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

RICIÉRI CLAYTON DALLA ROSA SILVA

PROJETO DE DIPLOMAÇÃO

FILTRO REDUTOR DE EFEITO DE BLOCO PARA H.264

Porto Alegre
2010

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

FILTRO REDUTOR DE EFEITO DE BLOCO PARA H.264

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

ORIENTADOR: Altamiro Amadeu Susin

Porto Alegre
2010

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

RICIÉRI CLAYTON DALLA ROSA SILVA

FILTRO REDUTOR DE EFEITO DE BLOCO PARA H.264

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

Orientador: _____

Prof. Altamiro Amadeu Susin, UFRGS

Formação Institut National Polytechnique – Grenoble, França

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique – Grenoble, França

Prof. Dr. Marcelo Götz, UFRGS

Doutor pela Universität Paderborn – Paderborn, Alemanha

Prof. Dr. Marcelo Soares Lubaszewski, UFRGS

Doutor pelo Institut National Polytechnique – Grenoble, França

Porto Alegre, dezembro de 2010.

DEDICATÓRIA

Dedico este trabalho aos meus pais, Sirlei e Jair Silva, em especial pela dedicação e apoio em todos os momentos difíceis.

AGRADECIMENTOS

Agradeço aos meus pais, pela confiança depositada e pelo apoio recebido ao longo de todos esses anos.

À minha namorada Gleici Fleck, pela compreensão nos momentos ausentes e pelo apoio quando necessário.

Aos colegas de curso pelo companheirismo e auxílio nas etapas difíceis pelas quais passamos ao longo do curso.

Gostaria de agradecer ao Professor Dr. Altamiro Susin pela confiança depositada como orientador desse projeto e pelos conhecimentos passados, ao Dr. André Borin Soares e ao Eng. M.Sc Alexsandro Cristóvão Bonatto pelo suporte fornecido, esclarecendo dúvidas quando solicitado.

Aos colegas de trabalho, que sempre me motivaram e me apoiaram quando não podia dar prioridade para o mesmo.

À Universidade, professores, funcionários pelos conhecimentos adquiridos e pelo ambiente propiciado para a aprendizagem.

RESUMO

Este trabalho tem o objetivo de estudar e implementar em VHDL um filtro redutor de efeito de bloco para um codificador de vídeo de acordo com o padrão H.264. Será apresentado o estudo teórico do comportamento e funcionamento do filtro, incluindo definições, operações matemáticas e regras do procedimento. A proposta de arquitetura e os detalhes de implementação, assim como a descrição de funcionamento de cada componente, também serão discutidos. O resultado final da síntese deve ser capaz de filtrar em tempo real vídeos em resolução HD (1920x1080@30).

Palavras-chaves: Engenharia Elétrica. Filtro redutor de efeito de bloco. H.264. Codificador de vídeo. VHDL.

ABSTRACT

This document aims to study and to implement in VHDL a deblocking filter for a video coder according to H.264 standard. Will be presented a theorical study of the filter behavior, including definitions, math operations and rules of procedure. The architecture propose and the implementation details, as the operation of each component, also will be discussed. The final synthesis results must be able to filter real-time high resolution videos (1920x1080@30).

Keywords: Electrical Engineering. Deblocking filter. H.264. Video coder. VHDL.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AVC: Advanced Video Coding

BS: Boundary Strength

DCT: Discrete cosine transform

HDTV: High-definition television

IDCT: Inverse discrete cosine transform

IEC: International Electrotechnical Commission

ISO: International Organization for Standardization

ITU: International Telecommunication Union

LOP: Line-of-Pixel

QP: Quantization Parameter

RGB: Red, Green and Blue

1 INTRODUÇÃO

Com a evolução dos dispositivos eletrônicos, é cada vez mais freqüente encontrar equipamentos que apresentem entre suas características a geração e a reprodução de vídeo. Com isso, o desenvolvimento de melhores codificadores teve grande crescimento nos últimos anos. Além de buscar melhor a qualidade no vídeo codificado, foi preciso encontrar uma solução para diminuir a grande quantidade de dados gerados nesse processo, e principalmente, conseguir enviar todos esses dados através da banda de transmissão disponível.

Analisando uma imagem de vídeo no formato RGB e HDTV (1920 x 1080), com 8 bits para cada cor, um quadro possui 49766400 bits. Considerando que são exibidos 30 quadros por segundo, seria necessário 1,5 Gigabits/s para ser possível exibir o vídeo de forma correta e em tempo real. Nesse ponto que entram os codificadores de vídeo, cuja função é comprimir o vídeo para taxas mais baixas de transmissão, perdendo o mínimo de qualidade possível. Para isso, os codificadores possuem sua base de funcionamento nos altos índices de redundância de informação que os vídeos apresentam após serem digitalizado. Os codificadores funcionam considerando três tipos de redundância: espacial (redundância dentro do mesmo quadro), temporal (redundância entre quadros) e entrópica (redundância que explora a probabilidade de ocorrência dos símbolos).

O codificador de vídeo H.264/AVC é o mais novo padrão aprovado pela ITU (Recommendation H.264) e pela ISO/IEC (MPEG-4 part 10, AVC) e foi desenvolvido em conjunto pelas duas organizações. Apresenta muitas melhorias em relação aos padrões anteriores, como melhor compressão, melhor qualidade de imagem e, como conseqüência, exige menos taxas de transmissão.

A figura 1 mostra os principais blocos de um codificador H.264, que são: a estimação de movimentos (ME), a compensação de Movimentos (MC), a Predição Intra Quadros, as Transformadas e Quantização diretas e inversas, o Filtro Redutor de Efeito de Bloco e a Codificação de Entropia. A ME e a MC juntas formam a Predição Inter Quadros.

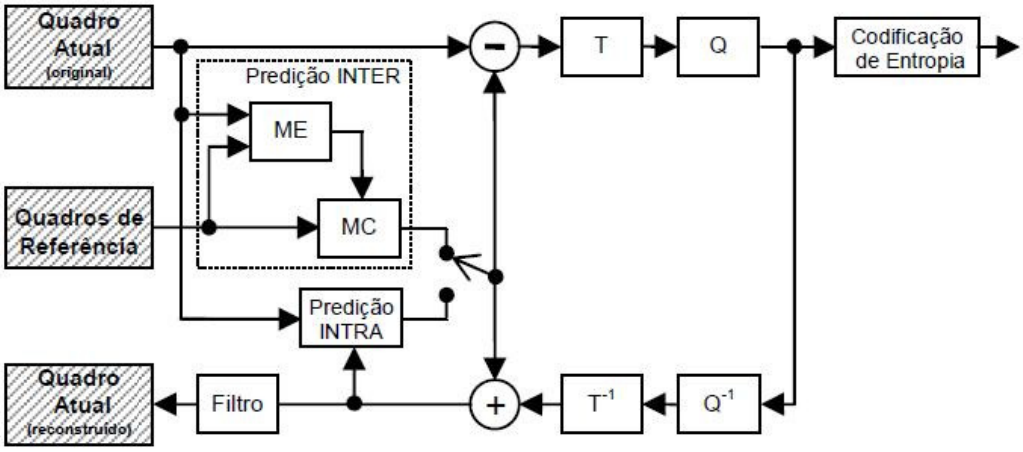


Figura 1 Diagrama de Blocos do processo de codificação.

O bloco das transformadas e quantização, juntamente com a predição de movimento, gera os artefatos de bloco. Para resolver esse problema, o padrão especifica o uso do filtro redutor de efeito de bloco. Na figura 2, é possível perceber duas imagens do mesmo quadro do vídeo, porém, em na imagem da esquerda ainda não foi realizada a filtragem, e o efeito de bloco está bem caracterizado. Já na imagem da direita, após a filtragem, percebe-se o quanto o filtro conseguiu suavizar as bordas entre os blocos. O filtro atua em todas as bordas entre os blocos que fazem parte de um macrobloco. Os fatores que influenciam na força do filtro e quais blocos serão filtrados serão descritos com mais detalhes nos capítulos seguintes.

Figura 2 Exemplo de imagem sem (esquerda) e com filtro (direita).

É notável a diferença entre as duas imagens. Com o avanço das transmissões em alta definição, os consumidores estão mais exigentes em relação à qualidade da imagem. Assistir a um programa com uma imagem carregada de blocos visíveis se tornaria um incômodo. É desse ponto que surge a motivação para a realização desse trabalho: prover uma qualidade de imagem seguindo os padrões H.264.

2 FILTRO REDUTOR DE EFEITO DE BLOCO

O processo de filtragem é aplicado para todas as bordas de um bloco 4x4 na imagem, exceto nos blocos das margens e nos blocos em que a filtragem foi desabilitada. A figura 3 mostra um macrobloco formado a partir 16 blocos de luminância (4x4), onde o bloco C é o bloco superior esquerdo do macrobloco. Perceba que a coluna a qual o bloco A está inserida pertence ao macrobloco da esquerda. O mesmo vale para a linha que o bloco B está inserido. A figura evidencia a filtragem vertical entre os blocos A e C, e a filtragem horizontal entre os blocos B e C.

De acordo com o padrão H.264, existe uma ordem definida para ocorrerem às filtragens. Para blocos de luminância, primeiro são filtradas as bordas mais da esquerda, seguindo pelas bordas verticais em direção à direita. Do mesmo modo, as bordas horizontais superiores são filtradas primeiro. E deve-se respeitar o fato de que antes de se realizar uma filtragem horizontal, garantir que ambas as bordas verticais do bloco já tenham passadas pelo processo de filtragem. Isso é devido ao fato de que o resultado das operações realizadas na borda vertical será usado para processamento dos sinais na borda horizontal. Por exemplo, o pixel Q1 da figura 3 será usado na filtragem vertical entre o bloco A e C, e o resultado dessas operações gerará um novo valor para Q1. Este novo valor será usado no processamento da outra borda vertical, entre os blocos D e C. Após isso acontecer, será possível avançar o bloco C adiante, para que se possa realizar a operação horizontal entre ele e o bloco B. A última borda do bloco C que será filtrada está localizada entre o bloco C e o bloco E. Se forem filtradas as quatro bordas do bloco, ele não será mais necessário no filtro. Porém, se uma das bordas fizer fronteira com um outro macrobloco que ainda não foi filtrado, faltando assim pelo menos uma borda para finalizar o processamento, este bloco deverá ser encaminhado para uma unidade de memória, onde aguardará a chegada do macrobloco necessário para que

todas as bordas sejam finalizadas. Esse comportamento é observado no bloco A, por exemplo, que fazia parte de um outro macrobloco e apenas após ter a borda que faz divisa com o bloco C filtrada, poderá ser encaminhado para a saída.

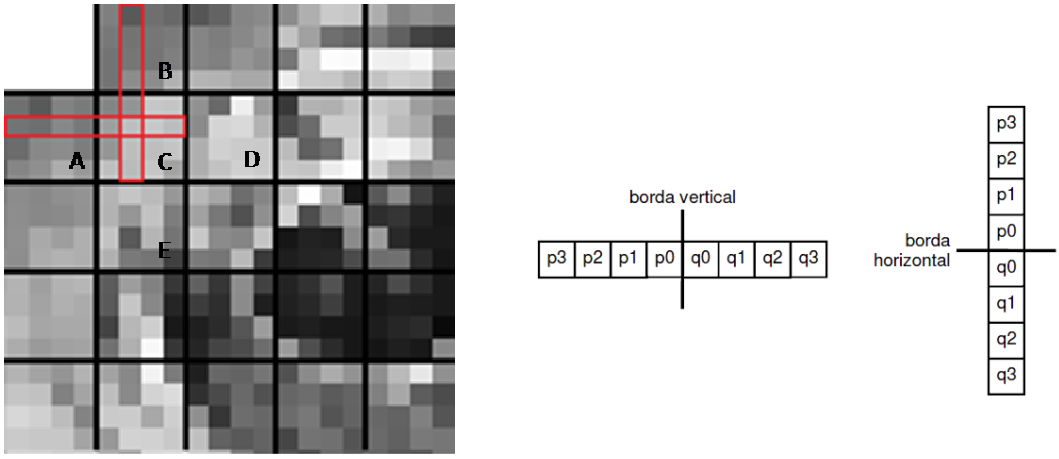


Figura 3 Macrobloco de luminância e detalhe da linha e coluna marcadas em vermelho.

A figura 4 mostra as bordas que são filtradas nos macroblocos de luminância e de crominância. São quatro bordas verticais e quatro horizontais para luminância, e duas bordas verticais e duas horizontais para cada elemento de crominância.

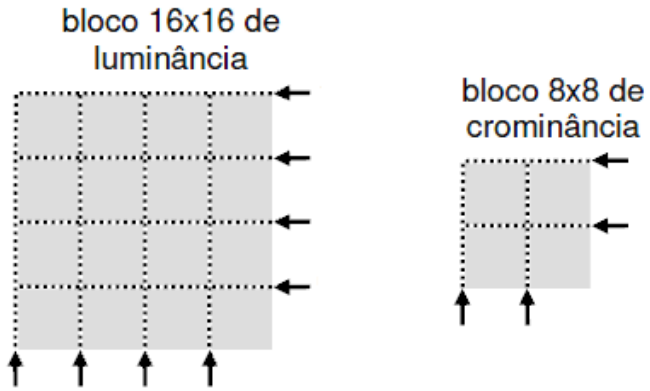


Figura 4 Bordas usadas pelo filtro.

Para evitar que uma borda real da imagem seja confundida com uma borda que precise ser filtrada, o filtro redutor de efeito de bloco é adaptativo em três níveis: nível de slice, nível de borda de bloco e nível de amostra.

Nível de Slice

No nível de slice, a força de filtragem global pode ser ajustada para características individuais de seqüência de vídeo.

São definidos dois thresholds, α e β, para determinar se a borda será filtrada ou não. O usuário pode controlar dois parâmetros, α-offset e β-offset, que servem para ajustar α e β. Quanto maior o valor desses dois parâmetros, mais bordas são filtradas. Além disso, pode-se redefinir o parâmetro de quantização QP e alterar a qualidade da imagem. Se QP possuir um valor baixo, então α e β também terão valores baixos. Neste caso, o efeito do bloco gerado tem importância menor e, por isso, o filtro permanecerá inativo durante mais tempo. Por outro lado, se QP possuir um valor elevado, então a quantização gerará mais perdas, gerando efeitos de bloco mais significativos. Neste caso, os valores de α e β serão maiores, fazendo com que o filtro fique mais tempo ativo e, assim, mais amostras da borda são filtradas (AGOSTINI, 2007).

Nível de Borda de Bloco

Para cada borda entre blocos de luminância 4x4, uma força de filtragem é escolhida. Esse parâmetro é conhecido como *Boundary Strength* (BS). O valor de BS pode variar de 0 até 4. Se BS tiver valor 4, significa que o filtro estará com força máxima. Para valores menores, o filtro é cada vez mais fraco. Se BS tiver valor 0,

significa que nenhuma filtragem será aplicada a borda. A tabela 1 mostra os fatores que determinam o valor de BS.

Para blocos de crominância, não é calculado um novo valor de BS para filtrar as bordas. Nesse caso, o valor calculado para os blocos de luminância é usado para filtrar as bordas de crominância.

Tabela 1 – Configuração do valor de BS

Condições	BS
Um dos blocos é intra e é borda externa de um macrobloco	4
Um dos blocos é intra	3
Um dos blocos tem coeficientes codificados	2
Diferença de vetores de movimento maior que uma amostra de luminância	1
Compensação de movimento de diferentes quadros de referência	1
Outro	0

Nível de Amostra

No filtro removedor de efeito de bloco, é importante distinguir entre bordas reais na imagem e aquelas criadas pela quantização dos coeficientes da DCT. Para preservar a nitidez da imagem, as bordas reais devem ser deixadas sem atuação do filtro tanto quanto possível, enquanto são filtradas apenas as bordas artificiais. Borda real é aquela que realmente faz parte da imagem, por exemplo, um bloco preto ao lado de um bloco vermelho, assim como era na imagem original. As seguintes equações devem ser verdadeiras para que ocorra a filtragem. Se uma delas for falsa, significa que se trata de uma borda real e nenhuma filtragem é aplicada.

$$|p0 - q0| < \alpha \quad (2.1)$$

$$|p1 - p0| < \beta \quad (2.2)$$

$$|q1 - q0| < \beta \quad (2.3)$$

2.1 DEFINIÇÕES INICIAIS DA FILTRAGEM

Existem algumas variáveis que controlam o funcionamento do filtro. Um delas é responsável por informar se a borda que está sendo filtrada é entre blocos de luminância ou crominância. De acordo com a norma ITU – Recommendation H.264, essa variável é chamada de chromaEdgeFlag, e possuindo valor 1, significa que se trata de um bloco de crominância. Se possuir valor 0, se trata de um bloco de luminância. É necessário saber o valor dessa variável antes de iniciar o processo de filtragem.

Outra variável que precisa ser determinada antes de iniciar a filtragem é a média do fator de quantização entre dois blocos adjacentes, QP_{av}. Se os dois blocos que serão filtrados pertencem ao mesmo macrobloco, significa que ambos possuem o mesmo parâmetro de quantização. Apenas se a filtragem estiver ocorrendo na borda do macrobloco, os parâmetros de quantização podem ser diferentes. QP_{av} pode ser calculado da seguinte maneira:

$$QP_{av} = \frac{(QP_p + QP_q + 1)}{2}, \tag{2.4}$$

onde QP_p é o valor de QP no bloco localizado a esquerda (filtragem vertical) ou localizado acima (filtragem horizontal), e QP_q é o valor de QP no bloco localizado a direita(filtragem vertical) ou localizado abaixo(filtragem horizontal). O valor de QP, de acordo com o padrão H.264,, é um inteiro cujo valor está entre 0 e 51.

Cada QP contém o parâmetro de quantização para luminância, QP_Y, e o parâmetro de quantização para crominância, QP_C. O valor de QP_C é obtido através da tabela 2, e depende de QP_Y. Se a filtragem ocorrer em blocos de luminância, então o valor de QP_Y é usado em QP_p e QP_p para calcular o QP_{av}. Mas se os blocos forem de crominância, então é usado o valor de QP_C.

Tabela 2 - QP_C em função de QP_Y.

QP_Y	40	41	42	43	44	45	46	47	48	49	50	51
QP_c	36	36	37	37	37	38	38	38	39	39	39	39

QP_Y	< 30	30	31	32	33	34	35	36	37	38	39
QP_c	QP_Y	29	30	31	32	33	34	34	35	36	36

Após calcular QP_{av}, é possível obter duas outras variáveis: $Index_A$ e $Index_B$. Além de dependerem de QP_{av}, dependem também de $Offset_A$ e $Offset_B$, que são dois parâmetros controlados pelo usuário. $Index_A$ e $Index_B$ são obtidas da seguinte maneira:

$$Index_A = Min(Max(0, QP_{av} + Offset_A), 51) \quad (2.5)$$

$$Index_B = Min(Max(0, QP_{av} + Offset_B), 51) \quad (2.6)$$

Com $Index_A$ e $Index_B$ calculados, é possível obter agora os fatores α e β. Esses fatores podem ser obtidos através da tabela 3. Esses dois fatores têm uma importância muito grande no processo de filtragem e desempenham diversas funções. Participam da etapa que diferencia bordas reais das bordas geradas pela quantização, e também são usados em algumas situações condicionais que determinam quais operações serão realizadas no processo de filtragem.

De acordo com a tabela 3, é possível observar que α e β aumentam com $Index_A$ e $Index_B$, e que α tem crescimento maior que β. O valor de α varia de 0 a 255, enquanto o valor de β varia de 0 a 18.

Tabela 3 - α em função de $Index_A$ e β em função de $Index_B$.

	INDEX A (Para α) INDEX B (Para β)												
	0	1	2	3	4	5	6	7	8	9	10	11	12
α	0	0	0	0	0	0	0	0	0	0	0	0	0
β	0	0	0	0	0	0	0	0	0	0	0	0	0

	INDEX A (Para α) INDEX B (Para β)												
	13	14	15	16	17	18	19	20	21	22	23	24	25
α	0	0	0	4	4	5	6	7	8	9	10	12	13
β	0	0	0	2	2	2	3	3	3	3	4	4	4

	INDEX A (Para α) INDEX B (Para β)												
	26	27	28	29	30	31	32	33	34	35	36	37	38
α	15	17	20	22	25	28	32	36	40	45	50	56	63
β	6	6	7	7	8	8	9	9	10	10	11	11	12

	INDEX A (Para α) INDEX B (Para β)												
	39	40	41	42	43	44	45	46	47	48	49	50	51
α	71	80	90	101	113	127	144	162	182	203	226	255	255
β	12	13	13	14	14	15	15	16	16	17	17	18	18

Tabela 4 - T_{C0} em função de $Index_A$ e BS

	INDEX A												
	0	1	2	3	4	5	6	7	8	9	10	11	12
BS=1	0	0	0	0	0	0	0	0	0	0	0	0	0
BS=2	0	0	0	0	0	0	0	0	0	0	0	0	0
BS=3	0	0	0	0	0	0	0	0	0	0	0	0	0

	INDEX A												
	13	14	15	16	17	18	19	20	21	22	23	24	25
BS=1	0	0	0	0	0	0	0	0	0	0	1	1	1
BS=2	0	0	0	0	0	0	0	0	1	1	1	1	1
BS=3	0	0	0	0	1	1	1	1	1	1	1	1	1

	INDEX A												
	26	27	28	29	30	31	32	33	34	35	36	37	38
BS=1	1	1	1	1	1	1	1	2	2	2	2	3	3
BS=2	1	1	1	1	1	2	2	2	2	3	3	3	4
BS=3	1	2	2	2	2	3	3	3	4	4	4	5	6

	INDEX A												
	39	40	41	42	43	44	45	46	47	48	49	50	51
BS=1	3	4	4	4	5	6	6	7	8	9	10	11	13
BS=2	4	5	5	6	7	8	8	10	11	12	13	15	17
BS=3	6	7	8	9	10	11	13	14	16	18	20	23	25

Quando a filtragem ocorre com BS possuindo um valor menor que 4, é necessário uma outra variável para que o processo seja realizado com sucesso. Essa variável é t_{C0}, e depende do valor de $Index_A$ e de BS. De acordo com a tabela 4, é possível obter t_{C0}. Essa variável t_{C0} é usada para calcular t_C, que também depende de a_p, a_q e de β.

$$a_p = |p_2 - p_0| \tag{2.7}$$

$$a_q = |q_2 - q_0| \tag{2.8}$$

Para calcular t_C, é necessário observar se a borda está localizada entre blocos de luminância ou de crominância. Para blocos de luminância, t_C é calculado da seguinte maneira:

$$t_C = t_{C0} + ((a_p < \beta)?1:0) + ((a_q < \beta)?1:0) \tag{2.9}$$

A função $((a_p < \beta)?1:0)$ tem a seguinte definição: se $a_p < \beta$ for verdadeiro, retorna 1. Se for falso, retorna 0. O mesmo vale para $((a_q < \beta)?1:0)$. Por exemplo, se as duas equações forem verdadeiras $t_C = t_{C0} + 1 + 1$, se apenas uma for verdadeira $t_C = t_{C0} + 1$, e se ambas forem falsas $t_C = t_{C0}$.

Para calcular o valor de t_C para bordas entre blocos de crominância, o cálculo é mais simples:

$$t_C = t_{C0} + 1 \tag{2.10}$$

Para limitar o valor de algumas variáveis calculadas, usa-se t_C. E, como as operações de filtragem para BS igual a 1, 2 ou 3, é o valor de t_C que diferencia uma filtragem da outra, pois possui valores diferentes para cada um desses valores de BS.

2.2 OPERAÇÕES MATEMÁTICAS DA FILTRAGEM

Há dois modos de filtragem baseado no parâmetro BS para um conjunto de amostras. Um deles é um modo especial de filtragem que aplica uma força maior ocorre quando BS é igual a 4. O outro, é um modo comum de filtragem e é aplicado quando BS tem valor 1, 2 ou 3. Independente do modo de filtragem aplicado, o valor do fator β é usado para uma melhor avaliação em duas outras condições de atividades espaciais que são aplicadas para determinar a amplitude da filtragem em casos de amostras de luminância. Essas duas condições são $a_p < \beta$ e $a_q < \beta$. São essas as condições que determinam quantos pixels serão alterados e como e quanto serão alterados. Posteriormente será discutido melhor como atuam essas duas condições.

2.2.1 BS MENOR QUE 4

Inicialmente, será tratado sobre as filtragens com o fator BS menor que 4. A partir desse ponto calcula-se o valor de Δ através de dois passos. O primeiro passo é calcular o valor de Δ_{0i} da seguinte maneira:

$$\Delta_{0i} = \frac{(4 \cdot (q_0 - p_0) + (p_1 - q_1) + 4)}{8} \tag{2.11}$$

O segundo passo é aplicar o valor de Δ_{0i} na seguinte equação para obter o valor de Δ.

$$\Delta = Min(Max(-t_C, \Delta_{0i}), t_C) \tag{2.12}$$

Com o valor de Δ calculado, o próximo passo é calcular os termos p_0' e q_0'. As equações para obter p_0' e q_0' são muito semelhantes. Esses dois valores são obtidos através de:

$$p_0' = Min(Max(0, p_0 + \Delta), 255) \tag{2.13}$$

$$q_0' = Min(Max(0, q_0 - \Delta), 255) \tag{2.14}$$

Se as amostras que estão passando pelo processo de filtragem são amostras de crominância e a força do filtro for menor que 4, o processo termina nesse ponto. Porém, se forem amostras de luminância, há mais um passo a seguir, que determina se mais alguns pixels serão filtrados.

Para continuar a filtragem para valores de BS menor que 4 para amostras de luminância, é necessário que as condições que envolvem β sejam satisfeitas. Essas condições foram tratadas anteriormente, e são $a_p < \beta$ e $a_q < \beta$. Isso influencia nos pixels p_1 e q_1. Se a condição $a_p < \beta$ for satisfeita, o pixel p_1 terá seu valor alterado. Se a condição $a_q < \beta$ for satisfeita, o pixel q_1 terá seu valor alterado. Se apenas uma das condições for satisfeita, apenas o pixel relacionado a ela será alterado, o outro permanecerá com seu valor original. Se nenhuma das condições for satisfeita, nem p_1 e q_1 sofrerão mudanças. Vale ressaltar que os outros pixels da amostra (p_2, p_3, q_2 e q_3) não sofrem alterações na filtragem que possui BS com valor menor que 4.

Se a condição para p_1 for satisfeita, seu novo valor é calculado usando a seguintes equações:

$$\Delta_{p1i} = \frac{(p_2 + \frac{(p_0 + q_0 + 1)}{2} - 2 \cdot p_1)}{2} \tag{2.15}$$

Aplica-se então Δ_{p1} em:

$$\Delta_{p1} = Min(Max(-t_{C0}, \Delta_{p1i}), t_{C0}) \tag{2.16}$$

Finalmente, para chegar ao valor final de p_1', realiza-se o cálculo:

$$p_1' = p_1 + \Delta_{p1} \tag{2.17}$$

Para q_1, o mesmo procedimento é realizado. Calcula-se inicialmente o valor Δ_{q1i}:

$$\Delta_{q1i} = \frac{(q_2 + \frac{(q_0 + p_0 + 1)}{2} - 2 \cdot q_1)}{2} \tag{2.18}$$

Aplica-se então Δ_{q1i} em:

$$\Delta_{q1} = Min(Max(-t_{C0}, \Delta_{q1i}), t_{C0}) \tag{2.19}$$

E por último, calcula-se q_1':

$$q_1' = q_1 + \Delta_{q1} \tag{2.20}$$

2.2.2 BS IGUAL A 4

Se o valor de BS for igual a 4, então o filtro estará atuando com força máxima e até três pixels por bloco podem sofrer alteração. O primeiro passo é checar se as amostras pertencem a blocos de luminância ou crominância. Se forem amostras de luminância, então outras condições dever ser analisadas para determinar qual procedimento será aplicado nos pixels. Para as amostras do bloco P, as condições são as seguintes, se tratando de blocos de luminância:

$$a_p < \beta \;\&\&\; |p_0 - q_0| < \frac{\alpha}{4} + 2 \tag{2.21}$$

Se essas condições forem verdadeiras, então as seguintes operações são realizadas:

$$p_0' = \frac{(p_2 + 2 \cdot p_1 + 2 \cdot p_0 + 2 \cdot q_0 + q_1 + 4)}{8} \tag{2.22}$$

$$p_1' = \frac{(p_2 + p_1 + p_0 + q_0 + 2)}{4} \tag{2.23}$$

$$p_2' = \frac{(2 \cdot p_3 + 3 \cdot p_2 + p_1 + p_0 + q_0 + 4)}{8} \tag{2.24}$$

Porém, se uma das condições não for satisfeita apenas o pixel p_0 sofre alteração de acordo com a equação:

$$p_0' = \frac{(2 \cdot p_1 + p_0 + q_1 + 2)}{4} \tag{2.25}$$

O mesmo ocorre para amostras do bloco Q de luminância. O que diferencia nesse caso são as condições que determinam as operações de filtragem, que são as seguintes:

$$a_q < \beta \;\&\&\; |p_0 - q_0| < \frac{\alpha}{4} + 2 \tag{2.26}$$

Se a condição acima for verdadeira, então três pixels sofrerão alterações de acordo com as equações que seguem:

$$q_0' = \frac{(q_2 + 2 \cdot q_1 + 2 \cdot q_0 + 2 \cdot p_0 + p_1 + 4)}{8} \tag{2.27}$$

$$q_1' = \frac{(q_2 + q_1 + q_0 + p_0 + 2)}{4} \tag{2.28}$$

$$q_2' = \frac{(2 \cdot q_3 + 3 \cdot q_2 + q_1 + q_0 + p_0 + 4)}{8} \tag{2.29}$$

Da mesma forma que acontece no bloco P, se uma das condições não for satisfeita apenas uma amostra do bloco Q recebe um novo valor, no caso, q_0.

$$q_0' = \frac{(2 \cdot q_1 + q_0 + p_1 + 2)}{4} \tag{2.30}$$

Quando a filtragem está relacionada a amostras de crominância, as operações realizadas são as mesmas que ocorrem quando as condições usadas para filtrar amostras de luminância não são atingidas, ou seja,

$$p_0' = \frac{(2 \cdot p_1 + p_0 + q_1 + 2)}{4} \text{ e } q_0' = \frac{(2 \cdot q_1 + q_0 + p_1 + 2)}{4}. \tag{2.31 e 2.32}$$

A figura 5 mostra o fluxograma que resume todo o processo de filtragem descrito até esse momento.

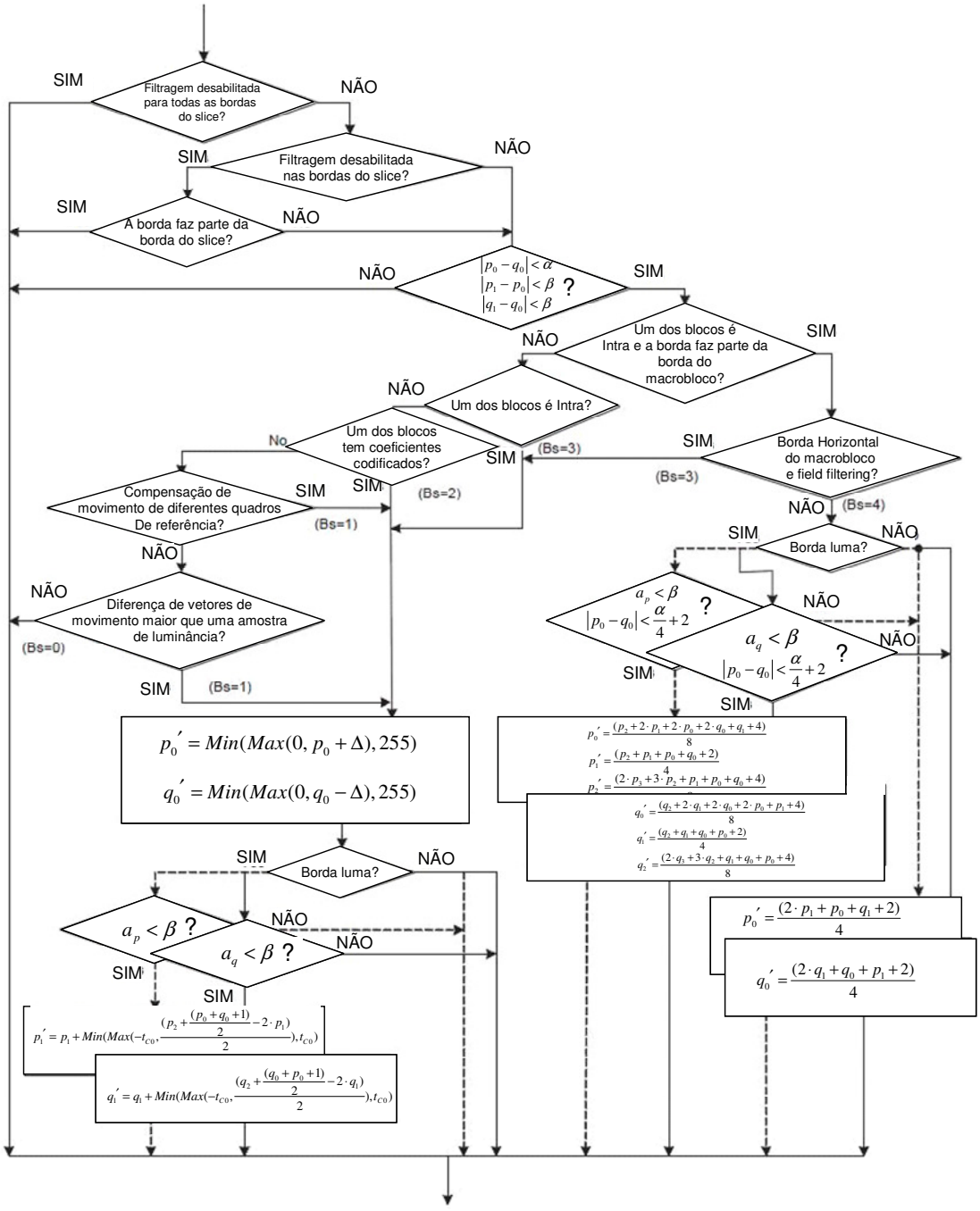


Figura 5 Fluxograma do processo de filtragem.

3 ANÁLISE DE ALTERNATIVAS

Inicialmente, foram tomadas algumas decisões de como proceder na implementação. Uma delas foi dividir o sistema em pequenos blocos, com a finalidade de melhor organizar o projeto ao invés de criar um grande arquivo contendo todas as operações. O uso dos packages também foi uma das alternativas encontradas para declarar tabelas, declarar componentes e funções matemáticas comumente usadas. Outro detalhe foi a escolha de tratar a arquitetura dos arquivos de duas formas: comportamental e estrutural. Os arquivos do tipo comportamental são aqueles que possuem no seu código descrição de funcionamento. Já os arquivos do tipo estrutural são usados para conectar os componentes criados entre si.

Ao invés de refazer o bloco responsável pelo processamento horizontal, optou-se por replicar o bloco que faz as filtragens verticais, já que as mesmas operações acontecem nos dois modos de filtragem.

A ordem de chegada de cada bloco do macrobloco e a prioridade de processamento das bordas verticais foram os fatores que levaram a escolha da arquitetura que será implementada.

Como a proposta é desenvolver em VHDL, para aplicações em FPGA, é preciso ter cuidado com a arquitetura escolhida. Desenvolver um projeto que realiza todas as operações em poucos ciclos de clock é uma abordagem que visa velocidade de processamento, porém acaba pagando em área da FPGA utilizada. Quanto menos ciclos de clock forem necessários para executar determinado processamento, maior será a área consumida. Em contrapartida, executando o processamento em muitos ciclos de clock, usa-se menos área. Como a proposta é que o filtro possa ser usado em um codificador com capacidade de trabalhar em tempo real, é preciso encontrar uma arquitetura que consiga conciliar área usada com velocidade de processamento.

A figura 6 mostra a ordem que chegarão os blocos, numerados de 0 a 15. Essa ordem é conhecida como Duplo-Z. As letras *a*, *b*, *c* e *d*, representam os blocos verticais do macrobloco da esquerda, e que deverão ser usados na filtragem. E as letras *e*, *f*, *g* e *h* representam os blocos horizontais do macrobloco superior.

	e	f	g	h
a	0	1	4	5
b	2	3	6	7
c	8	9	12	13
d	10	11	14	15

Figura 6 Ordem de chegada dos blocos.

O primeiro bloco a chegar é o bloco 0. Partindo do fato que o bloco *a* já está armazenado, a primeira borda já pode ser filtrada (entre *a* e 0). O próximo bloco a chegar será o bloco 1. Ocorre agora o processamento da borda vertical entre o bloco 0 e o bloco 1. O bloco *e* também já estava armazenado e o bloco 0 já passou pelas duas filtragens verticais. Já é possível fazer o processamento da borda horizontal entre esses dois blocos. Analisando a ordem de chegada dos blocos, percebe-se que o próximo bloco a ter suas duas bordas verticais filtradas é o bloco 2, já que o bloco 1 depende do bloco 4 para finalizar o processamento vertical. Então, é possível fazer a última filtragem do bloco 0, que é a horizontal entre ele e o bloco 2. Sendo assim, o bloco 0 terá todas as suas bordas filtradas e já pode ser enviado para a saída do filtro.

Pensando nisso, foi proposta a arquitetura que está na figura 7.

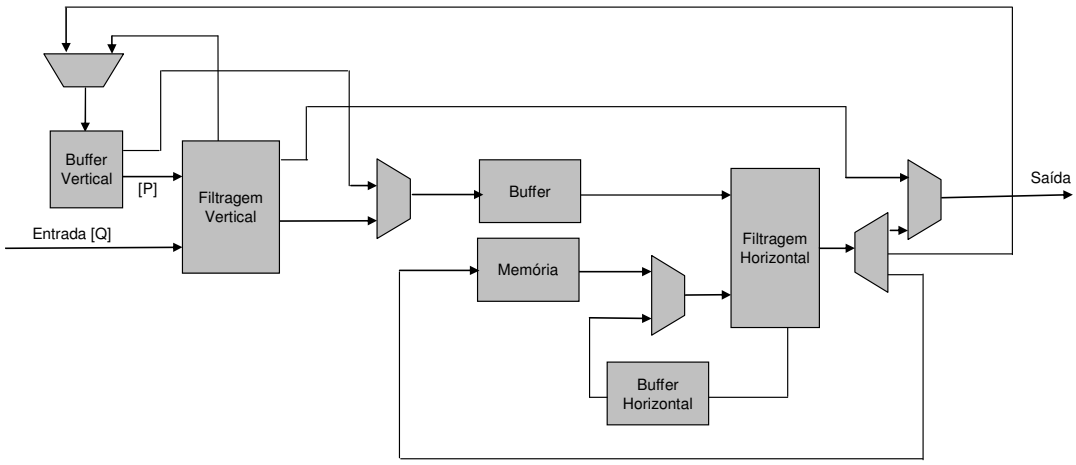


Figura 7 Arquitetura proposta.

4 IMPLEMENTAÇÃO E DESENVOLVIMENTO

4.1 FILTRO COMPLETO

A análise do filtro pode ser divida em um bloco que realiza a filtragem vertical, um bloco que realiza a filtragem horizontal e 3 pequenos blocos que reorganizam os dados de maneira transposta.

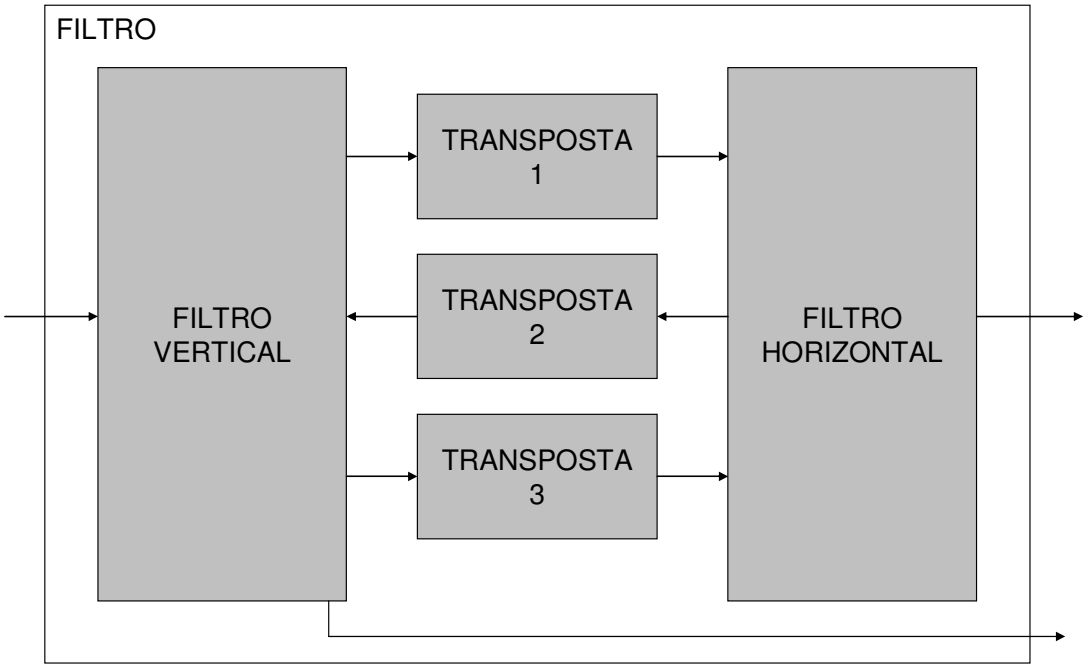


Figura 8 Análise do fluxo de dados no filtro.

A ordem de filtragem de cada bloco depende de sua posição no macrobloco, e está descrita na figura 9. Inicialmente os dados passam pela filtragem vertical. Ao terem as duas bordas verticais filtradas, passam para a transposta 1, onde são reorganizados e enviados para o filtro horizontal. Ao terem as duas bordas horizontais filtradas, são enviados para a saída. Porém, alguns blocos precisam de um tratamento especial. Os blocos 5, 7, 13 e 15 são enviados para o filtro horizontal com apenas uma borda vertical filtrada, pois devem respeitar a ordem de filtragem da figura 9. Os blocos 5, 7 e 13

passam por duas filtragens horizontais e são enviados de volta para o filtro vertical pela transposta 2, para que possam ter sua segunda borda vertical filtrada. Assim que isso ocorre, são enviados para a saída do filtro. O bloco 15 ao chegar ao filtro horizontal, passa por apenas uma filtragem, e é enviado de volta para a filtro vertical através da transposta 2. Ao ter sua segunda borda vertical filtrada, é enviado novamente para o filtro horizontal através da transposta 3 para ter sua última borda filtrada e ser enviado para a saída.

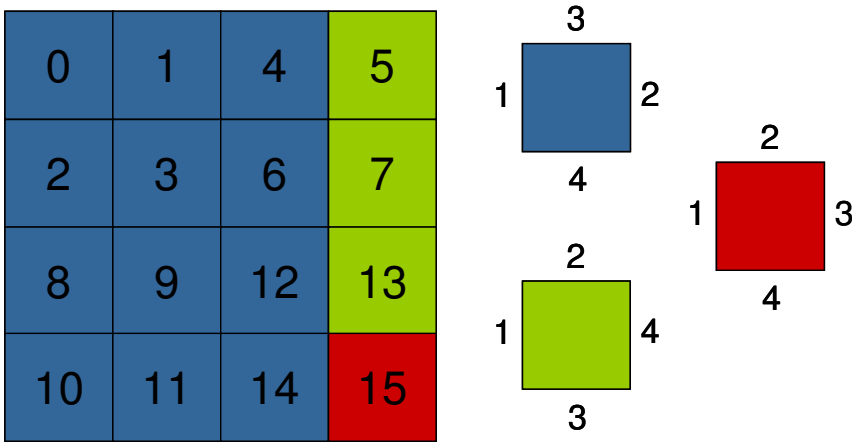


Figura 9 Ordem de filtragem.

O bloco Transposta 1, além de enviar as linhas de pixels do filtro vertical para o horizontal, leva também as informações de cada bloco, como o valor de QP, dos vetores de movimento, endereço do bloco, tipo de predição, indicador de coeficientes da IDCT e frame de referência. Os blocos Transposta 2 e Transposta 3 apenas enviam as linhas de pixels e endereço do bloco, já que as outras informações já estão armazenadas nos seus respectivos destinos.

Este arquivo é do tipo estrutural, ou seja, seu código VHDL apenas conecta os cinco componentes da figura 8.

O tempo total para a filtragem de um macrobloco é de 124 ciclos de relógio.

4.2 FILTRO VERTICAL

O filtro vertical é o bloco que contém todos os componentes que participam da filtragem das bordas verticais. Recebe como entrada a linha de pixels que sofrerá filtragem, um sinal que avisa que há entrada válida e as informações usadas no processo de filtragem, como o parâmetro QP, tipo de predição, vetores de movimento, coeficientes da IDCT e frames de referência. Também recebe como entrada os dados que voltam do filtro horizontal para o filtro vertical.

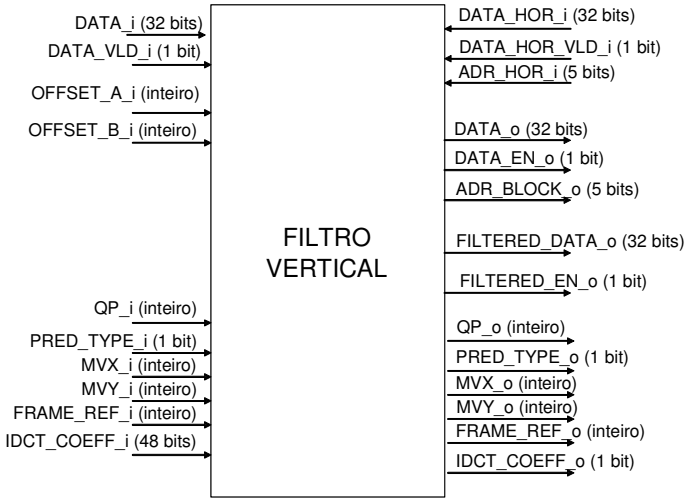


Figura 10 Bloco responsável pela filtragem vertical.

Quando há um dado válido no filtro vertical, o primeiro passo é analisar os coeficientes da IDCT. Atua nesse ponto o verificador de coeficientes da IDCT (bloco DCT na figura 10), que analisa se há algum coeficiente diferente de zero ou não. Enquanto ocorre essa etapa, paralelamente são gerados os endereços pelo gerador de endereços (bloco ADR na figura 10), e são buscados no buffer vertical (bloco BUFF) os dados necessários para o processamento. Também são buscadas nessa etapa as informações do bloco P armazenadas no bloco responsável por essa função (bloco INF).

Com os todos os valores de P e Q prontos, o próximo passo é obter o valor de BS, cujo responsável por essa função é o analisador de força do filtro (bloco BS). Com o valor de BS obtido, calculam-se as definições iniciais no bloco de mesmo nome (bloco INIT), que são enviadas ao bloco responsável por realizar as operações de filtragem (bloco FILT).

A última etapa de processamento é realizar as a filtragem em si. Depois disso, de acordo com o bloco, ele é enviado para o processamento das bordas horizontais ou é enviado para a saída. E o bloco que passou pela primeira filtragem vertical é armazenado pelo buffer vertical. Ao enviar os dados para passarem pela filtragem horizontal, é necessário enviar junto todas as informações do bloco, como QP, vetores de movimento, etc. A figura 11 mostra o fluxo de dados desse processo.

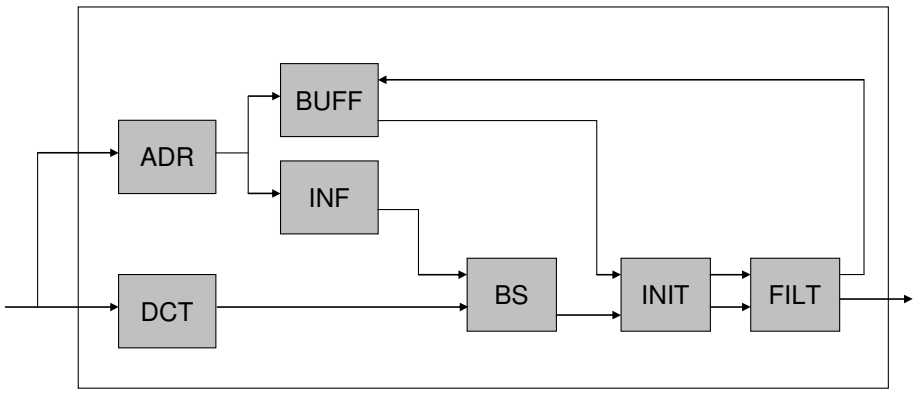


Figura 11 Processo de filtragem vertical.

O arquivo criado em VHDL para o filtro vertical é do tipo estrutural, e é responsável por conectar os componentes da figura 11. Além disso, possui alguns processos declarados no seu código. Um desses processos é o gerador de delays, que garante que algum sinal não chegasse antes do momento em que seria usado. O filtro vertical também possui alguns multiplexadores usados no gerenciamento de sinais. Um desses multiplexadores é responsável por determinar quando o filtro vertical está

enviando para a saída um dado válido. De acordo com as regras de filtragem, os blocos 5, 7, 13, 17 e 21 tem sua última borda filtrada no filtro vertical. Desse modo, a saída FILTERED_EN_o recebe o valor 1 quando um desses blocos está saindo do filtro.

O tempo entre chegar um dado válido e este ter uma de suas bordas verticais processadas é de 6 ciclos de relógio.

O funcionamento de cada um desses componentes, tais como suas entradas e saídas é descrito nos próximos tópicos.

4.3 FILTRO HORIZONTAL

O filtro horizontal é o bloco que contém todos os componentes que participam do processo de filtragem das bordas horizontais. Está ligado no filtro vertical através de blocos que reorganizam os dados de maneira transposta.

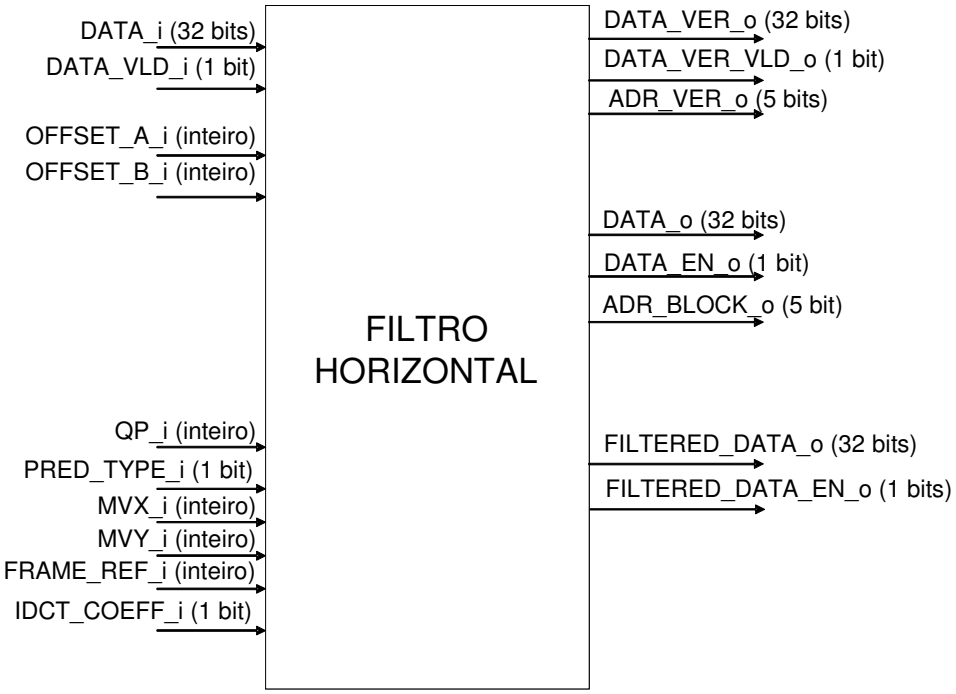


Figura 12 Bloco responsável pela filtragem horizontal.

Seu funcionamento se assemelha muito com o filtro vertical. Porém se aproveitam de alguns processamentos realizados na filtragem anterior para evitar processamentos redundantes. Por exemplo, ao invés de usar um novo gerador de endereços, o valor de endereço do bloco chega como uma entrada para o filtro horizontal. Outro valor que não necessita ser obtido e chega como uma entrada é o sinal que informa a existência ou não de coeficientes diferentes de zero da DCT inversa.

A principal diferença entre o processamento horizontal e vertical está na forma de armazenamento de dados que serão necessários no futuro. O filtro vertical armazena dados por um período curto de tempo, apenas até que outro macrobloco chegue, seja processado e substitua os dados antigos pelos novos no buffer. Já no filtro horizontal, é necessário armazenar os dados até que toda uma linha de macroblocos sofra a filtragem horizontal, e ao iniciar o processamento de uma nova linha de macroblocos, os dados sejam recuperados, processados e substituídos. Nesse caso, optou-se por usar RAMs parametrizáveis. Essas RAMs armazenam tanto os dados da linha de pixels quanto as informações usadas para obter o valor de BS.

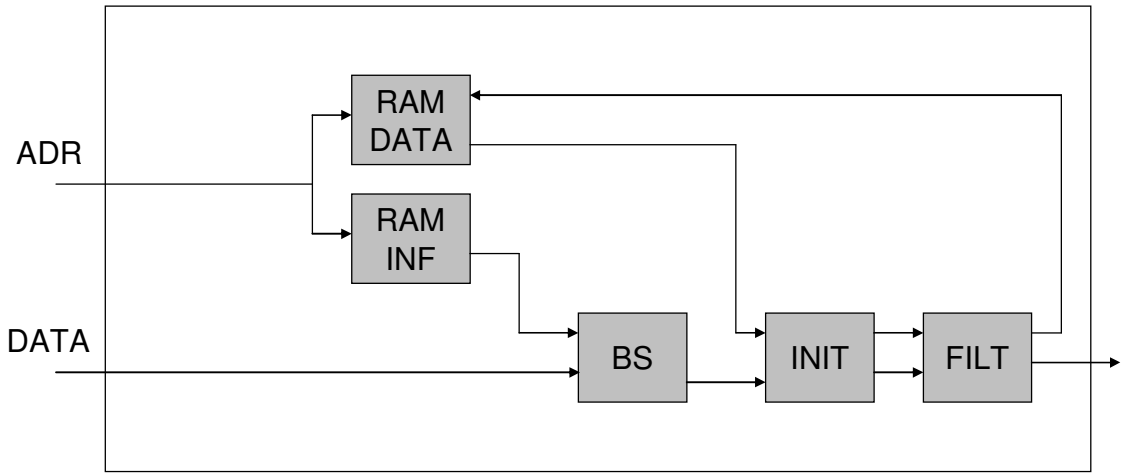


Figura 13 Fluxo de dados no filtro horizontal.

No filtro horizontal, ao chegar o endereço, este busca nas RAMs as informações para obtenção de BS e os dados da linha de pixels para processamento. Este endereçamento é feito de duas formas. Usa o endereço do bloco para escolher qual das RAMs será acessada e o endereço do macrobloco para acessar a posição na RAM escolhida.

Os valores buscados das RAMs são enviados para o bloco analisador da força do filtro e obtém-se assim o valor de BS. No próximo ciclo, são calculadas as definições iniciais no bloco responsável por essa função. Com esses valores calculados, é feita o processamento das bordas horizontais no bloco que realiza as operações de filtragem.

Após a operação ser realizada o bloco que teve apenas uma borda horizontal filtrada é armazenado RAM, e só terminará o processo quando o bloco inferior a ele chegar para realizar a filtragem horizontal. Dependo do bloco que teve suas duas bordas horizontais filtradas, duas situações podem ocorrer: se já teve as quatro 4 bordas filtradas, é encaminhado para a saída, mas se teve apenas três bordas (uma vertical e duas horizontais) é enviado de volta ao filtro vertical.

O arquivo do filtro horizontal também é do tipo estrutural. Assim como o filtro vertical, o filtro horizontal também apresenta processos que geram delays que servem para alinhar os dados. Multiplexadores também foram usados para gerenciamentos, como no caso do gerenciamento das RAMs, onde sinal com endereço do bloco é o responsável por escolher a RAM de dados que será usada na filtragem, funcionando como um sinal de controle para um multiplexador. Ele escolhe de qual RAM é buscado o dado que será usado na filtragem horizontal e em qual RAM escreverá o resultado após a filtragem. Cada RAM é responsável por uma coluna de blocos. Por exemplo, a RAM 0 armazena os blocos 0, 2, 8 e 10, sendo o último mantido até que a o bloco 0 na linha inferior de macroblocos chegue na filtragem horizontal.

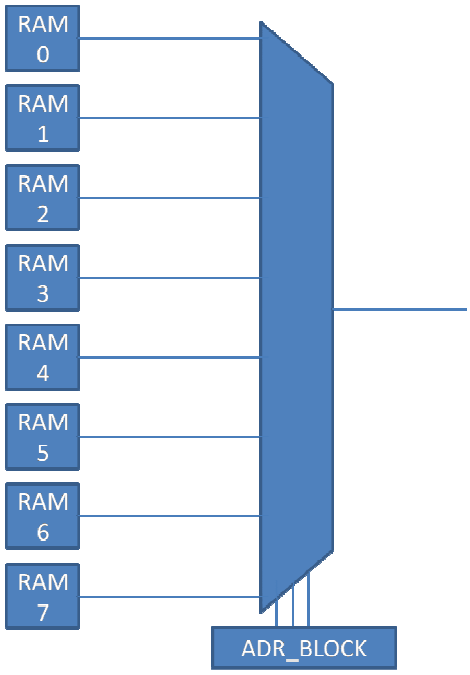


Figura 14 Multiplexador das RAMs.

O tempo entre a chegada de uma linha de pixels e o término da sua primeira filtragem é de 4 ciclos de clocks.

4.4 GERADOR DE ENDEREÇOS

Para que ocorra o processamento correto, é necessário saber qual bloco e qual macrobloco está chegando. Como essas informações não são fornecidas, foi criado um componente responsável por gerá-las. Só é possível gerar os endereços desse modo devido a ordem de chegada de cada bloco, que é conhecida. O arquivo criado para realizar esta tarefa é do tipo comportamental.

Além de gerar o endereço do bloco, ele gera o endereço da linha de pixels. Cada bloco possui quatro linhas de pixel. Ao chegar o primeiro dado válido, este recebe o endereço de bloco 0 e endereço de linha de pixels 0 do macrobloco 0.

Figura 15 Gerador de endereços.

Como cada macrobloco possui 24 blocos, e consequentemente 96 linhas de pixels, o gerador de endereços é um contador de 7 bits. Os cinco bits mais significativos indicam o bloco e os dois bits menos significativos qual das linhas do bloco está chegando.

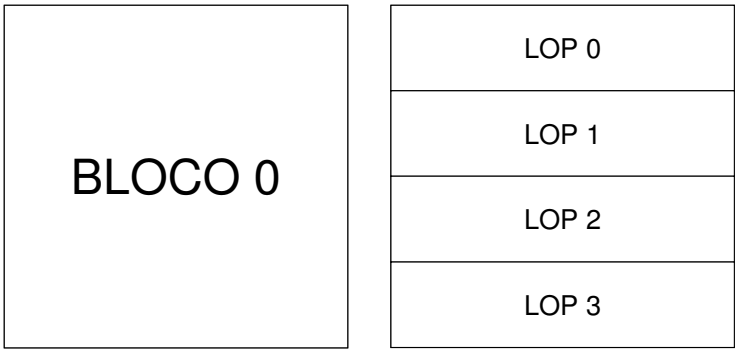


Figura 16 Bloco 0 e suas linhas de pixels.

Quando o contador chega ao valor 95, significa que está chegando um novo macrobloco e a contagem é reiniciada. Um segundo contador nesse ponto é incrementado. Esse segundo contador informa o endereço do macrobloco.

Como saída o gerador de endereços informa a o valor do bloco, da linha de pixel e do macrobloco. O processo leva um ciclo de relógio para cada contagem.

4.5 VERIFICADOR DE COEFICIENTES DA IDCT

Um dos fatores que influenciam na força da filtragem são os valores dos coeficientes da DCT inversa. Se o bloco que está passando pela filtragem possui pelo menos um valor da DCT inversa diferente de zero, todo o bloco será afetado. Assim como cada bloco possui quatro linhas de pixels, para a IDCT ocorre o mesmo, cada bloco também possui quatro linhas de coeficientes.

Figura 17 Verificador de coeficientes da IDCT.

Então, antes de fazer qualquer filtragem, o primeiro passo é verificar se o bloco que está chegando possui coeficientes da IDCT diferentes de zero. Essa etapa atrasa o processo em três ciclos de relógio, pois nenhum processamento pode ser feito sem esta informação.

O arquivo criado para descrever esse componente é do tipo comportamental. O processo funciona da seguinte forma: recebe-se no primeiro ciclo a primeira linha de coeficientes. Esta linha possui 48 bits que são armazenados em um sinal. No segundo ciclo, chega a segunda linha de coeficientes. Nesse ponto, ocorre a operação lógica OR dessa nova entrada com o valor da entrada anterior que está armazenado, e o resultado substitui o valor da primeira entrada no sinal onde ela estava guardada. No terceiro ciclo, chega a terceira linha de coeficientes e feito uma nova operação lógica OR entre a nova entrada e o resultado anterior. No quarto ciclo, chega a quarta e última linha de coeficientes e realiza-se a última operação lógica OR. Se o resultado final possuir todos

os bits iguais a zero, significa que esse bloco possui todos os seus coeficientes da DCT inversa igual a zero. Se algum bit do resultado for 1, significa que havia um coeficiente diferente de zero.

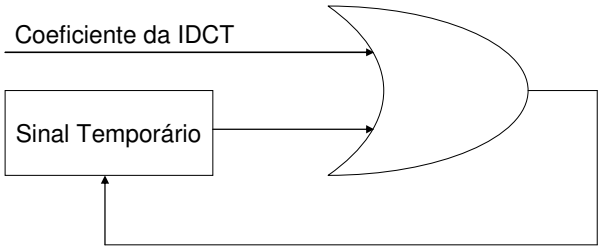


Figura 18 Operação lógica de checagem dos coeficientes da IDCT.

Para informar o resultado da operação, usou-se apenas um bit. Se todos os coeficientes eram zero, a saída recebia zero. Se não, recebia 1. Junto com a saída foi gerado um sinal que avisava o restante do processo que havia uma saída pronta e as operações que dependiam dessa informação poderiam ocorrer.

4.6 BUFFER VERTICAL

O buffer vertical é um bloco que armazena temporariamente os blocos que tiveram apenas uma das bordas verticais filtradas. De acordo com a figura 6, ao chegar um novo macrobloco para passar pelo processo de filtragem da borda vertical, inicialmente no buffer estarão os blocos *a*, *b*, *c* e *d*. O primeiro bloco a passar pela filtragem, será o bloco 0. Será feita a filtragem vertical do bloco 0 com o bloco *a*. Após terminar a filtragem entre esses dois blocos, o bloco *a* estará pronto para ser enviado para a saída. Sendo assim, o bloco 0, que nesse momento está apenas com uma borda vertical filtrada, assume o lugar do bloco *a* no buffer vertical. O bloco 0, por sua vez, será substituído pelo bloco 1, assim que tiver sua segunda borda vertical filtrada. As posições de cada bloco no buffer, assim como as substituições que ocorrem ao final de

cada filtragem, podem ser observadas na figura 19. Assim, é possível armazenar os blocos 0, 1, 4 e 5 na mesma posição do buffer.

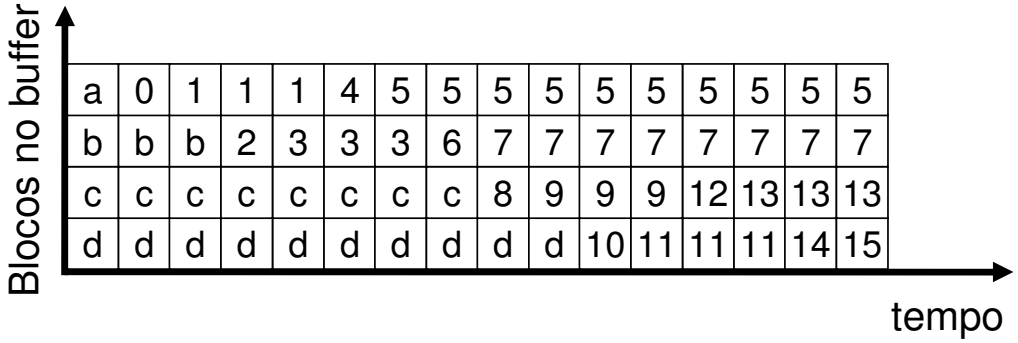


Figura 19 Demonstração de funcionamento do buffer vertical para os blocos de luminância.

O arquivo do buffer vertical é do tipo comportamental. Para realizar a implementação do buffer vertical partiu-se do princípio que chegariam 32 bits (linha de pixels) como entrada de dados e um sinal indicando que essa entrada é válida, além de saber a qual bloco essa linha pertence. Como cada bloco possui quatro linhas de pixels, o endereçamento da entrada possui sete bits. Os cinco bits mais significativos indicam o valor do bloco, enquanto os dois bits menos significativos são responsáveis por informar qual das quatro linhas do bloco pertence essa entrada.

Como o processamento acontece para 16 blocos de luminância e para 8 blocos de crominância, e cada bloco possui 4 linhas de pixels, o buffer vertical foi criado contendo 32 posições possíveis de 32 bits.

Ao chegar uma entrada válida, leva-se um ciclo de relógio para que ocorra o armazenamento.

A outra parte do funcionamento do buffer se refere à solicitação de dados que já estão armazenados. Para isso, apenas é necessário informar ao buffer qual bloco está chegando na entrada do processo. Por exemplo, quando o bloco 0 chega para realizar a

filtragem vertical, o buffer coloca na saída os dados que estão na primeira posição de armazenamento. O tempo para que essa etapa ocorra é de um ciclo de relógio.

Uma outra funcionalidade do buffer vertical é aproveitar os intervalos em que o não há saída válida para enviar para o filtro horizontal, e enviar os blocos que só tiveram uma borda vertical filtrada. Por exemplo, ao chegar o bloco 2 na filtragem vertical, há processamento de informações entre este bloco e o bloco 7 do macrobloco anterior. O bloco 2 terá apenas uma borda vertical filtrada e será enviado para o buffer. O bloco 7 terá passado pela sua última filtragem e será enviado para a saída. Sendo assim, aparece um intervalo que não há dados sendo enviados para o filtro horizontal. Nesse momento, o buffer vertical aproveita esse intervalo e envia blocos que só tiveram uma borda filtrada, como o bloco 5, 7 13 ou 15.

A partir dessa característica do sistema, criou-se uma segunda saída do buffer vertical que envia nesses intervalos os blocos 5, 7, 13 e 15.

4.7 BUFFER DE INFORMAÇÕES

Assim como o é necessário armazenar o valor da linha de pixel no buffer vertical, é preciso armazenar o valor de cada uma das informações de dos blocos em um outro componente, pois elas serão necessárias para calcular o valor de BS na segunda borda vertical desse bloco. Esse componente é o buffer de informações.

O funcionamento desse componente é muito semelhante ao buffer vertical, possuindo a mesma característica de ter uma saída especial que aproveita o intervalo onde não há dados válidos e envia junto com a linha de pixels, todas as outras informações necessárias para o processamento da filtragem horizontal.

4.8 ANALISADOR DE FORÇA DO FILTRO

Para que as operações de filtragem sejam realizadas de forma correta, é estabelecido pela norma um fator chamado responsável pela força de filtragem, chamado de. *boundary strength* (BS). Foi criado um arquivo do tipo comportamental.

Como a filtragem ocorre na borda entre dois blocos, é preciso saber algumas informações de ambos os blocos. Para definir o valor de BS é necessário saber o tipo de predição dos blocos (intraframe ou interframes), se os blocos possuem algum coeficiente da IDCT diferente de zero, quais os frames de referência dos blocos, os vetores de movimento (x e y) de cada bloco, se a filtragem está ocorrendo na borda do macrobloco e se está no modo progressivo ou entrelaçado.

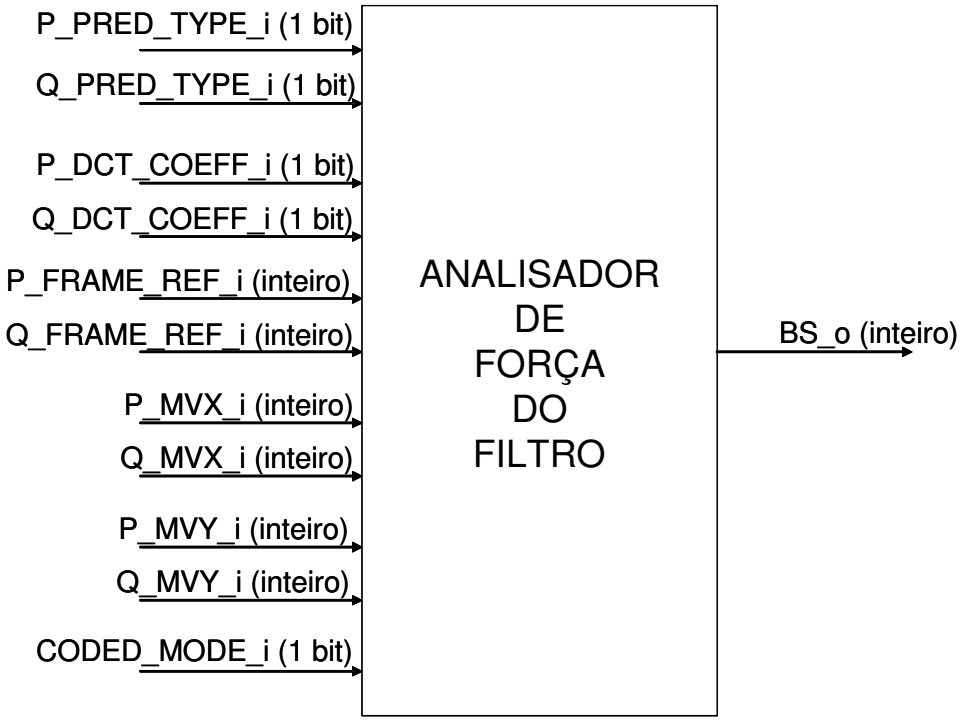


Figura 20 Bloco analisador de força de filtragem (*Boundary Strength*).

Para identificar se a filtragem está ocorrendo na borda do macrobloco, é necessário saber pelo menos o endereço de um dos blocos. Para filtragem vertical, o sinal informa se é borda de macrobloco ou não, é setado quando o bloco de entrada é 0, 2 , 8 ou 10. Caso seja filtragem horizontal, o sinal é setado quando o endereço do bloco é 0, 1, 4 ou 5.

Se a filtragem está ocorrendo na borda do macrobloco e um ou ambos os blocos possui predição intraframe, BS recebe o valor 4. Se não for borda de macrobloco e pelo menos um dos blocos possui predição intraframe, BS recebe o valor 3. Se ambos os blocos possuem predição interframes e um deles possui algum coeficiente da IDCT diferente de zero, BS recebe valor 2. Se não, se ambos blocos possuem predição interframes e possuem diferentes frames de referência, ou se a diferença entre os vetores de movimento x for maior que uma amostra de luminância, ou se a diferença entre os vetores de movimento y for maior que uma amostra de luminância, BS recebe o valor 1. Se não, BS recebe o valor 0 e nenhuma filtragem é aplicada. A implementação dessa tarefa foi realizada exatamente de acordo com essa descrição. A figura 21 mostra resumidamente o funcionamento dessa etapa.

Quando o vídeo é entrelaçado, o filtro não recebe força 4 quando faz filtragens horizontais. Se as condições para BS igual a 4 forem satisfeitas na filtragem horizontal no modo entrelaçado, BS recebe valor 3. Essa é a única diferença entre vídeo entrelaçado e progressivo.

Como esse bloco apresenta pequenas diferenças entre a filtragem vertical e horizontal, optou-se por torná-lo parametrizável, para que se possa usar o mesmo componente em ambos os processos. O parâmetro de configuração é uma string, e deve receber ou o valor “vertical” ou o valor “horizontal”, dependendo de onde o bloco será inserido.

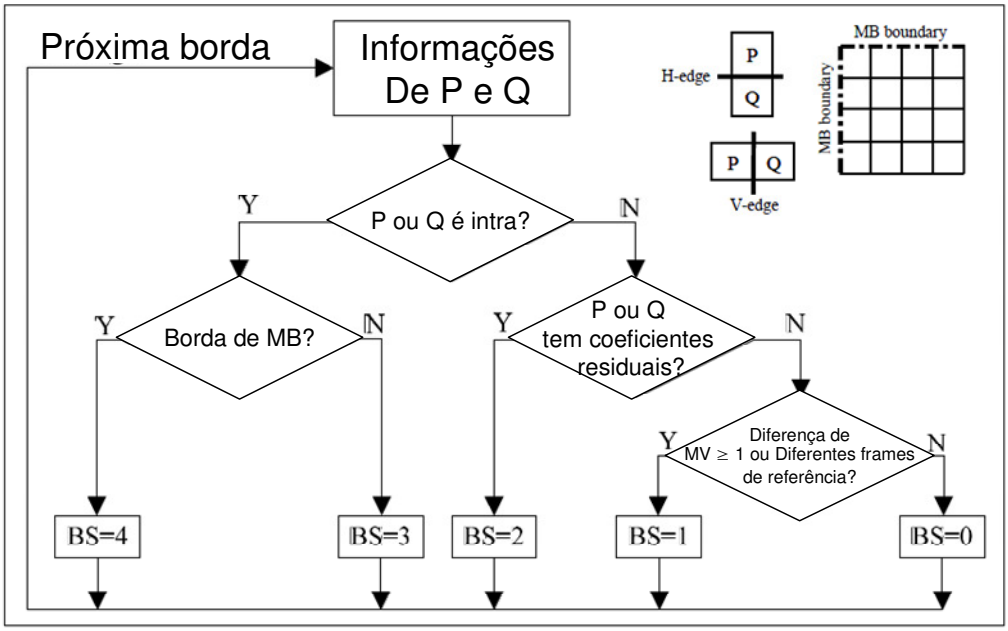


Figura 21 Escolha de BS.

Para blocos de crominância, o procedimento para obtenção do valor de BS não segue esse padrão, e baseia-se no valor de BS calculado para blocos de luminância e da posição do pixel Q0 do bloco de crominância. Na implementação dessa característica foi usado um buffer que armazena o valor de BS quando está analisando blocos de luminância e faz a leitura desse buffer quando chegam blocos de crominância.

Foi realizado uma simulação para os mesmos valores de linhas de pixels, sendo que foi forçado os valores de BS. Desse modo, é possível observar a influência de cada força na mesma amostra de pixels. Os resultados dessa simulação se encontram na figura 22. Como foi descrito nas equações de filtragem, para BS igual a 1, 2 e 3 são realizadas as mesmas operações matemáticas, sendo que a principal diferença é o valor máximo que cada pixel pode variar. Já para BS igual a 4, existem operações especiais. Outra diferença entre os dois modos é que para BS igual a 4, até 3 pixels por linha podem ser alterados na filtragem, e para BS igual a 1, 2 ou 3, no máximo 2 pixels

sofrem alterações. Para BS igual a zero, nenhuma filtragem ocorre, ou seja, são os valores originais da linha de pixels.

	LOP BLOCO P				LOP BLOCO Q			
BS = 4	27	31	35	37	42	44	45	46
BS = 3	27	28	34	39	43	43	46	46
BS = 2	27	28	34	39	43	43	46	46
BS = 1	27	28	34	39	43	43	46	46
BS = 0	27	28	30	39	43	47	46	46

Figura 22 Comparação de resultados para diferentes forças de filtragem.

O tempo entre os dados chegarem e BS sair é de um ciclo de clock.

4.9 DEFINIÇÕES INICIAIS

Antes das operações de filtragem serem realizadas, são feitos alguns testes que definem se haverá ou não filtragem. Esses testes devem ser feitos para identificar se essa borda é real, ou seja, ela existe na imagem original, ou se é uma borda de um artefato de bloco originada pela DCT.

Para isso, criou-se um componente responsável por realizar esses testes. As operações matemáticas são as equações (2.1) a (2.10).

O bloco funciona da seguinte maneira: recebe como entrada a linha de pixels do bloco P, a linha de pixels do bloco Q, o parâmetro de quantização de P e Q, o offset A e offset B configurados pelo usuário e se são blocos de luminância ou crominância. A partir dos valores de QP de P e Q, calcula-se Q_{PAV} de acordo com a equação (2.4). O resultado dessa operação é usado para obter INDEX_A e INDEX_B, que estão armazenados em uma tabela que está declara em um "package". Usa-se INDEX_A para buscar na tabela o valor de α, e INDEX_B para obter β, que também estão declarados no mesmo package anterior. Realizam-se os testes descritos nas equações (2.1), (2.2) e (2.3). Se todas forem verdadeiras e se BS for diferente de zero, significa que ocorrerá filtragem e um sinal que habilita a filtragem recebe valor 1. Se pelo menos uma das equações for falsa, ou se BS for zero, não haverá filtragem e o sinal que habilita a filtragem recebe valor zero. Isso nada mais é do que um AND dos três testes e do valor de BS.

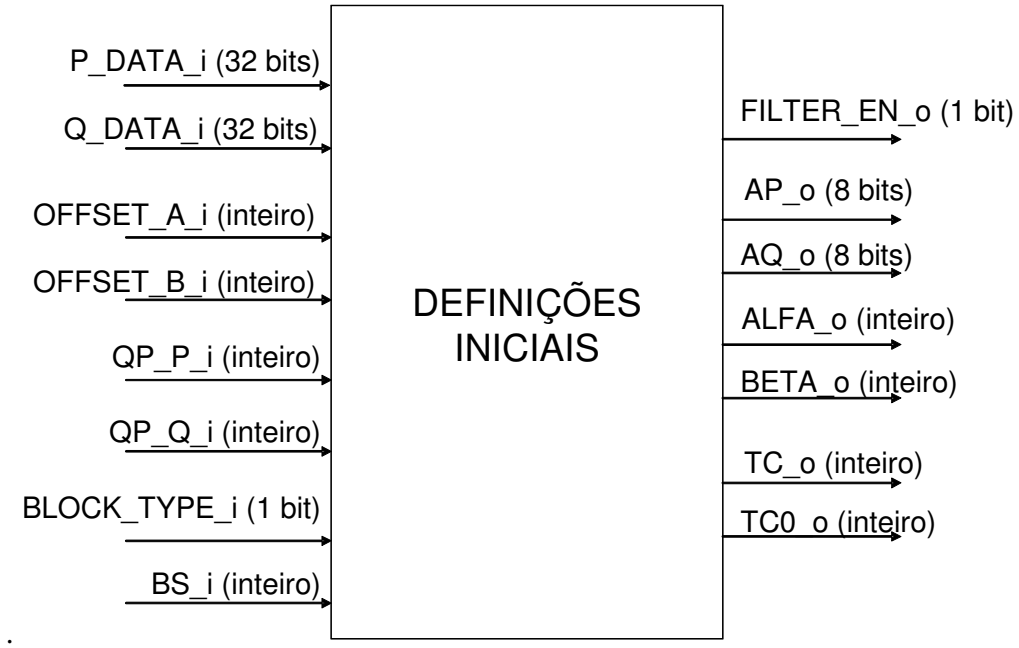


Figura 23 Bloco responsável por realizar as primeiras operações necessárias para a filtragem.

Aproveitou-se da disponibilidade de informações que esse bloco recebe para calcular outros fatores usados no processo de filtragem. Nesse bloco, obtém-se também o valor de t_{C0}, que é buscado no package que contém as tabelas usando o valor de INDEX_A e de BS. É calculado também o valor de a_P e a_Q através das equações (2.6) e (2.7). Com t_{C0}, a_P e a_Q obtidos, e usando a informação do tipo de bloco (luminância ou crominância), calcula-se o valor de t_C.

Como saída, esse bloco tem o sinal que indica se ocorrerá ou não filtragem, o valor de t_C, t_C0, α, β, a_P e a_Q.

Essa etapa leva um ciclo de relógio para ser realizada

4.10 PROCESSO DE FILTRAGEM

É nesse componente que acontecem as operações nos pixels, dependendo do sinal que habilita ou não a filtragem. Se não estiver habilitada, a linha de pixels sai exatamente como entrou. Esse arquivo foi declarado como comportamental.

Para realizar as operações matemáticas de forma correta, recebe o valor de BS e o tipo de bloco que está chegando (luminância ou crominância), além dos valores obtidos no componente que calcula as definições iniciais: t_C, t_{C0}, α, β, a_P e a_Q.

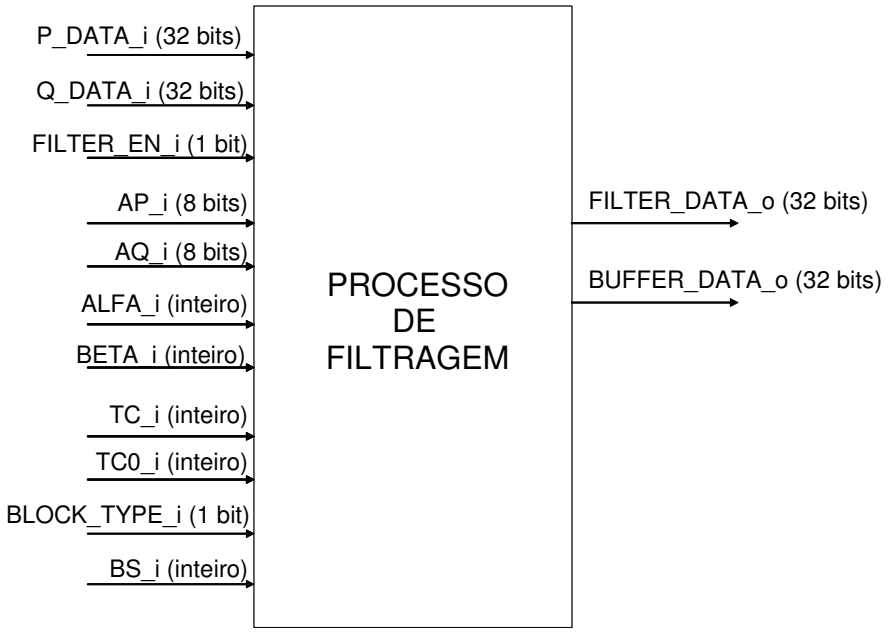


Figura 24 Bloco responsável pelas operações matemáticas de filtragem.

Com todos os dados disponíveis, as equações (2.11) a (2.32) são realizadas e a filtragem ocorre. O tempo entre a linha de pixels chegar e a saída filtrada sair é de um ciclo de relógio.

Para filtragens com BS igual a 1, 2 ou 3, foram implementados os testes $a_p < \beta$ e $a_q < \beta$ com mandos concorrentes. As equações (2.11), (2.12), (2.15), (2.16), (2.18) e (2.19) também foram implementadas com comandos concorrentes. As equações (2.13) e

(2.14) foram são calculadas com comandos seqüenciais. A equação (2.17), também implementada com comandos seqüenciais, é calculada se a condição $a_p < \beta$ for satisfeita e se forem amostras de luminância. O mesmo vale para (2.20) caso $a_q < \beta$ verdadeiro e se forem também amostras de luminância.

Para filtragens de força máxima (BS = 4), primeiramente realizam-se os testes (2.21) e (2.26). Para esses testes são usados comandos concorrentes. Se o teste (2.21) for satisfeito, as operações (2.22), (2.23), (2.24) serão realizadas. Se não, apenas a equação (2.25) acontece. Caso o teste (2.26) for satisfeito e as operações (2.27), (2.28) e (2.29) serão realizadas. Caso contrário, a equação (2.30) é realizada. Para amostras de crominância, apenas (2.31) e (2.32) são calculadas. Essas equações foram declaradas com comandos seqüenciais.

Caso o filtro esteja desativado, por não ter atendido algum dos testes ou por receber BS igual a zero, os dados saem exatamente como entraram.

Caso a linha de pixels da entrada esteja passando pela primeira filtragem (bloco Q), após terminar o processamento, ela será armazenada para que possa realizar a filtragem da segunda borda. Se a linha de pixels estiver passando pela segunda filtragem, ela será encaminhada para saída.

4.11 RAMS

Na filtragem horizontal, os blocos que ficam na borda inferior do macrobloco precisam esperar que o macrobloco abaixo dele chegue no filtro para realizar a última filtragem horizontal e assim, estar pronto para ser enviado para a saída. Enquanto o macrobloco inferior não chega, é necessário armazenar esses blocos. Porém, o

macrobloco inferior só chegará ao filtro quando a linha inteira de macroblocos tiver chegado ao final da filtragem. Por isso, é preciso armazenar uma linha completa de blocos. Para isso, foram usadas RAMs parametrizáveis. Os parâmetros que configuram as RAMs são a resolução e a largura de dados que deseja-se armazenar. Foram usadas 8 RAMs para armazenar os dados e mais 6 RAMs para armazenar informações dos blocos.

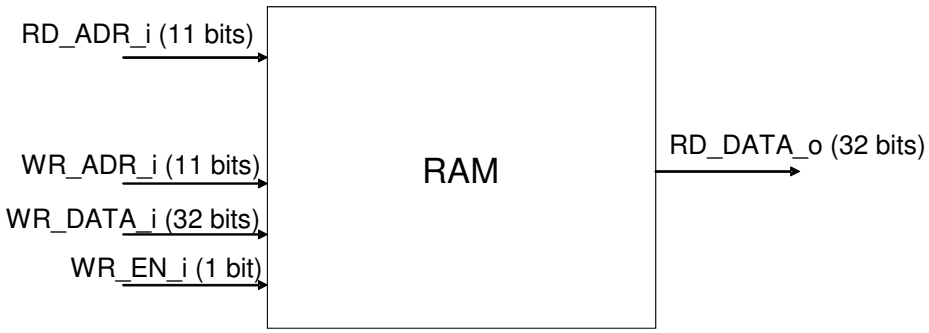


Figura 25 RAM.

O endereçamento das RAMs, escrita e leitura, é feito pelo endereço de macrobloco que o bloco pertence. E tanto a escrita como a leitura funcionam em modo síncrono, e levam 1 ciclo de relógio para realizar as operações.

A organização dos dados dentro da RAM está descrito na figura 25. O valor do macrobloco indica em qual posição da RAM o dado será escrito. Como cada RAM armazena um bloco, e cada bloco possui quatro linhas de pixel, o número de cada linha de pixels complementa o endereçamento da RAM.

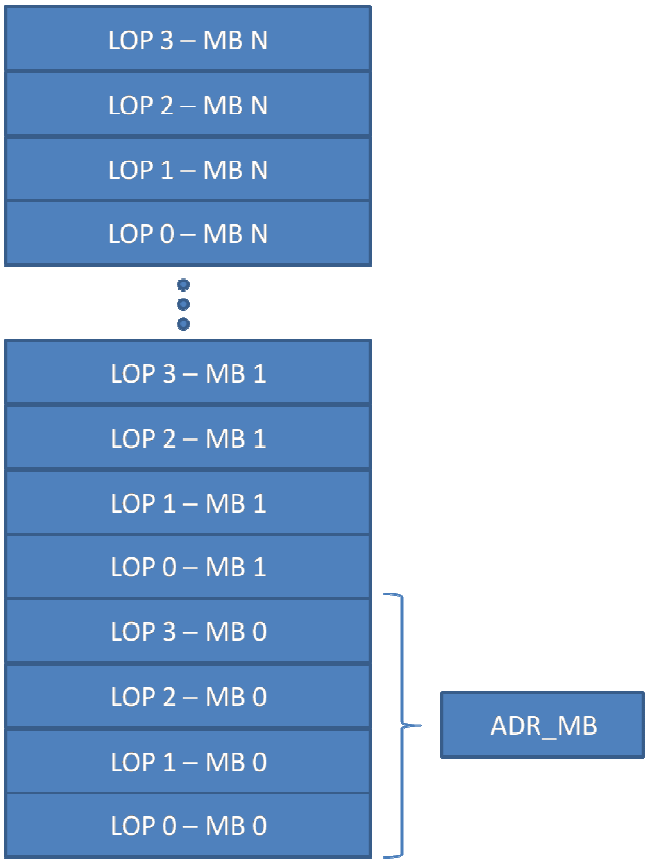


Figura 26 Organização interna dos dados na RAM.

Foram criados dois componentes diferentes para as RAMs: um que armazena sinais do tipo std_logic_vector e outro que armazena inteiros.

4.12 TRANSPOSTA

Com o intuito de aproveitar os componentes gerados na filtragem vertical para realizar a filtragem horizontal, foi criado um componente que reorganiza o bloco de maneira transposta, já que as operações de ambas as filtragens são idênticas.

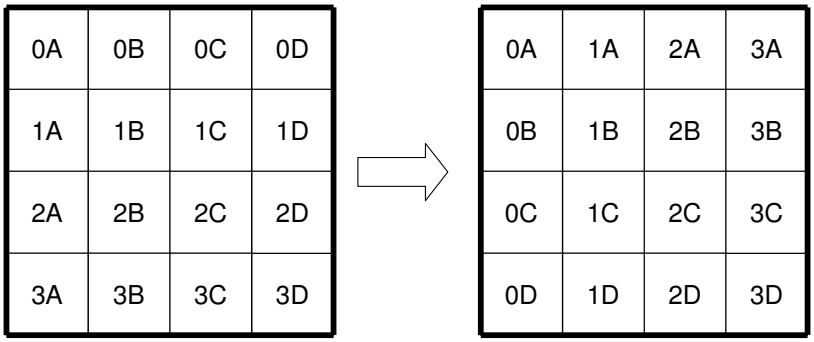


Figura 27 Bloco antes e depois de passar pela transposta.

Pode ser observado que a partir do momento que chega a primeira linha de pixels do bloco, haverá uma saída válida apenas quando a última linha de pixels chegar. Se para cada bloco que chegasse, fosse necessário esperar 4 ciclos para que houvesse uma saída válida, seria desperdício de tempo de processamento. Para isso, pensou-se em uma solução que eliminasse esse fator. Foi criado um buffer que é capaz de armazenar dois blocos por vez. Enquanto está armazenando um dos blocos, o outro é mandando para a saída.

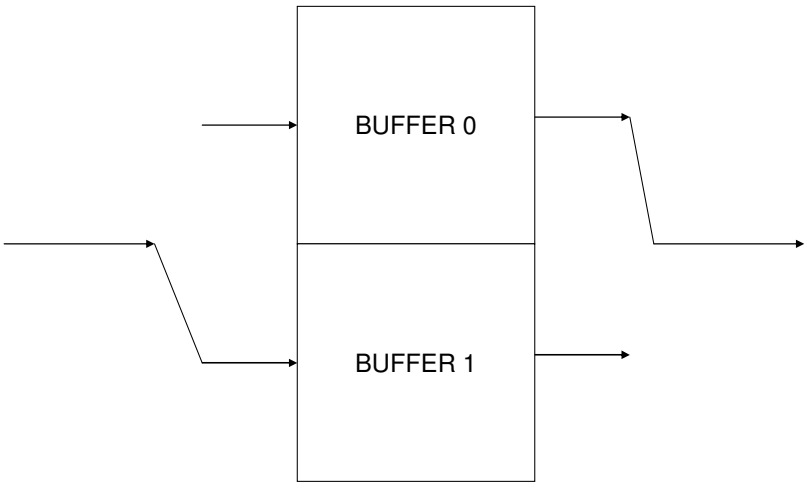


Figura 28 Funcionamento da transposta.

Além de receber os dados da linha de pixel, esse componente recebe o valor do fator de quantização, o tipo de predição, os valor dos vetores de movimento (x, y), o sinal que indica se o bloco possui ou não coeficientes da DCT igual a zero, e o frame de referência do bloco. Também recebe o endereço do bloco no macrobloco.

Esse bloco funciona da seguinte maneira: ao chegar um dado válido, habilita um contador de escrita (de 0 até 7), que é usado como endereçador nesse buffer. Para valores menor que 4, escreve no buffer 0, e para valores maior ou igual a 4 escreve no buffer 1.

A saída é válida assim que um buffer estiver cheio. Desse modo, na maior parte do processo, há uma saída válida indo em direção a filtragem horizontal.

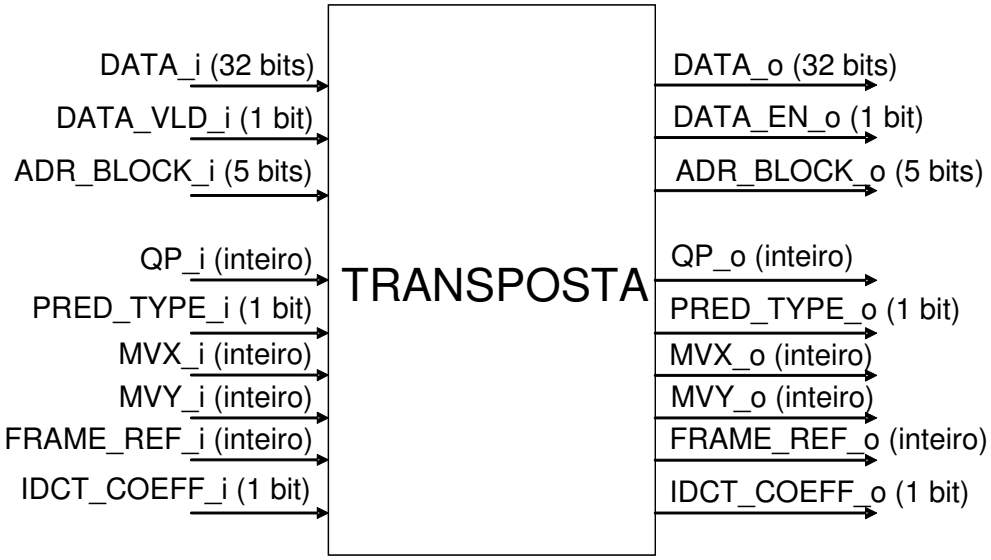


Figura 29 Bloco que faz a transposta.

4.13 PACKAGES

Foi criado um arquivo do tipo “package” apenas para declarar os blocos como componentes, chamado de “components_package”. Optou-se por essa abordagem para tornar mais organizado do que declarar os componentes junto com a descrição de funcionamento. Assim, sempre que fosse preciso mudar alguma entrada ou saída, em um ou mais componentes, modificava-se esse package, onde se encontravam todas as declarações.

Criou-se outro package para declarar alguns tipos de sinais muito usados, por exemplo, a linha de pixels de 32 bits. Nesse caso criou-se um subtype da largura de 32 bits, o que facilitou muito em todos os momentos que eram necessários criar um sinal dessa largura. Outra funcionalidade desse package foi a de declarar as tabelas 2, 3 e 4 como arrays constantes.

E, havia mais um package, usado para declarar algumas funções matemáticas comumente usadas, como a função CLIP, que funciona como um saturador.

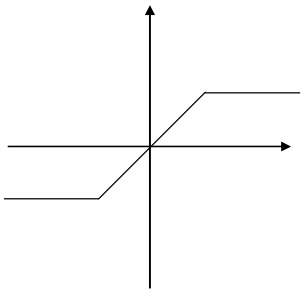


Figura 30 Saturador.

5 VALIDAÇÃO

Para realizar a validação do projeto, foi usado como referência de entrada e saída o software CodecVisa. Esse software é uma ferramenta analisadora de vídeos, e suporta vídeos codificados no padrão H.264.

Além de fornecer o valor de cada pixel no frame antes e depois de passar pelo processo de filtragem, o software é capaz de fornecer ainda o valor de QP do macrobloco, o tipo de predição, os vetores de movimento, quadro de referência e os coeficientes da IDCT, ou seja, através desse software é possível obter de maneira fácil todos os dados necessários para a realização dos testes de verificação.

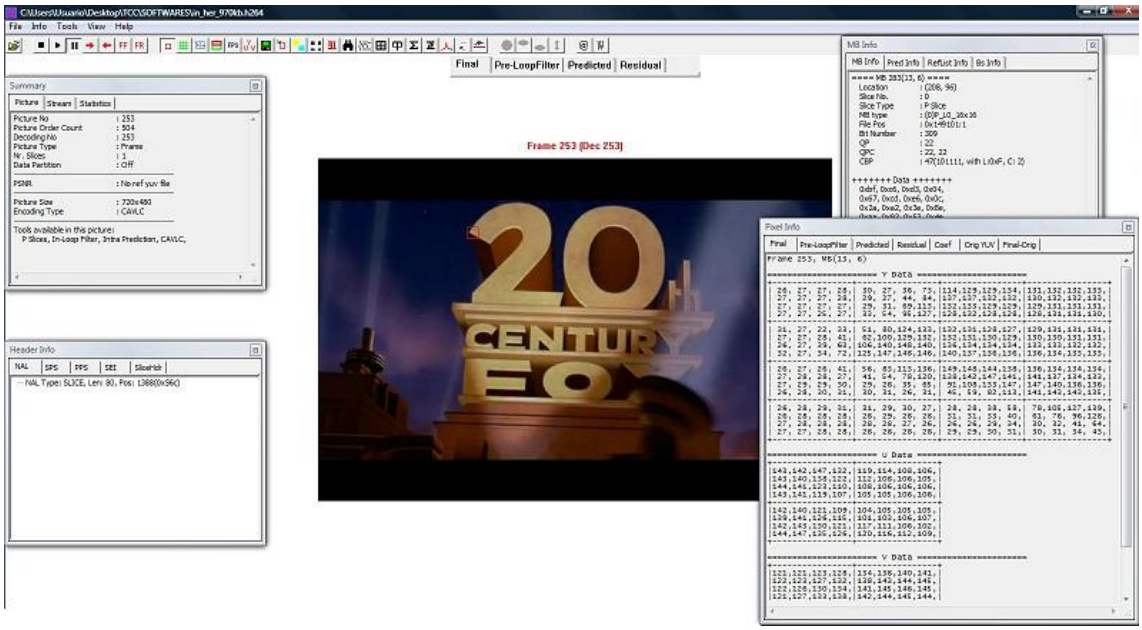


Figura 31 Software CodecVisa.

Uma das desvantagens desse software é o custo. Apesar de ser uma ótima ferramenta, o custo de aquisição de uma licença é elevado, chegando a custar $ 1.600,00. Porém, o desenvolvedor fornece gratuitamente uma versão trial, que permite a utilização do software por 30 dias.

A utilização do CodecVisa é muito simples, e basta abrir o vídeo escolhido, escolher o frame e o macrobloco. Em uma janela aparecerão informações sobre o macrobloco, como posição, fator de quantização, tipo de predição, vetores de movimento, frame de referência, offset A e offset B e valor do fator BS em cada borda do macrobloco. Em uma outra janela aparecem os valores dos pixels, tanto para blocos de luminância como para blocos de crominância. Nessa mesma janela é possível obter o valor do pixel antes e depois da filtragem, assim como o valor dos coeficientes da IDCT de cada bloco.

Todas as informações usadas para montar os arquivos de testes foram extraídas usando este software.

Após implementar todos os blocos e conecta-los de maneira correta, respeitando o sincronismo necessário para o correto funcionamento do sistema, foi usado o software de simulação Modelsim para analisar o comportamento do sistema. Com o Modelsim é possível conferir o valor de qualquer sinal criado em qualquer instante da simulação. Para realizar a simulação, foi criado um arquivo do tipo “testbench”, que simula as entradas do filtro e observas as saídas.

6 RESULTADOS ALCANÇADOS

Após o sistema estar funcionando de maneira correta, foi gerado um arquivo "testbench" para um quadro QCIF (176x144 pixels). Com os valores de cada bloco com as 4 bordas filtradas, foi usado o software YUV Tools que remonta o quadro a partir dos valores dos pixels.

Figura 32 Quadro antes e depois de passar pelo filtro redutor de efeito de bloco.

É possível perceber uma grande diferença entre os dois quadros. Enquanto o da esquerda apresenta muitos artefatos de bloco, o q imagem da direita já se mostra com uma aparência muito mais suavizada.

Em relação ao tempos de processamento para cada bloco, depende da posição onde o mesmo se encontra no macrobloco. Por exemplo, o bloco 0 é processado rapidamente pois na seqüência que chegam os blocos, ele não precisa ficar armazenado esperando algum bloco distante entrar no filtro. Já o bloco 10, por exemplo, precisa esperar acabar uma linha inteira de macroblocos para ter sua última borda processada. A tabela 5 mostra o tempo de processamento em algumas etapas da filtragem, e as referências de tempo valem para o bloco 0.

Tabela 5 – Tempos de processamento

Eventos	Ciclos
Processamento de uma borda vertical	9 ciclos
Processamento de uma borda horizontal	7 ciclos
Processamento de um bloco (todas as bordas)	32 ciclos
Processamento de um macrobloco	124 ciclos

Após validado o comportamento do sistema, todo o filtro foi sintetizado. Para isso, foi usado o compilador ISE 10.1 da Xilinx e configurado duas FPGAs: o XC2VP30-FF896 e o XC5V1X110T-FF1136. Essas são as FPGAs usadas no LAPSI, por isso foram escolhidas para gerar a síntese. O resultado da síntese encontra-se nas tabelas 6 e 7.

Tabela 6 – Resultado da síntese para XC2VP30

Utilização do dispositivo	Quantidade
Number of Slices	7318
Number of Slice Flip Flops	6027
Number of 4 input LUTs	10516
Number used as logic	10011
Number used as Shift registers	153
Number of IOs	341
Number of bonded IOBs	340
Number of FIFO16/RAMB16s	19
Number of GCLKs	1

O valor máximo de clock gerado foi 58,413 MHz para a XC2VP30 e 88,320 MHz para a XC5V1X110T. O objetivo é trabalhar em tempo real com vídeos em alta definição (HD). Como um frame possui 1920 pixels de largura e 1080 de altura, 30 frames por segundo e considerando que o vídeo é uma seqüência 4:2:0, é necessário uma taxa de transmissão aproximada a 750 Mbits/s. Como a cada ciclo de relógio saem 32 bits, para atender a taxa necessária é necessário um clock mínimo de 23,328 MHz. Desse modo, o sistema sintetizado atende as especificações. Outro dado interessante é o

número de pixels por segundo, que nesse caso é 233.652 megapixels por segundo para a primeira FPGA e 353,28 megapixels por segundo para a segunda.

Tabela 7 – Resultado da síntese para XC5V1X110T

Utilização do disposivo	Quantidade
Slice Logic Utilization:	
Number of Slice Registers:	6017
Number of Slice LUTs:	6794
Number used as Logic:	6554
Number used as Memory:	240
Number used as RAM:	88
Number used as SRL:	152
Slice Logic Distribution:	
Number of LUT Flip Flop pairs used:	10901
Number with an unused Flip Flop:	4884
Number with an unused LUT:	4107
Number of fully used LUT-FF pairs:	1910
Number of unique control sets:	177
IO Utilization:	
Number of IOs:	341
Number of bonded IOBs:	340
Specific Feature Utilization:	
Number of Block RAM/FIFO:	13
Number using Block RAM only:	13
Number of BUFG/BUFGCTRLs:	2

7 CONCLUSÃO

Foi fundamental para o sucesso na implementação um estudo cuidadoso do comportamento do filtro e dos conhecimentos adquiridos junto ao laboratório LAPSI.

Os resultados foram bastante satisfatórios, estando de acordo com o software de referência. Fica claro, observando os resultados da síntese, que o filtro redutor de efeito de bloco exige um grande gasto de área das FPGAs .

O conteúdo aprendido durante a realização do projeto, tanto sobre codificadores quanto sobre sistemas digitais e VHDL serão de extrema importância para entrar num mercado de trabalho onde cada detalhe faz diferença.

REFERÊNCIAS

CHAO, Y. C.; LIN, H. Y.; LIU, B. D.; YANG, J. Y. "A High Throughput and Data Reuse Architecture for H.264/AVCDeblocking Filter", *Proceedings of the 2006 IEEE Asia-Pacific Conference on Circuits and Systems,* pp. 1262-1265. (December 4-7, 2006, Singapore)

Draft ITU-T Recommendation and Final Draft international Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14496-10 AVC), March 2010

LIN, H.-Y.; YANG, J.-J.; LIU, B.-D.; YANG, J.-F. Efficient deblocking filter architecture for H.264 video coders. *2006 IEEE International Symposium on Circuits and Systems*, ISCAS 2006, 21-24 May 2006.

LIST, P.; JOCH A.; LAINEMA, J.; BJONTEGAARD, G.; KARCZEWICZ, M. Adaptive Deblocking Filter, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, pp.614-619, 2003

LIU, T.-M.; LEE, W.-P.; LIN, T.-A.; LEE, C.-Y. A Memory-Efficient Deblocking Filter for H.264/Avc Video Coding, *Circuits and Systems, 2005. ISCAS 2005*. IEEE International Symposium on 23-26 May 2005 Page(s):2140 – 2143

PARLAK, M.; HAMZAOGLU I. A Low Power Implementation of H.264 Adaptive Deblocking Filter Algorithm, *Second NASA/ESA Conference on Adaptive Hardware and Systems*, Aug. 2007

PURI, A.; CHEN, X.; LUTHRA, A. Video coding using the H.264/MPEG-4 AVC compression standard, *Signal Processing: Image Communication*, Vol 19, pp. 793-849, June 200.

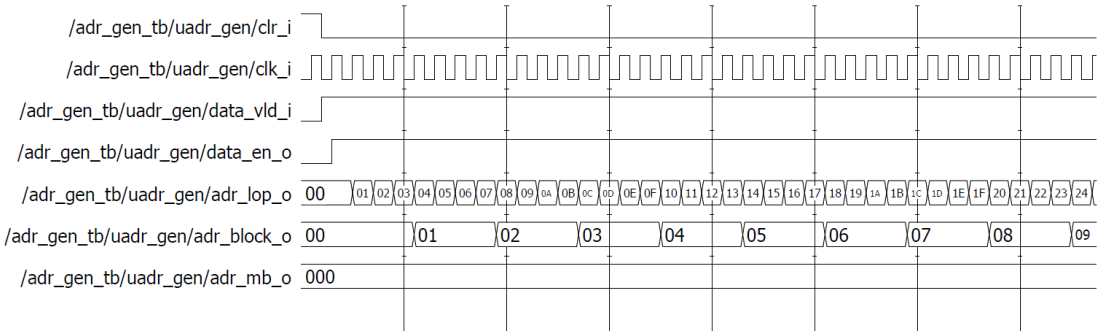
ROSA, V. S.; SUSIN, A. A.; and BAMPI, S. An HDTV H.264 deblocking filter in FPGA with RGB video output. *In Proceedings of VLSI-SoC*. 2007, 308-311

SHENG, B.; GAO, W.; WU, D. "*An Implemented Architecture of DeblockingFilter for H.264/AVC*"**,** *Proc. IEEE Int. Conf. on Image Processing*, pp. 665-668, October 2004

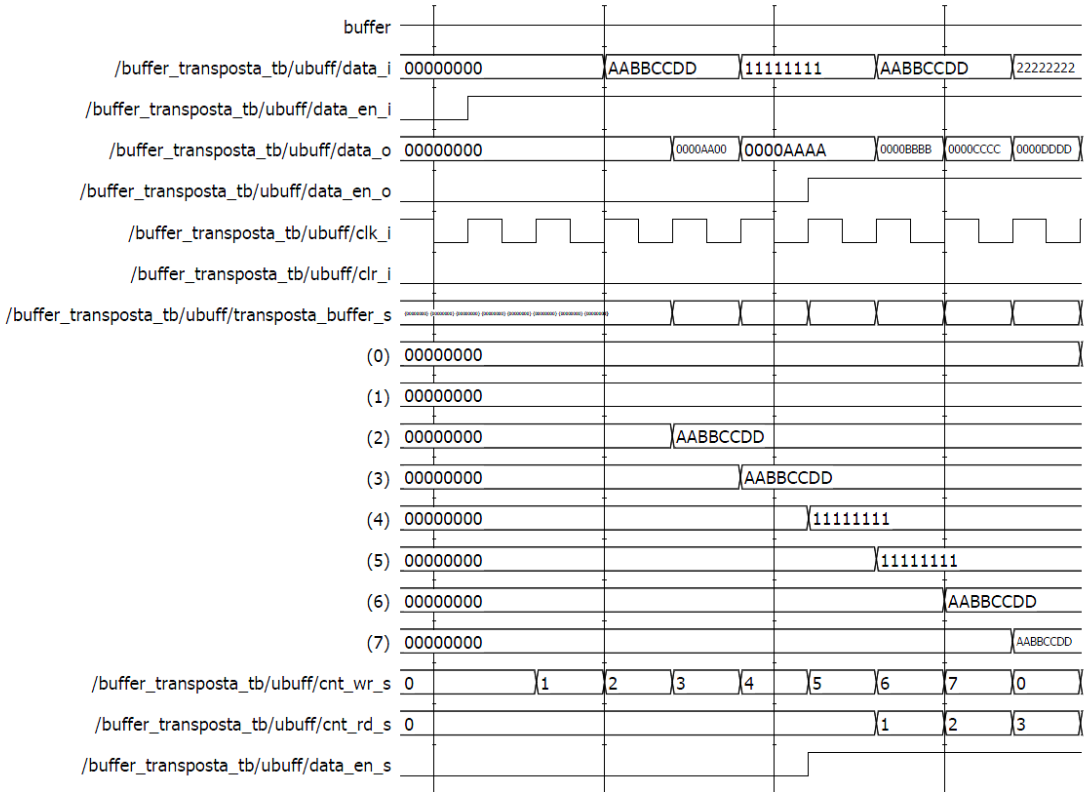
APÊNDICE:

Resultados de algumas simulações realizadas no ModelSim.

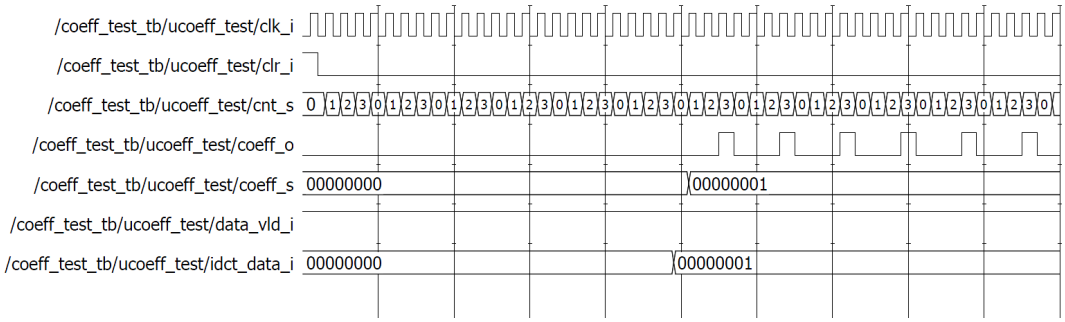
Gerador de endereços



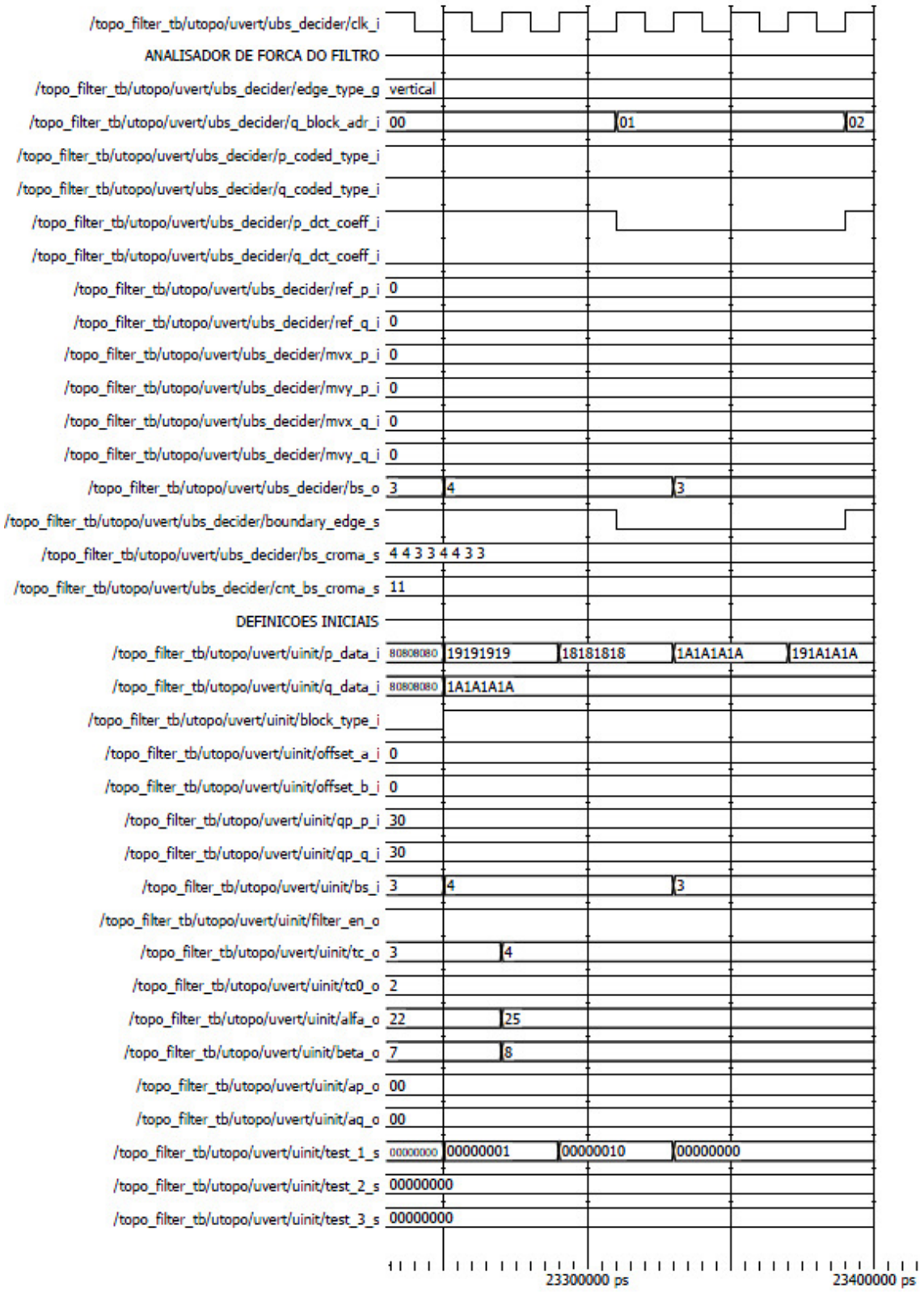
Transposta

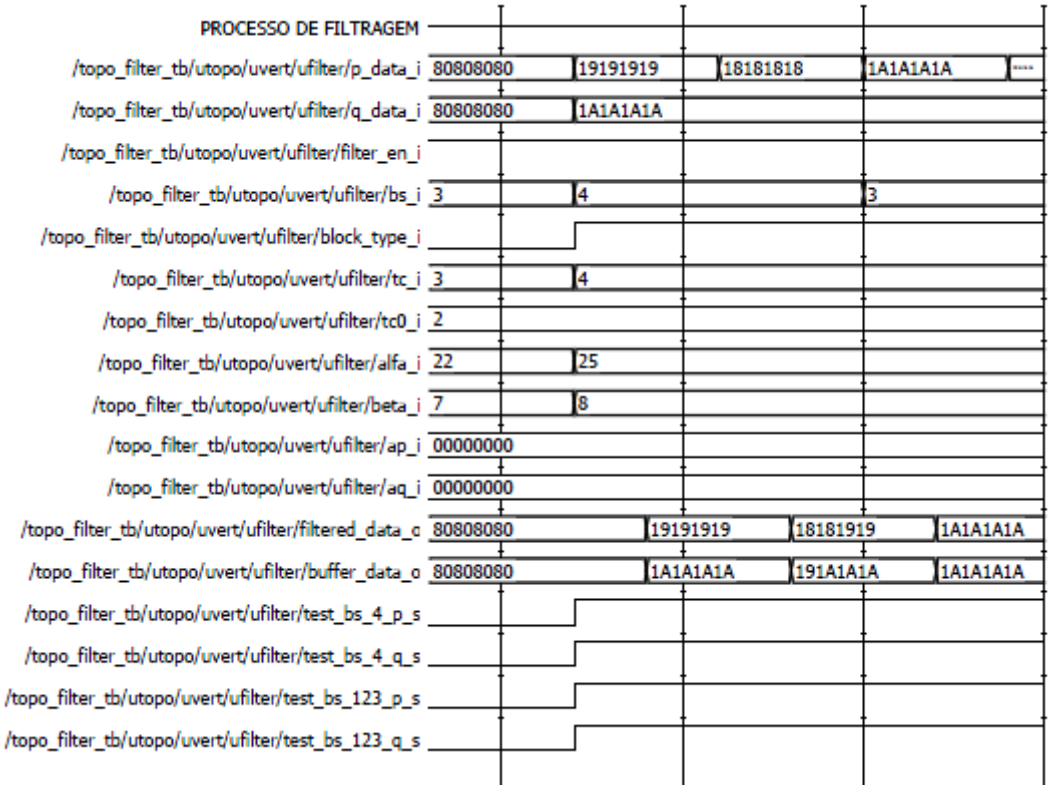

buffer
/buffer_transposta_tb/ubuff/data_i
/buffer_transposta_tb/ubuff/data_en_i
/buffer_transposta_tb/ubuff/data_o
/buffer_transposta_tb/ubuff/data_en_o
/buffer_transposta_tb/ubuff/clk_i
/buffer_transposta_tb/ubuff/clr_i
/buffer_transposta_tb/ubuff/transposta_buffer_s
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
/buffer_transposta_tb/ubuff/cnt_wr_s
/buffer_transposta_tb/ubuff/cnt_rd_s
/buffer_transposta_tb/ubuff/data_en_s
00000000
AABBCCDD
11111111
AABBCCDD
22222222
0000AA00
0000AAAA
0000BBBB
0000CCCC
0000DDDD

Verificador de coeficientes da IDCT



Analisador de força do filtro, Definições iniciais e Processo de filtragem




PROCESSO DE FILTRAGEM
/topo_filter_tb/utopo/uvert/ufilter/p_data_i
80808080
19191919
18181818
1A1A1A1A
/topo_filter_tb/utopo/uvert/ufilter/q_data_i
80808080
1A1A1A1A
/topo_filter_tb/utopo/uvert/ufilter/filter_en_i
/topo_filter_tb/utopo/uvert/ufilter/bs_i
3
4
3
/topo_filter_tb/utopo/uvert/ufilter/block_type_i
/topo_filter_tb/utopo/uvert/ufilter/tc_i
3
4
/topo_filter_tb/utopo/uvert/ufilter/tc0_i
2
/topo_filter_tb/utopo/uvert/ufilter/alfa_i
22
25
/topo_filter_tb/utopo/uvert/ufilter/beta_i
7
8
/topo_filter_tb/utopo/uvert/ufilter/ap_i
00000000
/topo_filter_tb/utopo/uvert/ufilter/aq_i
00000000
/topo_filter_tb/utopo/uvert/ufilter/filtered_data_o
80808080
19191919
18181919
1A1A1A1A
/topo_filter_tb/utopo/uvert/ufilter/buffer_data_o
80808080
1A1A1A1A
191A1A1A
1A1A1A1A
/topo_filter_tb/utopo/uvert/ufilter/test_bs_4_p_s
/topo_filter_tb/utopo/uvert/ufilter/test_bs_4_q_s
/topo_filter_tb/utopo/uvert/ufilter/test_bs_123_p_s
/topo_filter_tb/utopo/uvert/ufilter/test_bs_123_q_s