não escalável). Neste caso, os blocos de codificação interquadros estão disponíveis e podem, portanto, ser filtrados.

[2] Se todos os blocos 4x4 de luminância do macrobloco de referência correspondem a blocos da camada inferior codificados com codificação intra, o tipo macrobloco é dito I_BL e todos os seus blocos são ditos de tipo I_BL.

4 FILTRO REDUTOR DE EFEITO DE BLOCO

Neste capítulo, é apresentado o funcionamento geral do Filtro Redutor de Efeito de Bloco do padrão H.264/AVC e do **Filtro Redutor de Efeito de Bloco Intercamadas** da extensão SVC do mesmo padrão. São apresentadas as ordens de processamento para o filtro propostas por outros autores e a ordem de filtragem proposta e utilizada neste trabalho.

4.1 Motivação

O Filtro Redutor de Efeito de Bloco (ou, em inglês, *Deblocking Filter*) foi incluído no padrão H.264/AVC para proporcionar um nível maior na qualidade subjetiva da imagem, após o processo de reconstrução ou decodificação, através da remoção de artefatos que levam à percepção de descontinuidades artificiais nas fronteiras entre blocos. No padrão H.264/AVC, o Filtro está localizado após o módulo de transformada inversa no codificador (antes da reconstrução e armazenamento do macrobloco para predições futuras), conforme mostrado na Fig. 1, e no decodificador (antes da reconstrução e exibição do macrobloco).

A Fig. 4 mostra um quadro original, codificado de acordo com o padrão H.264/AVC. O resultado da decodificação sem o uso do Filtro Redutor de Efeito de Bloco é apresentado na Fig. 5. A Fig. 6 mostra o mesmo quadro decodificado, porém com a utilização do Filtro. Este exemplo mostra a importância do Filtro no padrão.

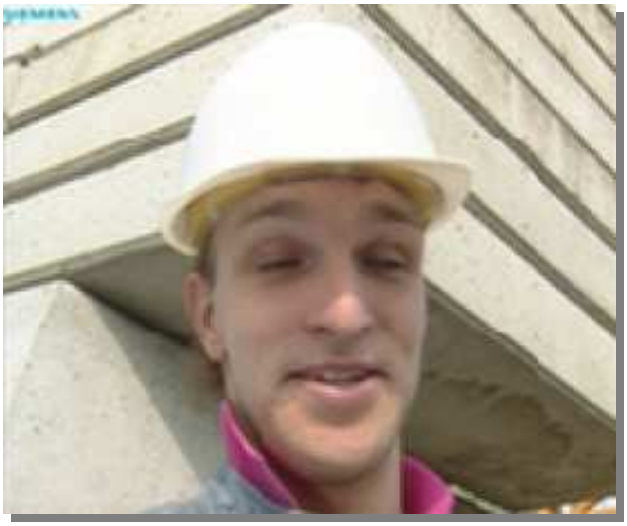

Figura 4 – Quadro original codificado conforme o padrão H.264/AVC

Figura 5 – Quadro decodificado sem a utilização do Filtro Redutor de Efeito de Bloco

Figura 6 – Quadro decodificado com a utilização do Filtro Redutor de Efeito de Bloco

4.2 Funcionamento do Filtro Redutor de Efeito de Bloco no Padrão H.264/AVC

O Filtro Redutor de Efeito de Bloco é um filtro adaptativo, ou seja, ele consegue distinguir uma aresta real da imagem, que não deve ser filtrada, de um artefato gerado por um elevado passo de quantização, que deve ser filtrado. A filtragem é aplicada nas bordas verticais e horizontais dos blocos 4x4 de um macrobloco, de acordo com os passos 1 a 4 apresentados abaixo e na Fig. 7.

1) Filtrar as quatro bordas verticais do componente de luminância (**a**, **b**, **c** e **d** na Fig. 7);
2) Filtrar as quatro bordas horizontais do componente de luminância (**e**, **f**, **g** e **h** na Fig. 7);
3) Filtrar as duas bordas verticais de cada componente de crominância (**i** e **j** na Fig. 7) e
4) Filtrar as duas bordas horizontais de cada componente de crominância (**k** e **l** na Fig. 7).

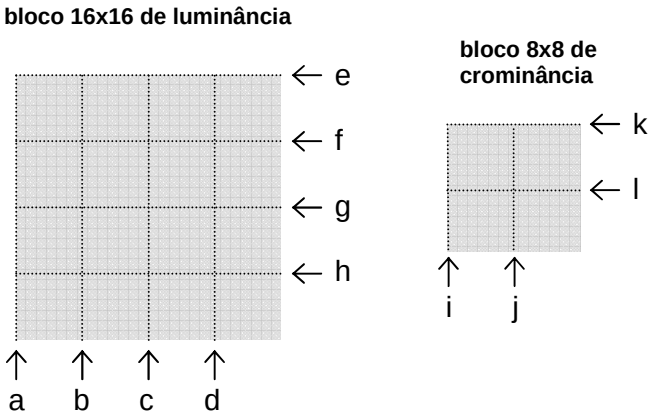

Figura 7 – Ordem de filtragem de bordas em macroblocos de luminância e de crominância

Considerando macroblocos de luminância, cada operação de filtragem afeta até três amostras de cada lado da borda. A Fig. 8 mostra quatro amostras de cada lado de uma borda vertical ou horizontal de blocos **p** e **q** adjacentes (p_0, p_1, p_2, p_3 e q_0, q_1, q_2 e q_3). As amostras de crominância são filtradas da mesma maneira, mas com a diferença de que apenas os valores de cinco pixels (p_0, p_1, q_0, q_1 e q_2) são usados nos cálculos e duas amostras (p_0 e q_0) são modificadas durante a filtragem.

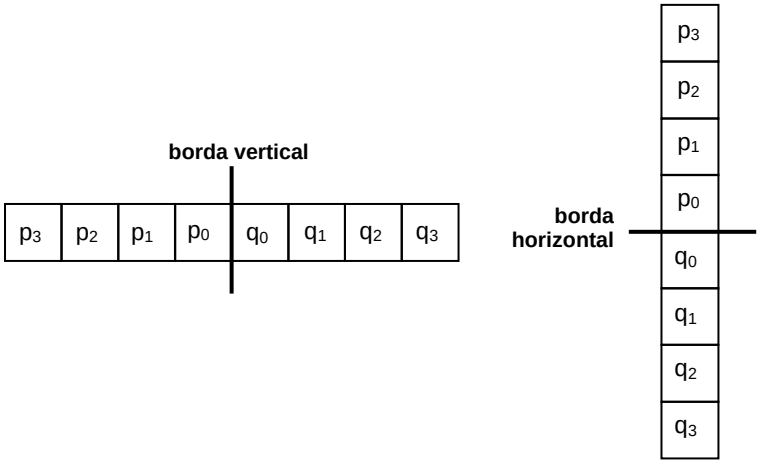

Figura 8 – Amostras de luminância adjacentes em bordas verticais e horizontais

4.2.1 Parâmetros de Filtragem do Filtro do Padrão H.264/AVC

A intensidade (ou força) da filtragem a ser realizada depende da quantização utilizada no bloco, do modo de codificação dos blocos vizinhos e do gradiente (intensidade da mudança de valores) das amostras da imagem através da borda.

Existem cinco diferentes forças de filtragem, sendo estas definidas pelo parâmetro bS (*boundary strength*), que pode variar de zero até quatro. Existem, portanto, cinco diferentes forças de filtragem, onde um bS = 4 define uma força máxima e um bS = 0 define que nenhuma filtragem é realizada. O parâmetro é definido através de uma série de regras definidas pelo padrão. As regras da versão não escalável do H.264/AVC não são relevantes para este trabalho. O algoritmo com as regras para o filtro utilizado na região intercamadas da extensão SVC será apresentado na seção 4.3.

O resultado da aplicação de tal algoritmo é que a filtragem é mais forte em locais onde há uma probabilidade maior de haver efeito de bloco mais significativo, tais como a borda de um macrobloco que sofreu codificação intra ou uma borda entre blocos que contêm coeficientes codificados (RICHARDSON, 2002).

A decisão de realizar a filtragem, contudo, não depende apenas do parâmetro bS. Considerando um grupo de amostras (p_2, p_1, p_0, q_0, q_1, q_2), a filtragem acontece somente se todas as condições apresentadas em (1), (2), (3) e (4) forem verdadeiras (RICHARDSON, 2002).

$$bS > 0 \quad (1)$$

$$| p_0 - q_0 | < \alpha(Index_A) \quad (2)$$

$$| p_1 - p_0 | < \beta(Index_B) \quad (3)$$

$$| q_1 - q_0 | \leq \beta(Index_B) \quad (4)$$

Os parâmetros α e β são limites (*thresholds*) definidos no padrão. Eles dependem dos índices $Index_A$ e $Index_B$ que, por sua vez, são dependentes da média dos parâmetros de quantização (QP) utilizados nos blocos filtrados e de *offsets* definidos no codificador ($Offset_A$ e $Offset_B$), conforme (5) e (6).

$$Index_A = Min\ (Max\ (0, QP + Offset_A), 51) \quad (5)$$

$$Index_B = Min\ (Max\ (0, QP + Offset_B), 51) \quad (6)$$

$Index_A$ e $Index_B$ são utilizados para acessar tabelas que possuem valores de α e β derivados através de testes (JVT, 2003). Os valores de $Offset_A$ e $Offset_B$ são transmitidos no cabeçalho de uma *slice* do macrobloco para fins de otimização da decodificação, ajustando os valores de α e β.

Os parâmetros α e β aumentam de acordo com a média do QP de dois blocos **p** e **q**. O efeito dos limites na decisão de filtragem é o de “desligar” o filtro quando há uma mudança significativa na borda do bloco da imagem original. Quando o QP é baixo e acontece uma mudança significativa de valores nas amostras de uma borda esta mudança provavelmente acontece devido a características da imagem que devem ser preservadas (e não por efeito de bloco). Neste caso, α e β são baixos. Quando o QP é alto, a quantização gera mais perdas e, portanto, a distorção por efeito de bloco é mais significativa. Neste caso, α e β devem ser mais altos para que mais amostras sejam filtradas.

4.2.2 Processo de Filtragem do Filtro do Padrão H.264/AVC

O processo de filtragem pode ser dividido em dois casos. No primeiro caso, o valor de bS é igual a 1, 2 ou 3. No segundo caso, o valor de bS é igual a 4 (LIST, 2003). Em ambos os casos, as condições (7) e (8) são avaliadas.

$$| p_2 - p_0 | < \beta(Index_B) \quad (7)$$

$$| q_2 - q_0 | < \beta(Index_B) \quad (8)$$

Essas condições avaliam a intensidade da mudança de valores que existe nas amostras internas a um dos blocos adjacentes. Se houver uma grande mudança de valores nas amostras em questão, uma filtragem forte resultaria em perda de detalhes existentes e que devem ser preservados. Caso contrário, uma filtragem forte resultaria em uma imagem mais suave.

4.2.2.1 Filtragem com bS de 1 a 3

Quando bS é igual a 1, 2 ou 3, um filtro de 4 *taps* é aplicado com quatro amostras de entrada (p_1, p_0, q_0 e q_1), produzindo duas saídas filtradas (p'_0 e q'_0). Caso a condição (7) seja avaliada como verdadeira e as amostras sejam de luminância, outro filtro de 4 *taps* é aplicado com entradas p_2, p_1, p_0 e q_0, produzindo uma saída filtrada p'_1. Se a condição (8) for verdadeira e se as amostras forem de luminância, um filtro de 4 *taps* é aplicado com entradas q_2, q_1, q_0 e p_0, produzindo uma saída filtrada q'_1.

Os cálculos de p'_0 e q'_0 deste caso acontecem de acordo com (9) e (10). O valor de Δ_0 é determinado através da truncagem ou, do inglês, do *clipping* do valor de Δ_{0i}, o qual é calculado de acordo com (11).

$$p'_0 = p_0 + \Delta_0 \quad (9)$$

$$q'_0 = q_0 - \Delta_0 \quad (10)$$

$$\Delta_{0i} = (4\,(q_0 - p_0) + (p_1 - q_1) + 4) >> 3 \quad (11)$$

Os valores de p'_1 e q'_1 são gerados através de (12) e (13). Os valores de Δ_{p1} e Δ_{q1} são calculados em um processo de dois passos, semelhantes ao de Δ_0, ou seja, eles são valores gerados após o *clipping* de Δ_{p1i} e Δ_{q1i}, respectivamente, conforme (14) e (15).

$$p'_1 = p_1 - \Delta_{p1} \quad (12)$$

$$q'_1 = q_1 - \Delta_{q1} \quad (13)$$

$$\Delta_{p1i} = (p_2 + ((p_0 + q_0 + 1) >> 1) - 2p_1) >> 1 \quad (14)$$

$$\Delta_{q1i} = (q_2 + ((q_0 + p_0 + 1) >> 1) - 2q_1) >> 1 \quad (15)$$

Se os valores resultantes de (11), (14) e (15) fossem utilizados diretamente pelo filtro, um efeito de borrão poderia ocorrer devido ao excesso de amostras filtradas. Assim, operações de *clipping* são aplicadas para limitar esses valores Δ (LIST, 2003). Os valores Δ utilizados na filtragem de amostras internas passam por um *clipping* na faixa de $-c_1$ a c_1, onde c_1 é um parâmetro determinado com base em uma tabela indexada por $Index_A$ e bS. O valor de c_1 cresce de acordo com $Index_A$ e bS, permitindo uma filtragem mais forte. Os valores de Δ_{p1} e Δ_{q1} são, portanto, calculados de acordo com (16) e (17).

$$\Delta_{p1} = \text{Min}\,(\text{Max}\,(-c_1, \Delta_{p1i}), c_1) \quad (16)$$

$$\Delta_{q1} = \text{Min}\,(\text{Max}\,(-c_1, \Delta_{q1i}), c_1) \quad (17)$$

Para a filtragem de p_0 e q_0, a faixa de *clipping* aplicada a Δ_{0i} é determinada com base no valor de c_1 e na avaliação das condições (7) e (8). O valor de c_0 é,

inicialmente, igual ao de c_1 e, então, incrementado em 1 para cada condição avaliada como verdadeira. A equação (18) mostra como o valor de Δ_0 é calculado.

$$\Delta_0 = \text{Min}\,(\text{Max}\,(-c_0, \Delta_{0i}), c_0) \quad (18)$$

Na filtragem de amostras de crominância, somente os valores de p_0 e q_0 podem ser modificados. Estas são filtradas da mesma forma que as amostras de luminância, exceto pelo valor de *clipping* c_0, que é sempre igual a $c_1 + 1$. Desta forma, não há necessidade de avaliar as condições (7) e (8).

4.2.2.2 Filtragem com bS igual a 4

Se o valor de bS for igual a 4, dois tipos diferentes de filtro podem ser usados, dependendo do conteúdo das amostras: um filtro muito forte de 4 e 5 *taps* que modifica a amostra mais próxima à borda e duas amostras internas de cada bloco, ou um filtro mais fraco de 3 *taps* que modifica apenas a amostra mais próxima à borda. O filtro mais forte é aplicado somente se a condição (19) for avaliada como verdadeira:

$$| p_0 - q_0 | < (\alpha >> 2) + 2 \quad (19)$$

Quando as condições (7) e (8) são verdadeiras, as amostras são filtradas conforme (20), (21) e (22) para as amostras do bloco **p** e conforme (23), (24) e (25) para as amostras do bloco **q**.

$$p'_0 = (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4) >> 3 \quad (20)$$

$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \quad (21)$$

$$p'_2 = (2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4) >> 3 \quad (22)$$

$$q'_0 = (q_2 + 2q_1 + 2q_0 + 2p_0 + p_1 + 4) >> 3 \quad (23)$$

$$q'_1 = (q_2 + q_1 + q_0 + p_0 + 2) >> 2 \quad (24)$$

$$q'_2 = (2q_3 + 3q_2 + q_1 + q_0 + p_0 + 4) >> 3 \quad (25)$$

As amostras de luminância que não satisfazem (7) ou (8) e as amostras de crominância são filtradas conforme (26) e (27).

$$p'_0 = (2p_1 + p_0 + q_1 + 2) >> 2 \quad (26)$$

$$q'_0 = (2q_1 + q_0 + p_1 + 2) >> 2 \quad (27)$$

4.3 Filtro Redutor de Efeito de Bloco Intercamadas (H.264/SVC)

Conforme explicado na seção 3.3.4.3, o Filtro Redutor de Efeito de Bloco Intercamadas existente no padrão H.264/AVC escalável é diferente do Filtro do padrão não escalável. A diferença está na definição da força de filtragem.

A força de filtragem bS, apresentada na seção 4.2.1, pode variar entre 0 e 4 no filtro do padrão não escalável. No padrão escalável, contudo, o valor de bS igual a 3 nunca é utilizado. O algoritmo da Fig. 9 define qual tipo de filtragem deve ser realizada, considerando uma borda entre dois blocos **p** e **q**. O algoritmo é executado somente se o tipo de predição é intercamadas e se pelo menos um dos blocos **p** e **q** pertence a um macrobloco do tipo I_BL.

```
Se p ou q pertencem a um MB intra diferente de I_BL
   Então bS = 4
         Fim.
Se p e q pertencem a um MB intra do tipo I_BL
   Então
   Se existe coeficiente ≠ 0 nos blocos de coeficientes
   de transformadas correspondentes à amostra p0 ou à
   amostra q0.
            Então bS = 1
                  Fim.
            Senão bS = 0
                  Fim.
Se p ou q pertencem a um MB inter
   Então
   Se p (ou q) pertence a um MB inter e o array de
   resíduos contém alguma amostra ≠ 0 no bloco de
   amostras correspondentes à amostra p0 (ou q0).
            Então bS = 2
                  Fim.
            Senão bS = 1
                  Fim.
```

Figura 9 – Algoritmo utilizado para definir a força de filtragem no Filtro Redutor de Efeito de Bloco Intercamadas

4.4 Ordens de Processamento do Filtro

Devido à sua complexidade, uma ampla pesquisa vem sendo realizada sobre a implementação do Filtro Redutor de Efeito de Bloco do padrão H.264/AVC ou SVC. A principal fonte de complexidade do filtro pode ser atribuída ao fato de que

cada pixel precisa ser lido múltiplas vezes, em diferentes sentidos, para a filtragem de um macrobloco.

A filtragem ocorre em bordas de blocos de 4x4 pixels, sendo que 16 desses blocos estão localizados dentro de um macrobloco e 8 estão localizados em macroblocos adjacentes (à esquerda e acima). Além disso, as amostras devem ser filtradas no sentido horizontal e vertical, o que gera um problema de alinhamento na leitura da memória. Este problema será abordado com mais detalhes no capítulo 5 deste trabalho.

O problema de alinhamento na leitura da memória, associado ao número de vezes que as amostras são lidas e escritas, levou muitos pesquisadores a trabalhar em métodos de reuso de dados. Já que os valores dos pixels precisam ser lidos múltiplas vezes e os resultados intermediários de filtragens são necessários para os próximos estágios, é possível modificar a ordem das filtragens para que os dados intermediários gerados possam ser utilizados mais cedo pelo filtro, liberando espaço na memória.

A única restrição imposta pelo padrão H.264/AVC com relação à ordem de processamento das amostras especifica que as filtragens horizontais que envolvem um pixel devem ocorrer antes das verticais. A ordem de processamento do filtro proposta pelo padrão H.264/AVC (JVT, 2003) está apresentada na Fig. 10. Como se pode perceber, as bordas verticais dos blocos de luminância e de crominância são todas filtradas antes das bordas horizontais. Como os resultados das filtragens verticais precisam ser utilizados na filtragem horizontal, esses dados devem ser armazenados. A ordem de processamento proposta em (JVT, 2003) é, portanto, extremamente custosa em termos de utilização de memória, pois há necessidade de armazenar as amostras de 24 blocos (16 de luminância, 4 de crominância azul e 4 de crominância vermelha) até que a filtragem horizontal ocorra.

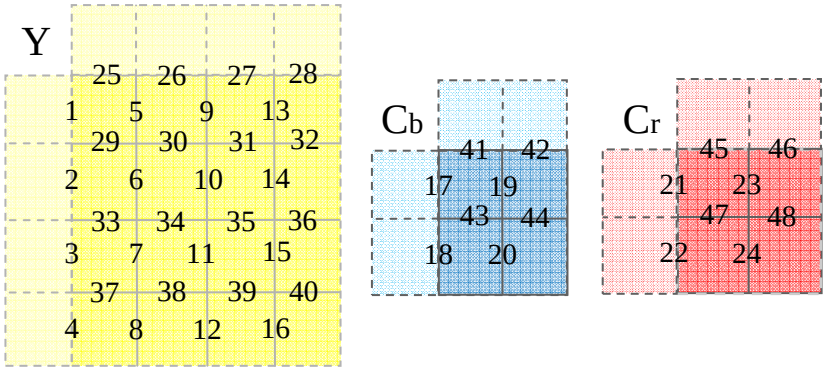

Figura 10 – Ordem de Processamento do Filtro Redutor de Efeito de Bloco proposta pelo padrão H.264/AVC (JVT, 2003)

A ordem de processamento proposta por (KHURANA, 2006) apresentada na Fig. 11 faz uso de uma alternância entre a filtragem horizontal e vertical dos blocos. O resultado da alternância é que uma linha de blocos 4x4 pode ser armazenada na memória local para ser reusada na próxima borda. Depois que os pixels são completamente filtrados (ou seja, filtrados nos dois sentidos), eles podem ser escritos na memória principal para serem exibidos ou para serem usados como referência.

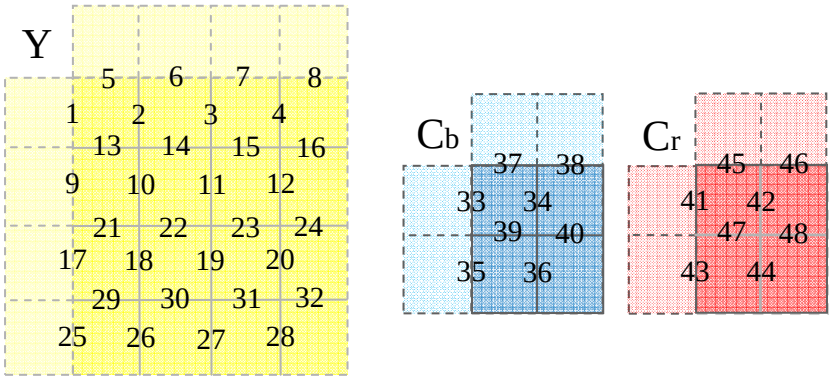

Figura 11 – Ordem de Processamento do Filtro Redutor de Efeito de Bloco proposta em (KHURANA, 2006)

Uma terceira ordem de processamento, proposta por (SHENG, 2004), baseia-se no mesmo princípio de alternância proposto por (KHURANA, 2006). Contudo, a frequência da mudança de filtragem horizontal para vertical (e vice-versa) é maior, ocorrendo quase toda vez que uma borda 4x4 é filtrada, como mostrado na Fig. 12. Como resultado, obtém-se uma redução no tamanho da memória local, já que o reuso de dados ocorre muito mais cedo.

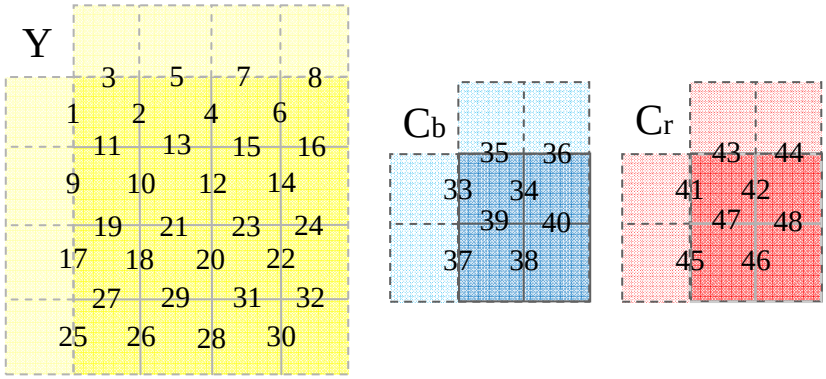

Figura 12 – Ordem de Processamento do Filtro Redutor de Efeito de Bloco proposta em (SHENG, 2004)

A proposta de (LI, 2005) baseia-se no mesmo princípio de reuso de dados, mas se aprofunda mais no sentindo de buscar a concorrência entre as filtragens. A ordem de processamento, mostrada na Fig. 13, reduz significativamente o número de ciclos necessários para a filtragem do macrobloco através da execução da filtragem horizontal e vertical ao mesmo tempo. Para isso, dois filtros separados são necessários, um para cada sentido de filtragem.

Com base nesta proposta, uma arquitetura concorrente que utiliza quatro núcleos de filtragem foi apresentada por (ERNST, 2007). Esta arquitetura será melhor detalhada no próximo capítulo.

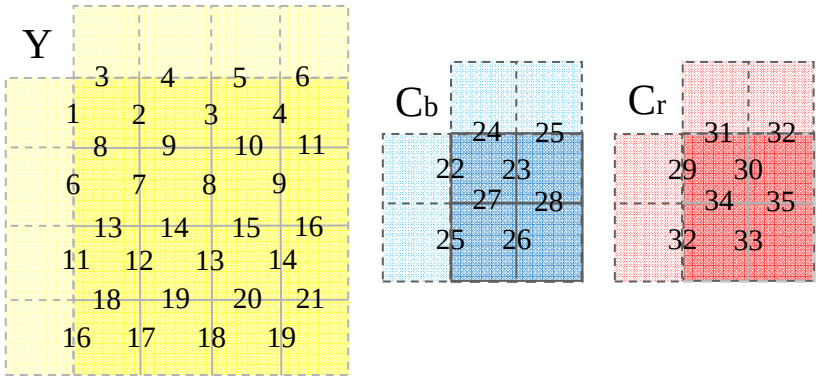

Figura 13 – Ordem de Processamento do Filtro Redutor de Efeito de Bloco proposta em (LI, 2005)

O número de filtragens concorrentes é, contudo, bastante limitado devido às dependências existentes entre os dados. Utilizando a abordagem de (LI, 2005), seria impossível realizar três ou mais filtragens concorrentes no mesmo macrobloco sem aumentar significativamente o tamanho da memória local.

Todas as ordens de processamento apresentadas até aqui são realizadas em nível de bloco, isto é, o processamento de uma borda de bloco 4x4 é realizado serialmente pelo mesmo filtro e a borda de um bloco só pode começar a ser filtrada quando a filtragem de todas as LOPs (*Line of Pixels* – Linha de Pixels) do bloco antecessor (à esquerda) terminar.

Neste trabalho, propõe-se uma ordem de processamento em nível de amostras, ao invés de blocos. Desta forma, a filtragem da primeira LOP de um bloco pode começar quando o resultado da filtragem da primeira LOP do bloco antecessor estiver concluída. Por exemplo, na Fig. 14, quando a filtragem entre LOP_{P1} e LOP_{Q1} estiver concluída, a filtragem entre LOP_{Q1} e LOP_{R1} pode começar, antes mesmo da conclusão das filtragens das outras três LOPs dos blocos à esquerda de R. Este tipo de abordagem facilita a execução de filtragens em paralelo, já que a dependência de dados é tratada a um nível de granularidade mais elevada.

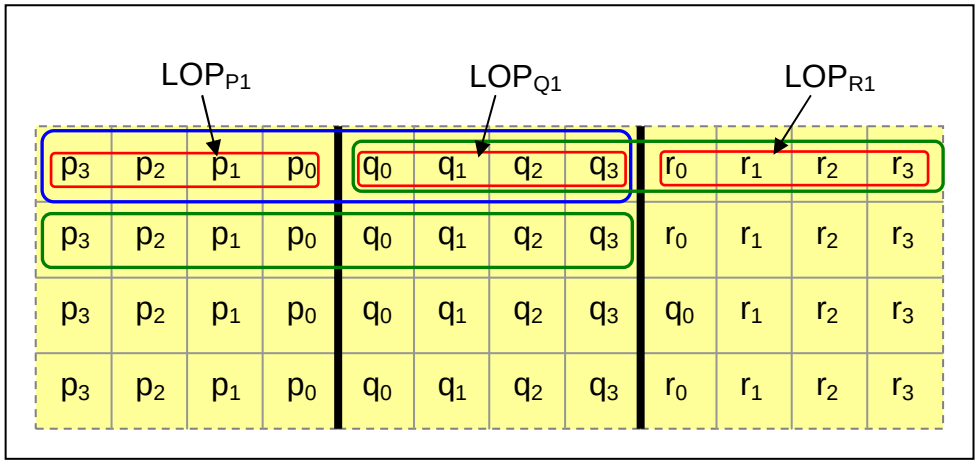

Figura 14 – Filtragem concorrente em nível de amostras

O processamento em nível de amostras permite, portanto, que o paralelismo da arquitetura seja explorado de melhor forma, sem aumentar significativamente o uso de memória temporária utilizada e sem aumentar o número de núcleos de filtragem utilizados, conforme explicado no próximo capítulo. A Fig. 15 mostra a ordem de processamento proposta neste trabalho. Os números iguais correspondem a filtragens ocorrendo em paralelo, no mesmo ciclo, por núcleos diferentes. Como se pode perceber, são realizadas quatro filtragens em paralelo, o que resultou no projeto de uma arquitetura com quatro núcleos de filtragem detalhada no capítulo 5.

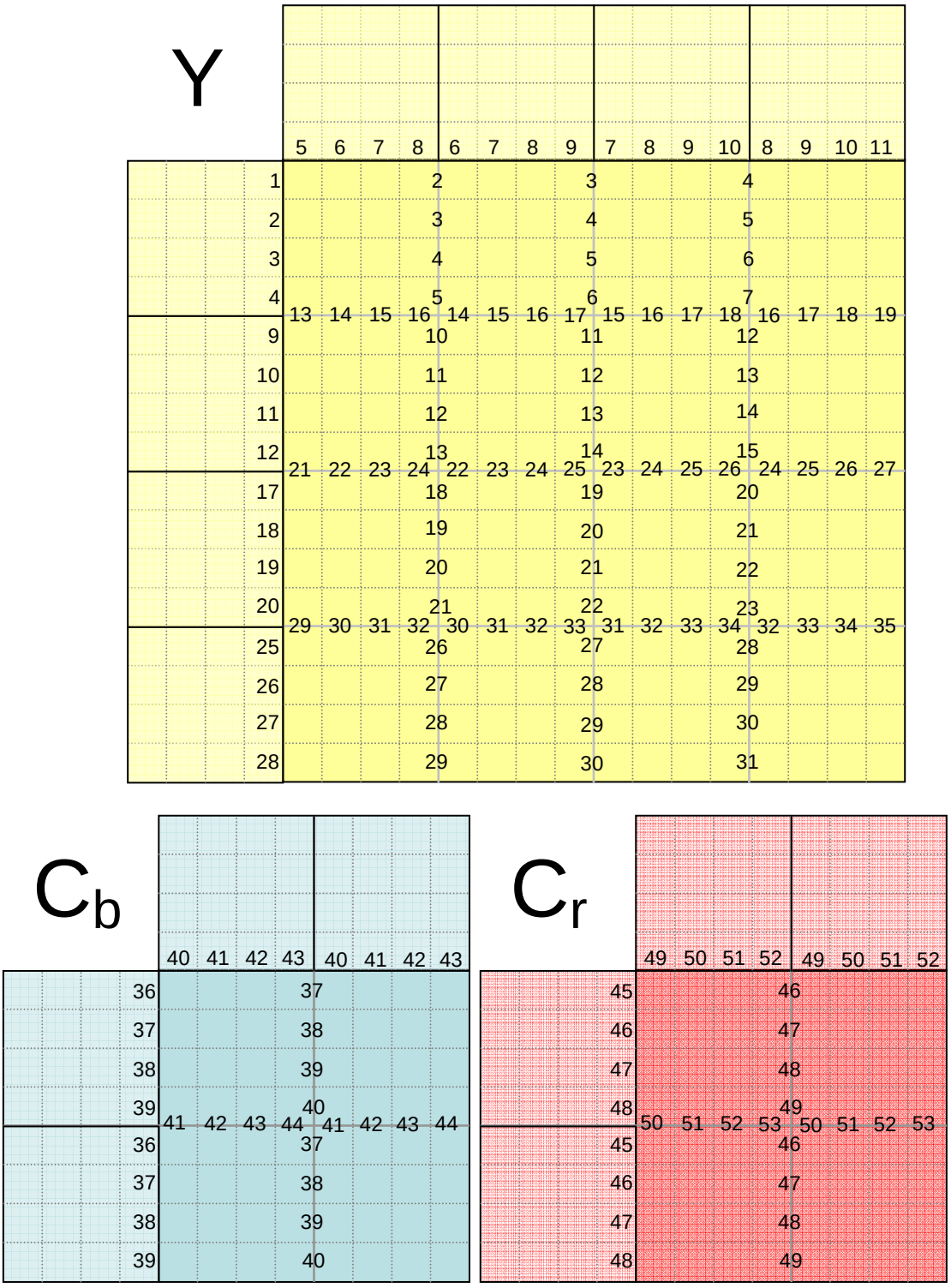

Figura 15 – Ordem de Processamento do Filtro Redutor de Efeito de Bloco proposta neste trabalho

A ordem de processamento proposta pelo padrão H.264/AVC (Fig. 10) é a que utiliza a maior quantidade de memória para armazenamento de resultados intermediários, já que os resultados de todas as filtragens verticais dos 24 blocos são armazenados até que as filtragens horizontais comecem. Com o aumento da

frequência de alternância entre filtragens verticais e horizontais dos blocos (Fig. 11 e Fig. 12), a quantidade de memória utilizada reduz. A proposta de (SHENG, 2004) (Fig. 12) é a que utiliza, dentre todas as propostas, a menor quantidade de memória, conforme será mostrado na seção 5.3 desta monografia. A ordem proposta por (LI, 2005), contudo, é a que melhor se assemelha à proposta de processamento paralelo apresentada neste trabalho e é utilizada, em nível de blocos, na arquitetura de (ERNST, 2007).

A seção 5.3 apresenta os resultados obtidos neste trabalho, incluindo os resultados em termos de ciclos utilizados e de memória utilizada para a ordem de processamento proposta neste trabalho.

5 DESENVOLVIMENTO DO FILTRO

A arquitetura desenvolvida para o Filtro Redutor de Efeito de Bloco Intercamadas é apresentada neste capítulo, com base na ordem de processamento proposta no capítulo 4. Antes, entretanto, é relatada a análise do software de referência do padrão SVC.

5.1 Estudo do Software de Referência do SVC

O trabalho objeto deste relatório iniciou-se com uma investigação aprofundada acerca do funcionamento do Filtro Redutor de Efeito de Bloco Intercamadas. Para tanto, o funcionamento do software de referência do padrão escalável, o JSVM – *Joint Scalable Video Model* (versão 9.12.2) (ITU-T e ISO/IEC, 2008) foi analisado para complementar o estudo que foi feito da norma H.264/AVC com extensão SVC e aprofundar a compreensão de como o filtro é aplicado.

De acordo com o manual do software de referência, o código referente ao Filtro Redutor de Efeito de Bloco está na biblioteca *H264AVCCommonLibStatic*, a qual provê classes que são usadas tanto pelo codificador quanto pelo decodificador, tais como estruturas de dados de macroblocos, *buffers* e algoritmos do filtro.

O arquivo *Loopfilter.cpp* contém todos os métodos que são utilizados tanto pelo filtro intercamadas, quanto pelo filtro não escalável (mas também aplicado à imagem final no codificador ou decodificador escalável). Contudo, alguns testes realizados com as variáveis lógicas *bInterLayerFlag* (indica a ocorrência de predição intercamadas) e *bSpatialScalabilityFlag* (indica a ocorrência de escalabilidade espacial) fazem com que o funcionamento dos métodos aconteça de acordo com o filtro utilizado. A não existência de arquivos e métodos separados para cada filtro dificulta a sua compreensão.

O método *xFilterMb*, o mais externo da filtragem de um macrobloco, recebe todas as informações de codificação de um macrobloco, um objeto da classe *YuvPicBuffer,* que contém os valores de todas as amostras de luminância e

crominâncias do macrobloco, um objeto da classe *YuvPicBuffer* com os valores de todas as amostras de resíduo, um *array* com os parâmetros de quantização (QPs) de todos os blocos do macrobloco e a *flag bSpatialScalabilityFlag*. O primeiro conjunto de operações realizadas diz respeito ao cálculo da força de filtragem (bS). Para isso, são invocados os métodos *xGetVerFilterStrength* e *xGetHorFilterStrength*, que realizam o cálculo da força de filtragem no sentido vertical e horizontal, respectivamente. Logo após, são invocados os métodos *xLumaVerFiltering*, *xLumaHorFiltering*, *xChromaVerFiltering* e *xChromaHorFiltering*, responsáveis pela filtragem das bordas verticais e horizontais de luminância e verticais e horizontais de crominância, respectivamente. É importante notar que a ordem de invocação dos métodos obedece àquela ordem imposta pelo padrão H.264/AVC, ou seja, *xLumaVerFiltering* filtra todas as bordas verticais de luminância, *xLumaHorFiltering* filtra todas as bordas horizontais de luminância e assim por diante.

Os métodos *xLumaVerFiltering*, *xLumaHorFiltering*, x*ChromaVerFiltering* e *xChromaHorFiltering*, realizam os cálculos dos parâmetros α, β e $Index_A$ utilizados em cada filtragem e invocam o método *xFilter*, que realiza as operações de filtragem entre dois blocos, recebendo os valores de α, β, $Index_A$ e bS.

Da mesma forma que o software de referência, a arquitetura apresentada neste trabalho também realiza o cálculo dos parâmetros utilizados na filtragem por módulos distintos, o que facilita uma futura validação da arquitetura através de comparações com resultados obtidos usando o software JSVM.

A fim de verificar-se o efeito da aplicação do filtro intercamadas, realizou-se a decodificação de um vídeo com o filtro funcionando normalmente e, posteriormente, a decodificação do mesmo vídeo com o filtro intercamadas desabilitado. Como o funcionamento dos dois filtros não é explicitamente separado no código, o filtro intercamadas não pode ser desativado apenas através da *flag* correspondente ao Filtro Redutor de Efeito de Bloco nos arquivos de configuração do codificador ou do decodificador. Para isso, foi necessária uma modificação no software de referência para que a força de filtragem fosse sempre igual a zero quando *bInterLayerFlag* é verdadeira. O efeito da modificação no software é semelhante àquele apresentado na Fig. 2 deste trabalho.

5.2 Arquitetura Desenvolvida para o Filtro

As subseções seguintes descrevem a arquitetura proposta neste trabalho. A descrição foi feita em linguagem VHDL (*VHSIC Hardware Description Language*) (ASHENDEN, 1998) e todos os módulos foram sintetizados através do software Quartus II da Altera (ALTERA, 2009). Para a validação dos módulos, as suas operações foram escritas em código C, com base no código de referência, e os resultados da execução foram comparados com os resultados obtidos em simulações da arquitetura a partir do software Quartus II. Todos os módulos foram validados separadamente, com exceção do módulo de controle. Apesar de sintetizada, arquitetura completa não foi validada através de simulações neste trabalho por limitações de tempo. Pretende-se, contudo, realizar a validação completa através da ferramenta ModelSim, da Mentor Graphics (MENTOR, 2009), em um trabalho futuro.

5.2.1 Matriz de Transposição

Conforme mencionado na seção 4.4, como as amostras de um bloco são filtradas no sentido vertical e no sentido horizontal, existe um problema de alinhamento na leitura da memória em que os blocos estão armazenados.

Por exemplo, quando uma filtragem de borda vertical ocorre, uma linha da memória possui todas as amostras de um lado da borda que serão utilizadas. Contudo, quando a filtragem é horizontal, as amostras estão localizados em uma coluna que passa por quatro linhas da memória. Desta forma, um circuito de transposição é necessário para que realinhar os dados utilizados nas filtragens de bordas horizontais. Um circuito deste tipo seria utilizado duas vezes: antes da filtragem horizontal e após, para a escrita das amostras filtradas em uma memória externa.

Para evitar a necessidade de um circuito que consuma ciclos adicionais aos utilizados pela filtragem, a arquitetura de uma matriz de transposição baseada na proposta de (KHURANA, 2006), é utilizada neste trabalho. A diferença da arquitetura utilizada aqui, contudo, é a possibilidade de leitura da memória nos dois sentidos concorrentemente. Além dos valores das amostras que compõem cada bloco, a matriz de transposição também possui quatro registradores que armazenam algumas informações de codificação do bloco.

A Fig. 16 ilustra a arquitetura de uma matriz de transposição 4x4, composta por oito *flip-flops* (FFs) que armazenam pixels de oito bits. A arquitetura possibilita que as amostras sejam escritas nos dois sentidos, conforme os sinais de controle recebidos pela matriz.

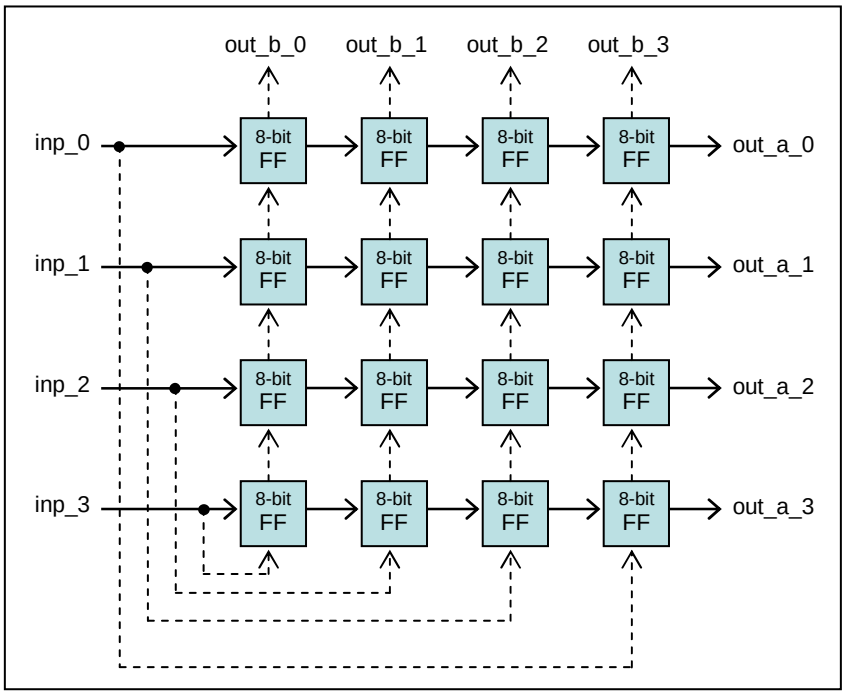

Figura 16 – Arquitetura de uma matriz de transposição 4x4

A Fig. 17 mostra todos os sinais de entrada e saída da matriz de transposição projetada. Os sinais *clk* e *flag_clk* habilitam a matriz de transposição e os quatro registradores adicionais que armazenam as informações de codificação do bloco, respectivamente. Os sinais *vert_w*, *vert_r* e *vert_r2*, identificam os sentidos que a escrita e as leituras são realizadas na memória, respectivamente. Como duas leituras concorrentes podem ser realizadas, dois sinais para leitura são necessários. *write_line*, *read_line* e *read_line2* identificam qual linha da matriz de transposição é escrita e lida. O sinal *write_enable* habilita a escrita na matriz e *write_three* identifica uma escrita de apenas três amostras na memória, conforme explicado anteriormente. A entrada das amostras acontece através dos sinais *inp_0*, *inp_1*, *inp_2* e *inp_3* e as informações de codificação estão identificadas pelos sinais *RB_nZ_in* (*flag* que identifica se existe alguma amostra diferente de zero no bloco de resíduos correspondente ao bloco armazenado na matriz), *TLB_nZ_in* (identifica a existência de algum coeficiente diferente de zero no bloco de coeficientes de

transformadas correspondente)[3], *QP_in* (parâmetro de quantização aplicado ao bloco) e *type_MB_in* (tipo de codificação aplicada no macrobloco). As saídas identificadas de *out_a_0* até *out_a_3* e *out_b_0* até *out_b_3* correspondem às amostras lidas. As demais saídas correspondem às informações de codificação.

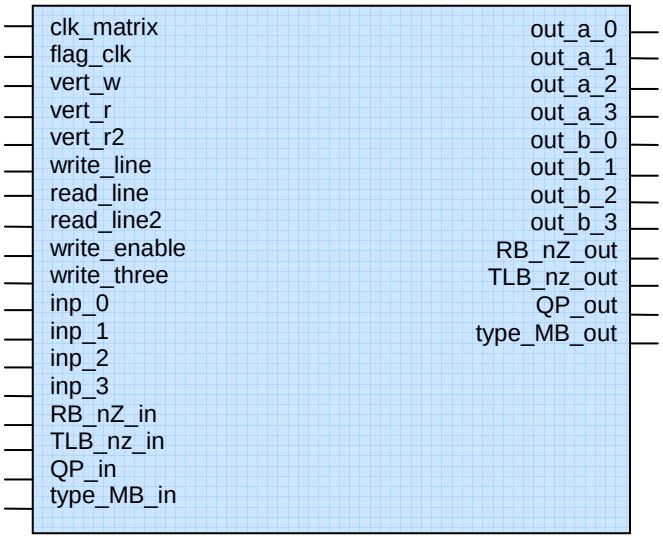

Figura 17 – Sinais de entrada (esquerda) e saída (direita) de uma matriz de transposição 4x4

5.2.2 Calculador de limites (*thresholds*)

Esta unidade, apresentada na Fig. 18, é responsável por determinar os limites (também chamados de *thresholds*) α, β e $Index_A$, que são entradas do núcleo de filtragem. Os QPs de entrada são correspondentes aos dois blocos que compõem a borda (**p** e **q**). Os *offsets* dependem do codificador e são informados no cabeçalho de cada *slice* do macrobloco.

[3] É importante, para o correto funcionamento do filtro projetado, que as *flags RB_nZ_in* e *TLB_nZ_in* estejam disponíveis entre as informações de codificação de cada bloco filtrado.

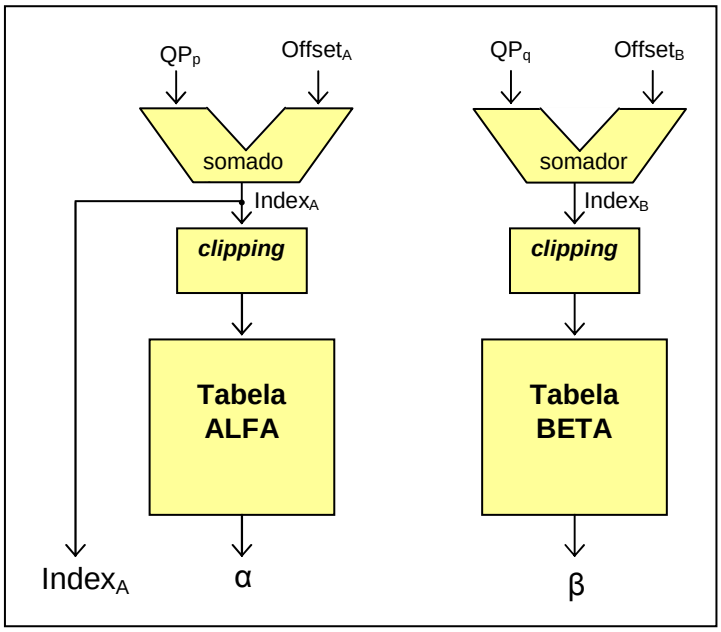

Figura 18 – Arquitetura do calculador de limites

A Fig. 19 apresenta todos os sinais que compõem a entrada e a saída do calculador de limites.

Figura 19 – Sinais de entrada (esquerda) e saída (direita) do módulo calculador de limites

5.2.3 Calculador de bS

Este módulo é responsável pelo cálculo da força de filtragem de cada borda dentro de um macrobloco. A força de filtragem calculada é enviada ao núcleo de filtragem do Filtro. A arquitetura deste módulo baseia-se, basicamente, na realização de diversos testes, sem que qualquer operação aritmética seja realizada. Os testes realizados foram apresentados na Fig. 9.

A Fig. 20 mostra todos os sinais de entrada e saída do calculador de bS. Os sinais *p0_type* e *q0_type* indicam, respectivamente, o tipo de codificação aplicado ao macrobloco que contém a amostra p_0 ou q_0. Os sinais *TLB_nZ_p* e *TLB_nZ_q* são *flags* que indicam a existência de algum coeficiente diferente de zero nos blocos de coeficientes de transformadas correspondentes ao blocos **p** e **q**, respectivamente.

Da mesma forma, *res_nZ_p* e *res_nZ_q* indicam a existência de amostras diferentes de zero nos blocos de resíduos correspondentes ao blocos **p** e **q**.

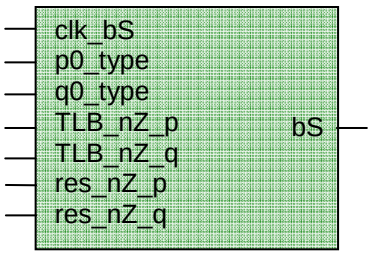

Figura 20 – Sinais de entrada (esquerda) e saída (direita) do módulo calculador de bS

5.2.4 Calculador de c_1

O módulo calculador de c_1 contém três memórias com os valores de truncagem definidos pelo padrão H.264/AVC quando o valor de bS é igual a 1, 2 ou 3. Para indexar as memórias, o valor de $Index_A$, recebido do calculador de limites, é utilizado. A Fig. 21 mostra os sinais de entrada e saída do calculador de c_1.

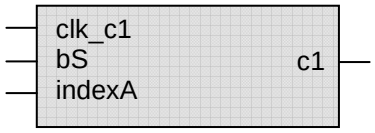

Figura 21 – Sinais de entrada (esquerda) e saída (direita) do módulo calculador de c_1

Quando bS = 1, os valores de c_1 são mais baixos e, portanto, menos bits são necessários para armazenar a tabela. Neste caso, é utilizada uma memória de 52 palavras de 4 bits (208 bytes). Nos outros dois casos (bS = 2 e bS = 3), são necessárias palavras de 5 bits. Cada uma das memórias possui, portanto, 260 bytes.

5.2.5 Núcleo de Filtragem

Este é o módulo principal do Filtro Redutor de Efeito de Bloco, pois realiza a filtragem propriamente dita. A Fig. 22 apresenta todas as entradas e saídas do módulo.

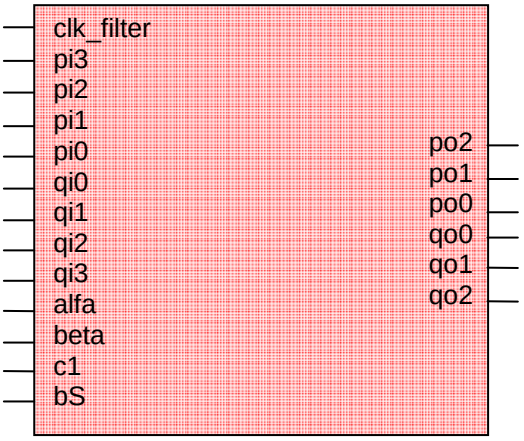

Figura 22 – Sinais de entrada (esquerda) e saída (direita) do núcleo de filtragem

Os sinais *pi0* até *pi3* e os sinais *qi0* até *qi3* correspondem às quatro amostras pertencentes ao bloco **p** e às quatro amostras do bloco **q** que são filtradas, respectivamente. Igualmente, os sinais *po0* até *po2* e *qo0* até *qo2* correspondem às seis amostras alteradas após a filtragem.

As operações realizadas pelo filtro estão descritas na seção 4.2.2 deste trabalho. Contudo, para fins de simplificação da arquitetura, algumas modificações foram realizadas nos tipos e na ordem de realização das operações. Em primeiro lugar, como as operações realizadas pelo filtro são, basicamente, aritméticas, é possível perceber que uma grande quantidade de cálculos se repete ao longo da execução da filtragem. Assim, sinais intermediários foram criados para guardar os resultados de cálculos que podem ser compartilhados entre as expressões. Além disso, operações de multiplicação, bastante onerosas em termos arquiteturais, foram substituídas por deslocamentos, muito mais simples e rápidos.

Por exemplo, a equação (11) apresentada na seção 4.2.2 foi substituída pela equação (28), apresentada abaixo. Neste caso, a multiplicação por quatro foi substituída por dois deslocamentos à esquerda.

$$\Delta_{0i} = ((q_0 - p_0) << 2) + (p_1 - q_1) + 4) >> 3 \qquad (28)$$

A Fig. 23 mostra uma parte da arquitetura do núcleo de filtragem, correspondente aos cálculos das amostras quando bS é igual a 1, 2 ou 3. São utilizados cinco somadores, sete subtratores, sete deslocadores e três unidades de truncagem.

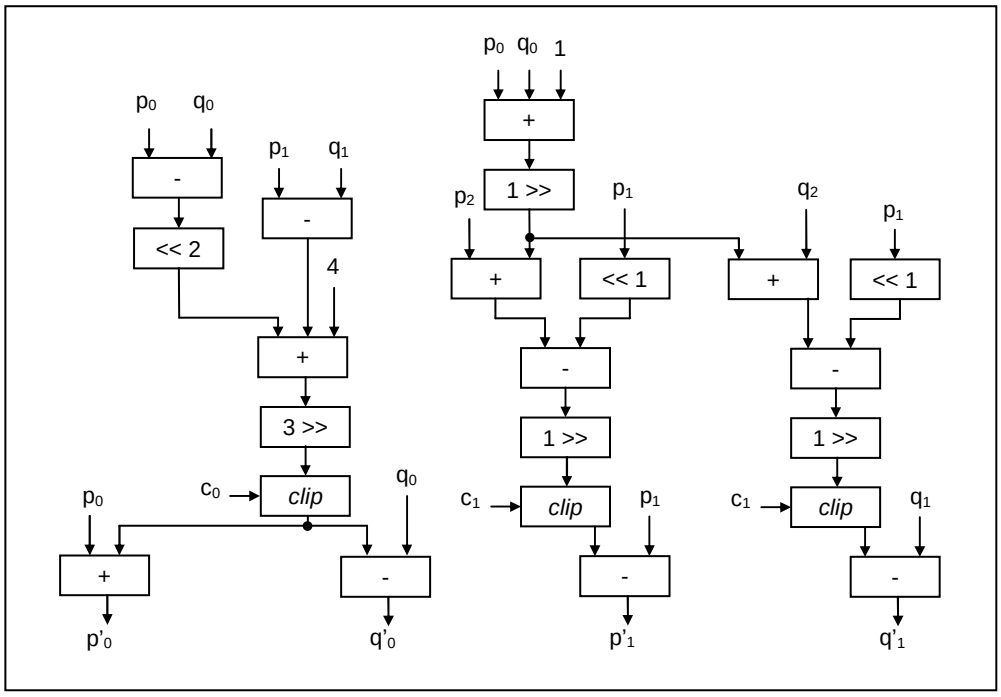

Figura 23 – Parte do núcleo de filtragem para cálculos quando bS = 1, 2 e 3

Quando bS = 4, as partes do módulo de filtragem apresentadas na Fig. 24 e na Fig. 25 geram os resultados das amostras p'_0, p'_1, p'_2, q'_0, q'_1 e q'_2. Neste caso, são utilizados 19 somadores e 14 deslocadores.

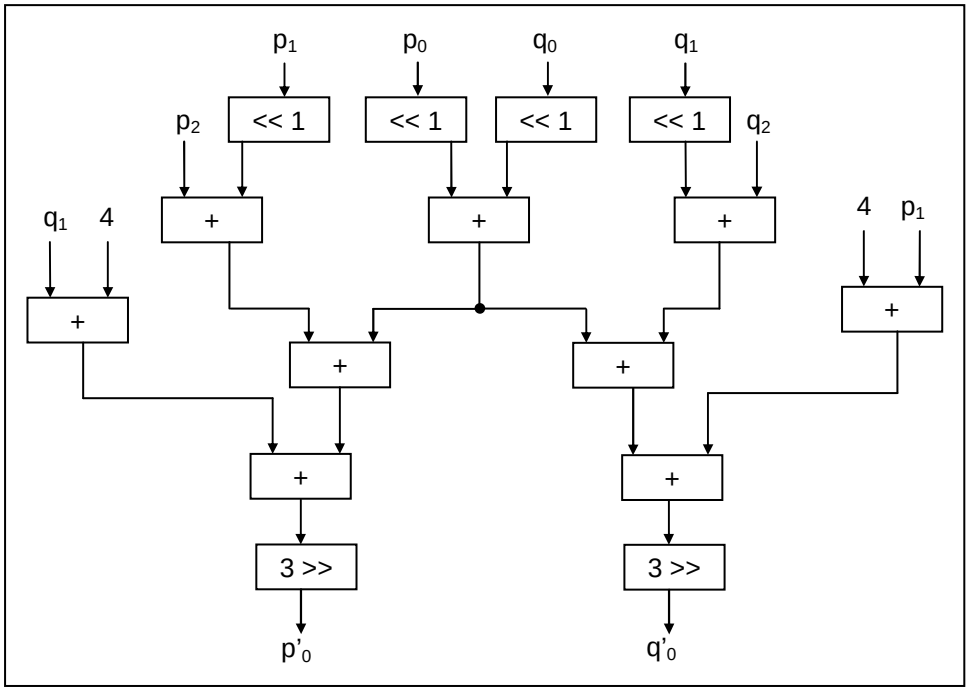

Figura 24 – Parte do núcleo de filtragem para cálculos de p'_0 e q'_0 quando bS = 4

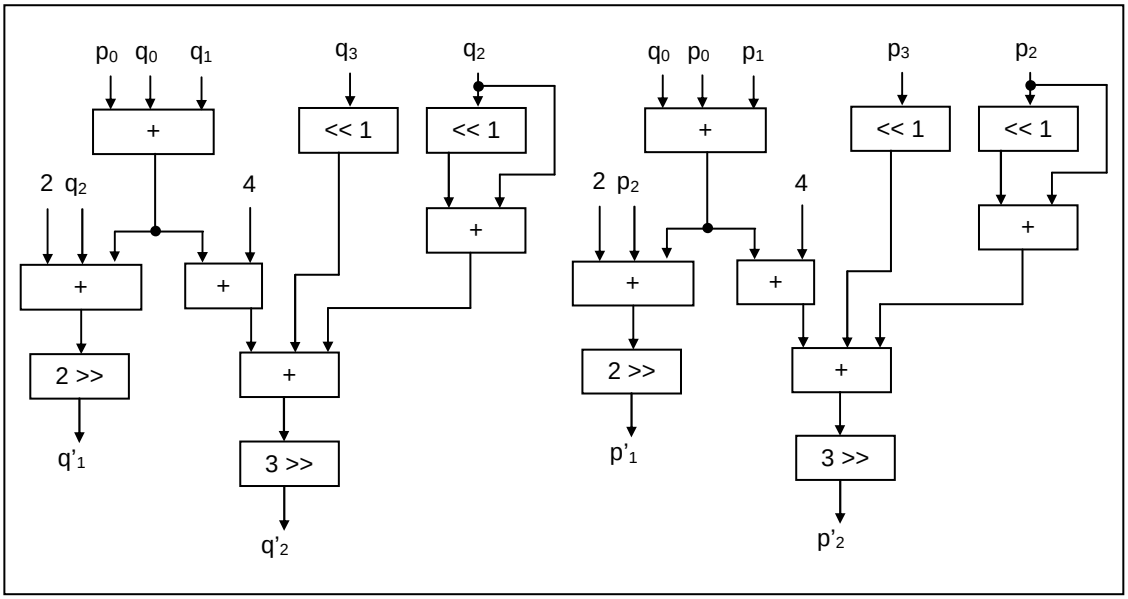

Figura 25 – Parte do núcleo de filtragem para cálculos de q'_1, q'_2, p'_1 e p'_2 quando bS = 4

No total, foram utilizados 24 somadores, 7 subtratores, 21 deslocadores e 3 unidades de truncagem na arquitetura do núcleo de filtragem.

5.2.6 Arquitetura Completa

A arquitetura completa do Filtro é composta por um módulo calculador de bS, um módulo calculador de limites, um módulo calculador de c_1, oito matrizes de transposição e quatro núcleos de filtragem, conforme ilustra a Fig. 26.

Os módulos calculadores de bS, limites e c_1 (bS_calc, thr_calc e c1_calc na Fig. 26, respectivamente) são ligados a multiplexadores que entregam os seus resultados ao núcleo de filtragem correspondente. Como esses valores não mudam entre as LOPs de um mesmo bloco, são necessários *buffers* para guardar os valores de bS, α, β, $Index_A$ e c_1 durante os três ciclos seguintes à primeira filtragem, já que tais valores são necessários em todas as filtragens de um mesmo bloco.

Os núcleos de filtragem (FE1, FE2, FE3 e FE4) adjacentes na Fig. 26 estão interligados devido à possibilidade de filtragem concorrente em nível de amostras. Por exemplo, na Fig. 14, quando a filtragem entre LOP_{P1} e LOP_{Q1} termina, a nova $LOP_{P'1}$ filtrada será escrita na sua matriz de transposição. Contudo, como a filtragem entre $LOP_{Q'1}$ e LOP_{R1} já acontece no ciclo seguinte, o resultado $LOP_{Q'1}$ não é escrito na memória, mas é entregue ao núcleo responsável pela filtragem entre os blocos **q** e **r**.

As matrizes de transposição estão organizadas de forma que duas linhas completas de um macrobloco (8 blocos 4x4) possam estar armazenadas ao mesmo tempo. Assim, a primeira matriz de transposição (T1, na Fig. 26) contém as amostras do primeiro bloco do macrobloco, a segunda matriz contém as amostras do segundo bloco e assim por diante. Quando todas as filtragens forem realizadas em uma linha do macrobloco, as quatro matrizes correspondentes aos blocos filtrados têm os seus dados substituídos pelas amostras de duas linhas abaixo. Assim, no caso de um macrobloco de luminância, as matrizes T1 até T4 armazenam os dados dos blocos de 1 a 4 ou de 9 a 12 e as matrizes T5 até T8 armazenam os dados dos blocos de 5 a 8 ou de 13 a 16. No caso de macroblocos de crominância, as matrizes T1 e T2 armazenam os dados dos blocos 1 e 2 e as matrizes T5 e T6 armazenam os dados dos blocos 3 e 4. É importante observar na arquitetura da Fig. 26 que cada matriz de transposição está ligada a dois filtros. Isso ocorre porque as amostras de cada bloco são filtradas duas vezes em cada sentido – uma vez com o bloco à esquerda (ou acima) e outra vez com o bloco à direita (ou abaixo).

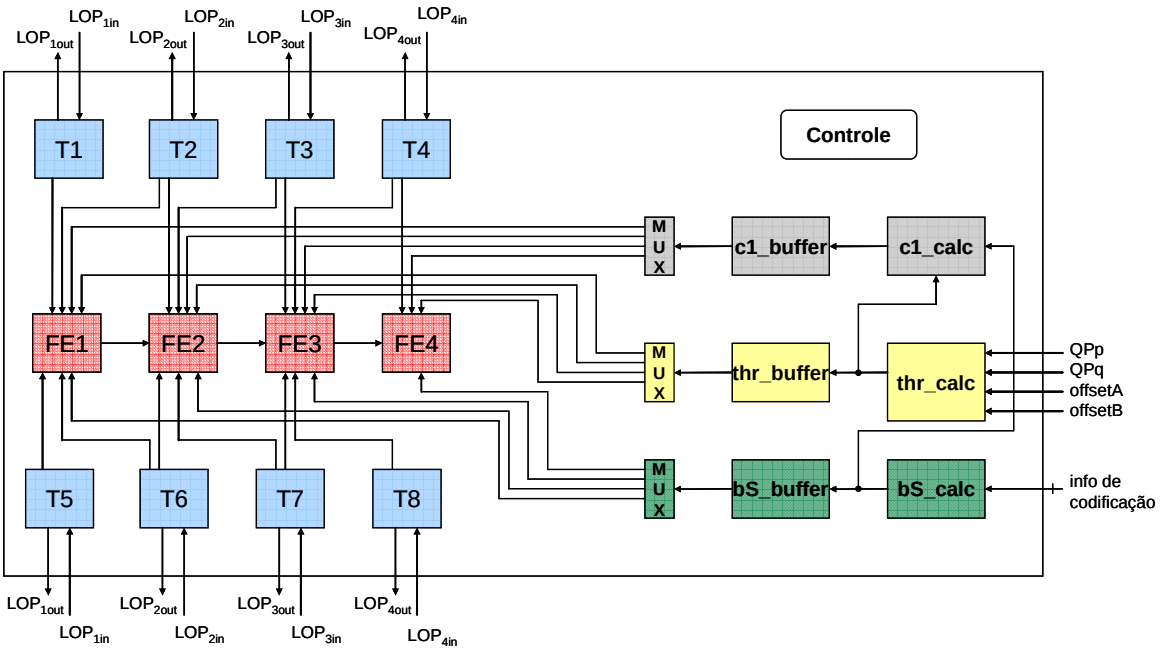

Figura 26 – Arquitetura completa do Filtro Redutor de Efeito de Bloco Intercamadas

5.2.7 Controle do Filtro

A Fig. 26 também apresenta um módulo de controle da arquitetura, que, embora não apareça interligado a nenhum outro módulo para simplificação do diagrama, gera os sinais de ativação e seleção, *flags* e endereços de memória

mencionados nas seções anteriores deste capítulo. O módulo de controle está, portanto, conectado a todos os outros módulos desta arquitetura, incluindo os multiplexadores, registradores e *buffers*.

A ideia básica do funcionamento da arquitetura está em uma estrutura de *pipeline* que executa as filtragens concorrentemente. Desta forma, a filtragem de cada bloco consome 7 ciclos, embora uma linha de blocos (4 blocos 4x4) possa ser filtrada em 10 ciclos, conforme ilustra a Fig. 27. Nos primeiros quatro ciclos, as quatro primeiras LOPs dos quatro primeiros blocos de um macrobloco são escritas nas matrizes de transposição. No segundo ciclo, a leitura das segundas LOPs ocorre em paralelo com os cálculos de limites e bS correspondentes à primeira borda (entre o macrobloco à esquerda e o primeiro bloco do macrobloco). No terceiro ciclo, a leitura das terceiras LOPs acontece em paralelo com os cálculos de limites e do bS para a segunda borda e com o cálculo de c_1 da primeira borda. A partir do quarto ciclo, as filtragens começam a ocorrer no primeiro núcleo e, em cada um dos três ciclos seguintes, um novo núcleo de filtragem é habilitado. Na Fig. 27, os quadros em laranja indicam as leituras de cada linha de um MB, os quadros azuis indicam os cálculos de limites e bS, os quadros em cinza indicam os cálculos de c_1 e os quadros em rosa indicam as filtragens das bordas entre os blocos, realizadas sempre entre duas LOPs de blocos vizinhos.

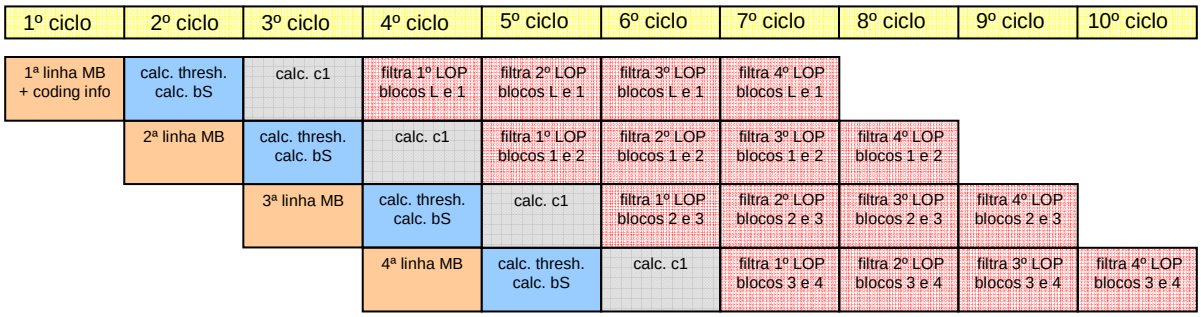

Figura 27 – Fluxo de execução da filtragem horizontal dos primeiros quatro blocos

As máquinas de estados que representam o funcionamento de cada uma das filtragens estão apresentadas na Fig. 28. As quatro máquinas de estados são exatamente iguais, embora os estados de cada máquina estejam atrasados em uma unidade em relação à filtragem anterior. Por exemplo, na primeira filtragem (a), a primeira leitura de LOP acontece no estado 0, enquanto que na segunda filtragem

(b), a primeira leitura de LOP acontece no estado 1. Os sinais *ctrl1*, *ctrl2*, *ctrl3* e *ctrl4* ativam cada uma das filtragens nos ciclos 1, 2, 3 e 4, respectivamente.

Com exceção da *flag* que define se uma filtragem deve ocorrer no sentido horizontal ou vertical, todos os outros sinais podem ser gerados a partir da máquina de estados da Fig. 28. Para a definição do sentido de filtragem, contudo, um sinal externo ao controle é utilizado para controlar o número de ocorrências de filtragens em um sentido, pois após o início de quatro filtragens em um sentido, as próximas quatro filtragens que tiverem início devem ocorrer sentido contrário (vide Fig. 15).

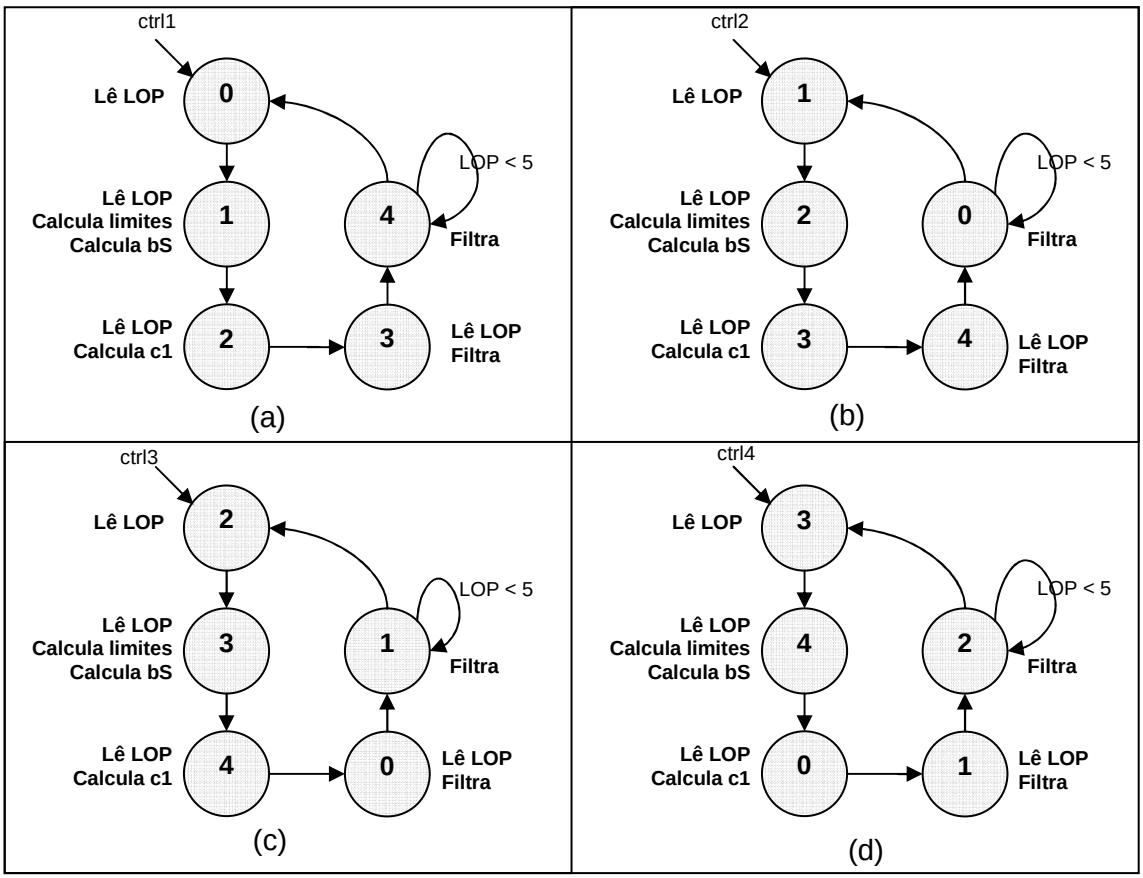

Figura 28 – Máquinas de estados correspondentes a cada uma das filtragens

5.3 Resultados Obtidos e Comparação com Outros Trabalhos

Antes da síntese, foi realizada uma análise do número de ciclos necessários para a filtragem de um macrobloco completo com a arquitetura desenvolvida. Como se pode perceber na Fig. 15, o processo de filtragem de um macrobloco completo (luminância, crominância azul e crominância vermelha) está completo após 53 ciclos (35 ciclos para luminância e 9 para cada crominância).

Em (ERNST, 2007), uma arquitetura que também possui quatro núcleos de filtragem e utiliza a ordem de processamento apresentada por (LI, 2005), é proposta. Esta arquitetura realiza a filtragem, em nível de blocos, de um macrobloco completo em 75 ciclos. Desta forma, a arquitetura proposta aqui apresenta uma diminuição de 25% no número de ciclos consumidos para a filtragem.

Apesar de não utilizarem quatro núcleos de filtragem, outros trabalhos são comparados com este na Tab. 1. Como se pode perceber, dentre todas as propostas, este trabalho utiliza o menor número de ciclos para filtrar um macrobloco completo. Além disso, a arquitetura proposta utiliza pouco mais da metade da quantidade de memória necessária na arquitetura proposta em (ERNST, 2007).

Tabela 1 – Comparação entre o número de ciclos utilizados na filtragem de um macrobloco completo, número de núcleos e tamanho da memória

Autor	**# ciclos**	**# núcleos**	**tamanho da memória**
(JVT, 2003)	192	1	512 bytes
(KHURANA, 2006)	192	1	128 bytes
(SHENG, 2004)	192	1	80 bytes
(LI, 2005)	140	2	112 bytes
(ERNST, 2007)	75	4	224 bytes
Este trabalho	**53**	**4**	**128 bytes**

A arquitetura apresentada foi sintetizada para o dispositivo FPGA EP3SL50F484C2 da família Stratix III da Altera (ALTERA, 2009). Os módulos foram sintetizados separadamente para que fosse possível realizar a validação de cada um. A síntese para a arquitetura completa também foi realizada.

A Tab. 2 mostra os resultados de síntese em termos de ALUTs combinacionais e registradores dedicados do FPGA utilizadas por cada módulo do Filtro para o dispositivo FPGA da família Stratix III. Como esperado, o núcleo de filtragem é o módulo que utilizou a maior quantidade de elementos lógicos. A quantidade de cálculos realizados por este módulo faz com ele se torne o componente com o atraso crítico da arquitetura (aproximadamente 10,5 ns).

Tabela 2 – Recursos lógicos do dispositivo EP3SL50F484C2 (Stratix III) utilizados por cada módulo do Filtro

Módulo	**# ALUTs comb.**	**# registradores dedicados**
Matriz de Transposição	199	206
Calculador de limites	60	0
Calculador de bS	7	0
Calculador de c_1	38	0
Núcleo de Filtragem	737	0

A Tab. 3 mostra os resultados de síntese em termos de ALUTs combinacionais e registradores dedicados do FPGA utilizadas pela arquitetura completa do Filtro para o dispositivo FPGA da família Stratix III.

Tabela 3 – Recursos lógicos do dispositivo EP3SL50F484C2 (Stratix III) utilizados pela arquitetura completa do Filtro

Recurso lógico	**Quantidade**
ALUTs combinacionais	7868
Registradores Dedicados	206

A ferramenta Quartus II não consegue gerar resultados de análise de *timing* totalmente confiáveis para o dispositivo escolhido, pois os modelos de *timing* ainda estão em desenvolvimento. O primeiro resultado de freqüência apresentado na Tab. 4 foi gerado a partir do modelo *Slow 900mV 85C* e o segundo resultado a partir do modelo *Slow 900mV 0C*. A Tab. 4 mostra os resultados de síntese em termos frequência máxima de operação do Filtro, segundo os dois modelos, para o dispositivo FPGA EP3SL50F484C2 da família Stratix III.

Tabela 4 – Frequência máxima de operação do Filtro para o dispositivo EP3SL50F484C2 (Stratix III)

Modelo de *timing*	**Frequência (MHz)**
Slow 900 mV 85C Model	265,67
Slow 900 mV 0C Model	288,52

Apesar de os resultados de análise de *timing* serem preliminares, eles são bastante satisfatórios. Tomando-se uma frequência de operação em torno de 270 MHz (valor entre o apresentado pelos dois modelos), e considerando-se uma

resolução HDTV (1920 x 1080 pixels), a taxa de processamento do Filtro projetado ficaria em torno de 623 quadros por segundo, o que satisfaz com folga a taxa mínima de 24 a 30 quadros por segundo, que é necessária para processamento dos vídeos em tempo real em resoluções elevadas.

6 CONCLUSÕES E TRABALHOS FUTUROS

Esta monografia relatou o estudo realizado acerca da codificação de vídeo escalável com foco na extensão SVC do padrão H.264/AVC e a implementação de um subsistema do seu codificador/decodificador. Após a identificação dos mecanismos necessários à implementação da escalabilidade, passou-se a um estudo aprofundado do bloco escolhido como alvo deste trabalho, o Filtro Redutor de Efeito de Bloco Intercamadas, definido na extensão SVC. Para isso, foi necessário um estudo prévio em torno do funcionamento do Filtro Redutor de Efeito de Bloco do padrão H.264/AVC (não escalável) a partir de artigos que apresentam o seu funcionamento, da versão preliminar do documento que padroniza a extensão SVC e do software de referência do padrão. Após isso, passou-se ao estudo das ordens de processamento de amostras já existentes para o Filtro, visto que esta é forma mais usual de lidar com o problema de utilização de memória e paralelismo de operações do filtro.

A principal contribuição deste trabalho é a proposta de uma ordem de processamento baseada em filtragens concorrentes em nível de amostras, ao contrário dos trabalhos anteriores que vinham propondo a concorrência em nível de blocos. Foi mostrado, no capítulo 5, que a ordem de processamento proposta reduziu em 25% a quantidade de ciclos necessários para a filtragem de um macrobloco completo, em comparação com um trabalho equivalente, com o mesmo número de núcleos de filtragem que a arquitetura proposta. Considerando outra arquitetura presente na literatura, com dois núcleos de filtragem, os ganhos da solução proposta neste trabalho chegam a 62% em termos de redução no número de ciclos utilizados. Finalmente, em relação às arquiteturas com um único núcleo de filtragem, a redução no número de ciclos ficou em torno de 72% com o uso da solução desenvolvida neste trabalho.

A arquitetura desenvolvida foi dividida em módulos calculadores conforme o software de referência do padrão divide os seus cálculos em métodos, a fim de

facilitar uma futura validação da arquitetura com base em dados recolhidos a partir da execução do software. Posteriormente, foi projeta a arquitetura completa com a ligação de todos os módulos e com o projeto do módulo de controle, considerando a ordem de processamento definida no final do capítulo 4.

A arquitetura foi descrita em VHDL e sintetizada para dispositivo FPGA das famílias Stratix III da Altera. A síntese, realizada através do software Quartus II da Altera, foi seguida pela análise de *timing* dos modelos Slow 900mV 85C e Slow 900mV 0C.

Entre os resultados de síntese obtidos para a arquitetura completa do filtro, chegou-se a uma frequência de operação máxima entre 265 e 288 MHz para os dois modelos de timing utilizados pela ferramenta. Os resultados de síntese confirmaram a ideia original de que o núcleo de filtragem seria o ponto crítico da arquitetura, apresentando a maior quantidade de elementos lógicos utilizados e o maior atraso, devido ao número de operações aritméticas em sequência. Contudo, os resultados da análise de *timing* mostraram que a arquitetura é capaz de processar até 623 quadros por segundo para uma resolução alta (HDTV – 1920x1080 pixels), um resultado que fica muito acima dos requisitos necessários para o processamento em tempo real (24 a 30 quadros por segundo).

Na continuação deste trabalho, pretende-se realizar a validação completa da arquitetura através da ferramenta ModelSim (MENTOR, 2009). Pretende-se extrair os dados anteriores e posteriores à aplicação do Filtro Redutor de Efeito de Bloco Intercamadas e compará-los aos dados de entrada e saída da arquitetura, respectivamente.

Além disso, pretende-se realizar a prototipação desta arquitetura e a sua integração com o *upsampling* para a formação de um bloco de Predição Intra Intercamadas.

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.

ERNST, E. **Architecture Design of a Scalable Adaptive Deblocking Filter for H.264/AVC**. Dissertação de mestrado. Rochester, New York, 2007

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.

KHURANA, G; KASSIM, T; CHUA, T., MI, M. **A pipelined hardware implementation of In-loop Deblocking Filter in H.264/AVC**. IEEE Transactions on Consumer Electronics, 2006.

LI, L; GOTO, S; IKENAGA, T. **A highly parallel architecture for deblocking filter in H.264/AVC**. IEICE Transactions on Information and Systems, 2005.
List, P.; Joch, A.; Lainema, J.; Bjontegaard, G.; Karczewicz, M. **Adaptive Deblocking Filter** (IEEE Trans. on Circuits and Systems for Video Tech), 2003.

MENTOR Graphics, “Mentor Graphics: The EDA Technology Leader”. Disponível em: www.mentor.com. Acesso em: janeiro, 2009.

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.

SHENG, B; GAO, W; WU, D. **An Implemented Architecture of Deblocking Filter for H.264/AVC**. Proceedings - International Conference on Image Processing, ICIP, 2004.

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

TOME, T; et al. **Abordagem Sistêmica no Sistema Brasileiro de Televisão Digital (SBTVD)**. Disponível em: <http://cpqd.com.br/file.upload/07_artigoforum_sbtvd.pdf>. Acesso em: abril 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

Best Poster Award, 1º Lugar, Chip in the Pampa – SForum 2008, Gramado - Brasil, 2008.

Prêmio Jovem Pesquisador 3º 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

DORNELLES, Robson; SAMPAIO, Felipe; PALOMINO, Daniel; CORRÊA, Guilherme; NOBLE, Diego; AGOSTINI, Luciano Volcan. **Arquitetura de um Módulo T Dedicado à Predição Intra do Padrão de Compressão de Vídeo H.264/AVC para Uso no Sistema Brasileiro de Televisão Digital**. Hífen, 2008, Uruguaiana, Brasil.

A.2.2 Trabalhos completos em anais de eventos

CORRÊA, Guilherme; MESQUITA, Eduardo; FRANCK, Helen; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Evaluating Fault-Tolerant Fast Adders Implemented in FPGAs**. In: 4th Southern Conference on Programmable Logic, 2008, Bariloche, Argentina.

BRAGA, Matheus P.; CORRÊA, Guilherme; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Sensitivity Analysis to SETs Considering Timing and Logic Masking.** In: 9th IEEE Latin-American Test Workshop, 2008, Puebla, Mexico.

CORRÊA, Guilherme; REDIESS, Fabiane; AGOSTINI, Luciano Volcan; 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, Brasil.

BRAGA, Matheus P.; CORRÊA, Guilherme; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **A Two-Step SET Sensitivity Estimation Technique**. In: XXIII Simpósio Sul de Microeletrônica – SIM2008, 2008, Bento Gonçalves, Brasil.

CORRÊA, Guilherme; FRANCK, Helen; MESQUITA, Eduardo; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Fault-Tolerant Fast Adders Implemented in**

FPGAs. In: XXIII Simpósio Sul de Microeletrônica – SIM2008, 2008, Bento Gonçalves, Brasil.

CORRÊA, Guilherme; MESQUITA, Eduardo; FRANCK, Helen; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Arquitetura de Somador de Alto Desempenho Baseada no Recálculo Parcial com Carry Invertido**. In: XXXIV Conferencia Latinoamericana de Informática, 2008, Santa Fé, Argentina.

PALOMINO, Daniel; CORRÊA, Guilherme; DORNELLES, Robson; SAMPAIO, Felipe; NOBLE, Diego; AGOSTINI, Luciano. **Implementation and Analysis of Architectures for the 4x4 2-D Forward Hadamard Transform of H.264/AVC**. In: XXXIV Conferencia Latinoamericana de Informática, 2008, Santa Fé, Argentina.

SAMPAIO, Felipe; DORNELLES, Robson; PALOMINO, Daniel; CORRÊA, Guilherme; NOBLE, Diego; AGOSTINI, Luciano Volcan. **Architectural Templates for the 4X4 Transforms of the H.264/AVC Standard targeting the Intra Prediction Coder**. In: Students Forum on Microelectronics - SForum 2008, 2008, Gramado, Brasil.

CORRÊA, Guilherme; FRANCK, Helen; MESQUITA, Eduardo Macedo; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Set-Tolerant Fast Adders Implemented in FPGAs**. In: Students Forum on Microelectronics - SForum 2008, 2008, Gramado, Brasil.

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

MESQUITA, Eduardo; CORRÊA, Guilherme; BRAGA, Matheus Porciuncula ; FRANCK, Helen; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Reducing TMR Resource Overhead in Hardened Carry-Select Adders**. In: 8th IEEE Latin-American Test Workshop, 2007, Cuzco, Peru.

BRAGA, Matheus Porciuncula; CORRÊA, Guilherme; MATEUS, Gustavo; VICOSA JR., Elvio; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Timing Aware Pre-Analysis of Single Event Transient Propagation**. In: XXII Simpósio Sul de Microeletrônica, 2007, Porto Alegre, Brasil.

CORRÊA, Guilherme; PETRY, Rafael; GÜNTZEL, José Luís; AGOSTINI, Luciano Volcan. **A Comparison Between Multiplier Architectures Implemented in Altera FPGAs**. In: XXI Simpósio Sul de Microeletrônica, 2006, Porto Alegre, Brasil.

A.2.3 Resumos em anais de eventos

REDIESS, Fabiane; CORRÊA, Guilherme; AGOSTINI, Luciano Volcan. **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 da UFPel, 2008, Pelotas, Brasil.

CORRÊA, Guilherme; BRAGA, Matheus Porciuncula; GUNTZEL, José Luís; AGOSTINI, Luciano Volcan. **Análise Automática da Propagação de Single-Event Transients Utilizando os Algoritmos PODEM e SAT**. In: XVI Congresso de Iniciação Científica da UFPel, 2007, Pelotas, Brasil.

CORRÊA, Guilherme; MESQUITA, Eduardo; FRANCK, Helen; GÜNTZEL, José Luís; AGOSTINI, Luciano Volcan. **Análise da Proteção de Somadores Tolerantes a Radiação Implementados em FPGAs**. In: XVI Congresso de Iniciação Científica da UFPel, 2007, Pelotas, Brasil.

BRAGA, Matheus Porciuncula; CORRÊA, Guilherme; GÜNTZEL, José Luís Almada. **Análise da Propagação de Single-Event Transients em Circuitos Combinacionais Implementados em FPGAs**. In: XVIII Salão de Iniciação Científica da UFRGS, 2006, Porto Alegre, Brasil.

CORRÊA, Guilherme; BRAGA, Matheus Porciuncula; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Análise Automática da Propagação de Single-Event Transients em Circuitos Combinacionais Implementados em FPGAs**. In: XV Congresso de Iniciação Científica da UFPel, 2006, Pelotas, Brasil.

BRAGA, Matheus Porciuncula; CORRÊA, Guilherme; AGOSTINI, Luciano Volcan; GÜNTZEL, José Luís. **Avaliação de Arquiteturas de Somadores Tolerantes à Radiação Implementados em FPGAs**. In: XV Congresso de Iniciação Científica da UFPel, 2006, Pelotas, Brasil.

A.3 Trabalhos aceitos para publicação ou sob avaliação

CORRÊA, Guilherme; AGOSTINI, Luciano Volcan; CRUZ, Luís. **Filtro Redutor de Efeito de Bloco Intercamadas do Padrão H.264/AVC Escalável**. XV Workshop Iberchip 2009, Buenos Aires, Argentina (**ACEITO**).

CORRÊA, Guilherme; AGOSTINI, Luciano Volcan; CRUZ, Luís. **A fast FPGA implementation of the inter-layer deblocking filter for H.264/SVC**. 7th Conference on Telecommunications – ConfTele 2009, Santa Maria da Feira, Portugal (**SUBMETIDO**).