The Algorithm of H.264 (MPEG-4 part 10 AVC)

Jechang Jeong, Ph.D

Hanyang University

1

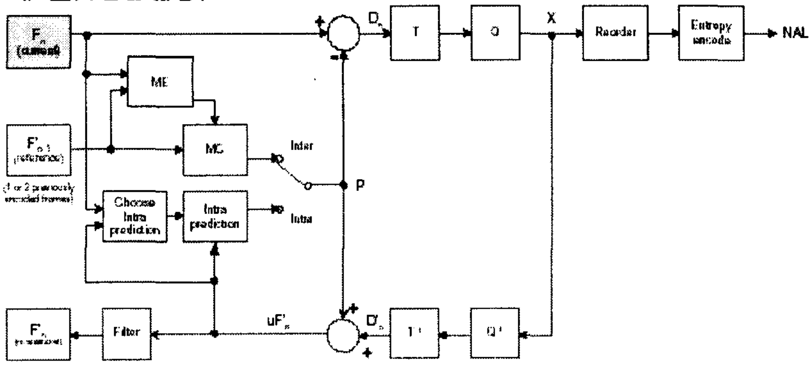
Contents

- ◈ Overview of H.264
  - Profiles in H.264
- ◈ Intra Prediction
- ◈ Inter Prediction
- ◈ Transform & Quantization
- ◈ Reconstruction Filter
- ◈ Entropy Codes
  - CAVLC vs. CABAC
- ◈ SP slices

2

Overview of H.264

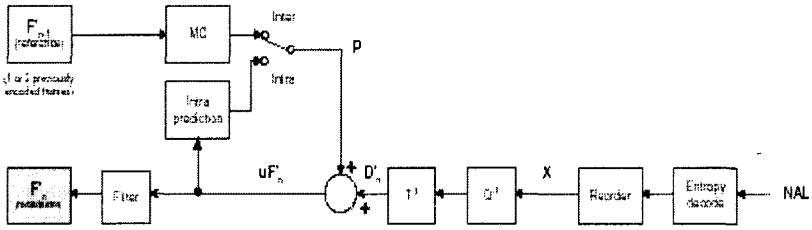
◆Encoder



3

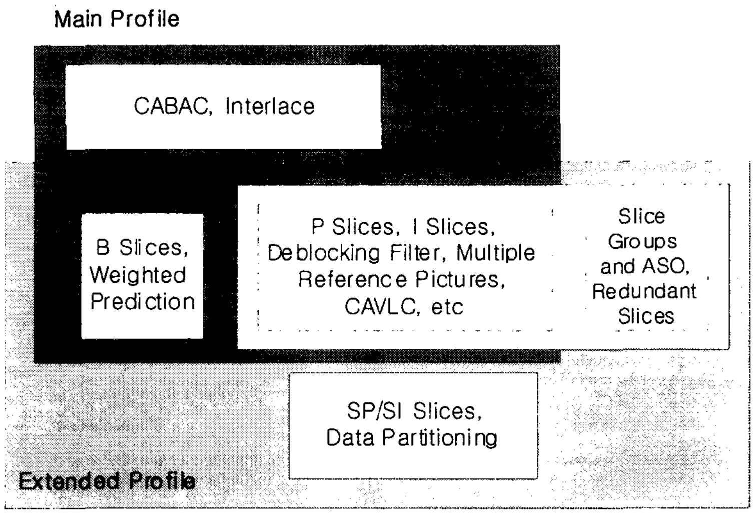
Overview of H.264

◆Decoder



4

Profiles in H.264

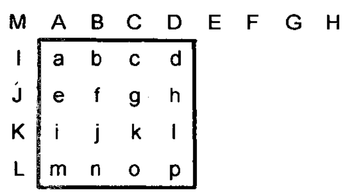


5

Intra Prediction

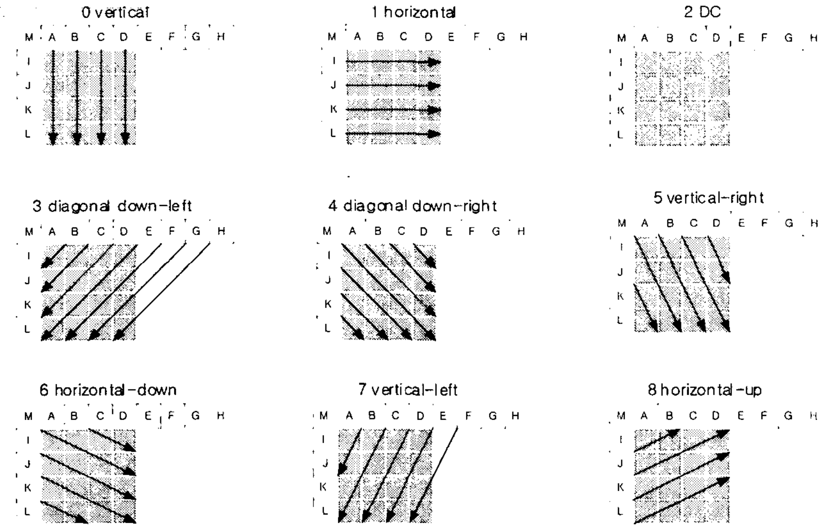
◆4×4 luma prediction mode

- When samples E-H are not available, the sample value of D is substituted for samples E-H
- For the luma signal, there are nine intra prediction modes labelled 0, 1, 3, 4, 5, 6, 7, and 8. Mode 2 is 'DC-prediction



6

Intra Prediction



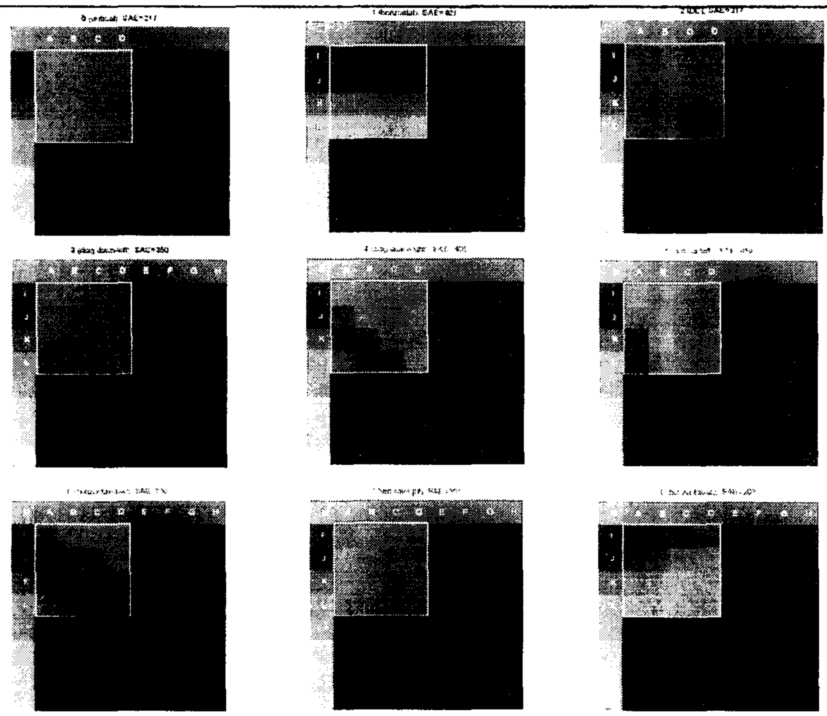
7

Intra Prediction

◈An example

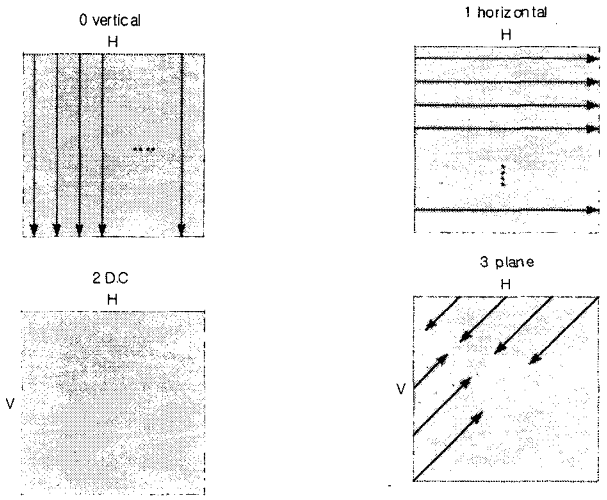
Original macroblock; 4×4 luma block to be predicted

8



Intra Prediction

◆16×16 luma prediction mode



10

Intra Prediction

◆8X8 chroma prediction mode

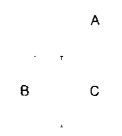
- The chroma in intra macroblocks is predicted in a manner very similar to the luma block in Intra_16x16 macroblock type, using one of four prediction modes.
- The same prediction mode is applied to both chroma blocks, but it is independent of the prediction mode used for the luma

11

Intra Prediction

◆Encoding intra prediction modes

- The choice of intra prediction mode must be signalled and this require a large number of bits
- Intra modes for neighboring 4X4 blocks are highly correlated
- Use '**most_probable_mode**' and '**use_most_probable_mode**'

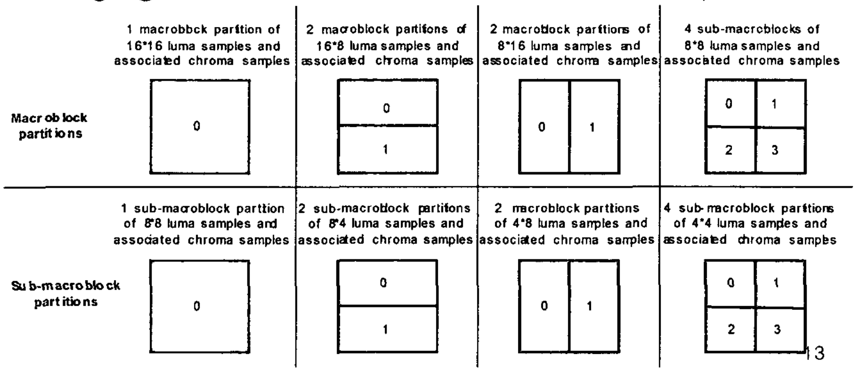


remaining mode selector	prediction mode for block C
0	0
1	2
2	3
3	4
4	5
5	6
6	7
7	8

Inter Prediction

◆ Important differences from earlier standards include the support for a range of block sizes and fine-pixel motion vectors (1/4 pixel in the luma component)

◆ H264 supports motion compensation block sizes ranging from 16x16 to 4x4 luminance samples

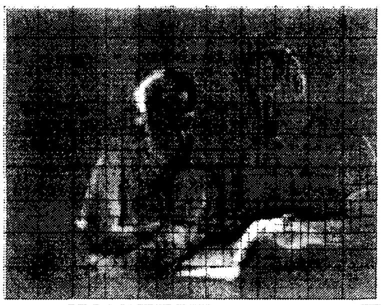


13

Inter Prediction

◆ Tree structured compensation

- Each motion vector and the choice of partition must be encoded in the compressed bitstream
- Generally, a large partition size is appropriate for homogeneous areas of frame and a small partition size may be beneficial for detailed areas
- The encoder selects the best partition size, i.e. the partition size that minimizes the coded residual and motion vectors



14

Inter Prediction

◆Sub pixel motion vector

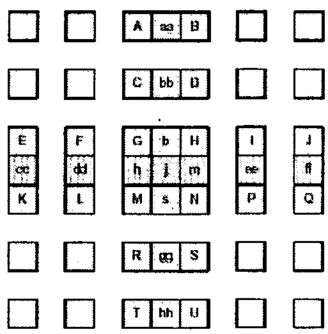
- The motion vector has ¼ pixel resolution (mandatory)
- The luma and chroma sub-pixel positions are created by using interpolation from nearby image samples
- Compression performance *vs* complexity

15

Inter Prediction

◆Interpolation of luma half pel position

- half pel sample(luma)
  - using 6 tab FIR filter which weight are (1/32,-5/32,5/8,5/8,-5/32,1/32)
  - j is generated by filtering cc, dd, h, m, ee, ff

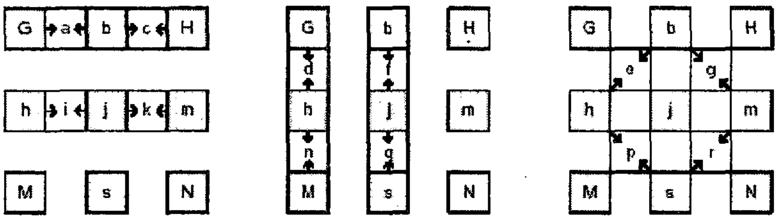


16

Inter Prediction

◈Interpolation of luma quarter pel position

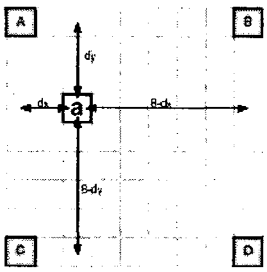
- quarter pel sample
  - Quarter pel positions are produced by linear interpolation
  - Remaining positions are linearly interpolated between a pair of diagonally opposite half pel samples



Inter Prediction

◈Interpolation of chroma eighth pel position

- Quarter pel resolution motion vectors in the luma component will require eighth pel resolution vectors in the chroma components (for 4:2:0 format)
- Linear interpolation is used



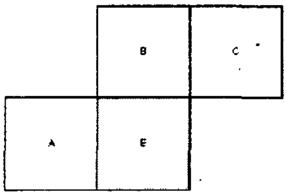
$$a = round(((8-d_x)(8-d_y)\cdot A + d_x\cdot(8-d_y)\cdot B + (8-d_x)\cdot d_y\cdot C + d_x\cdot d_y\cdot D)/8^2)$$

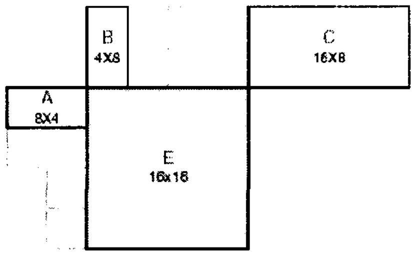
18

Inter Prediction

◆Motion vector prediction

- Motion vectors for neighbouring partitions are highly correlated and so can be predicted
  - MVp: predicted vector
  - MVD: the difference between the current vector and the MVp, which is encoded and transmitted





Transform & Quantization

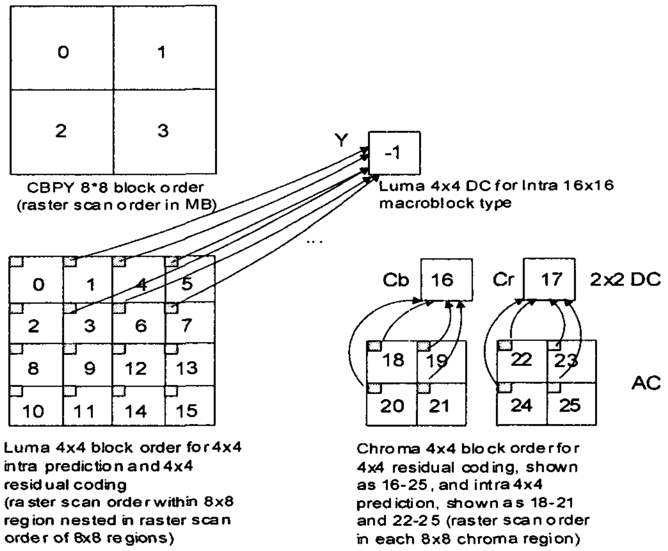
◆ Baseline profile of H.264 uses three transforms:

- 4×4 luma DC coefficients in intra macroblocks
- 2×2 chroma DC coefficients
- All other 4×4 blocks in the residual data

◆ If the optional 'adaptive block size transform (ABT)' mode is used, further transforms are chosen depending on the motion compensation block size (4×4, 8×4, 8×8, 16×8, etc)

20

Transform & Quantization



Transform & Quantization

- 4x4 residual transform and quantization
  - This transform operates on 4x4 blocks of residual data after motion-compensated prediction or intra prediction.
  - The transform is based on the DCT but with some fundamental differences :
    - Integer transform
    - Mismatch between encoders and decoders should not occur.
    - Core part of the transform is multiply-free.
    - A scaling multiplication is integrated into the quantizer.

Transform & Quantization

◆Derivation from the 4×4 DCT

- 4×4 DCT of an input array X is given by

$$Y = AXA^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} & & & \\ & X & & \\ & & & \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

$$Y = (CXC^T) \otimes E = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} & & & \\ & X & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

$$where,\quad a = \frac{1}{2},\quad b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),\quad c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right),\quad d = \frac{c}{b} \approx 0.414$$

23

Transform & Quantization

- To simplify the implementation of the transform

$$select\quad d = \frac{1}{2},\quad then\quad a = \frac{1}{2},\quad b = \sqrt{\frac{2}{5}}$$

$$Y = C_f X C_f^T \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} & & & \\ & X & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

- **Core part,** CXC^T : only additions, subtractions and shifts
- Inverse transform is given by

$$X' = C_i^T (Y \otimes E_i) C_i = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left(\begin{bmatrix} & & & \\ & Y & & \\ & & & \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

24

Transform & Quantization

◆Quantization

- H.264 uses a scalar quantizer
- Requirements of quantizer:
  - to avoid division and/or floating point arithmetic
  - to incorporate the post- and pre-scaling matrices in transform
- Basic forward quantizer operation:

$$Z_{ij} = \text{round}(Y_{ij} / \text{Qstep})$$

where, Z_{ij} : quantized coefficient
Y_{ij} : a coefficient of the transform
Qstep : a quantizer step size

25

Transform & Quantization

- QP range: 0~51 (52 steps)
- Qstep(i+6) = 2Qstep(i), 12.5% increase at each step

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	···
QStep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	···
QP	···	18	···	24	···	30	···	36	···	42	···	48	···	51
QStep	···	5	···	10	···	20	···	40	···	80	···	160	···	224

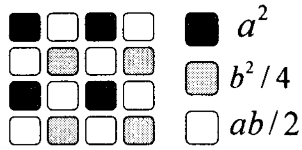
< Quantization step sizes in H.264 CODEC >

26

Transform & Quantization

- Forward quantization: post-scaling factor
- First, the input block, X is transformed into a block of unscaled coefficients $W = CXC^T$. Then, each coefficients W_{ij} is quantized and scaled in a single operation:

$$Z_{ij} = \text{round}\left(W_{ij} \cdot \frac{PF}{Qstep} \right)$$



27

Transform & Quantization

$$Z_{ij} = \left(W_{ij} \cdot MF + f \right) >> \text{qbits}$$

where $\frac{MF}{2^{qbits}} = \frac{PF}{Qstep}$ and $\text{qbits} = 15 + \text{floor}(QP/6)$

f : $2^{qbits}/3$ for Intra blocks or $2^{qbits}/6$ for Inter blocks

QP	Positions (0,0), (2,0), (2,2), (0,2)	Positions (1,1), (1,3), (3,1), (3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

28

Transform & Quantization

◆Inverse Quantization

- Basic rescale operation is : $Y'_{ij} = Z_{ij} \cdot \text{Qstep}$
- Pre-scaling factor is incorporated, together with a scaling factor of 64 to avoid rounding errors : $W'_{ij} = Z_{ij} \cdot \text{Qstep} \cdot \text{PF} \cdot 64$

$$W'_{ij} = Z_{ij} \cdot V_{ij} \cdot 2^{\text{floor}(QP/6)}$$

where $Vij = (Qstep \cdot PF)$ □ 6

QP	Positions (0,0), (2,0), (2,2), (0,2)	Positions (1,1), (1,3), (3,1), (3,3)	Other positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

29

Transform & Quantization

◆ 4x4 luma DC coeff. (16x16 Intra-mode only)

- Each 4x4 residual block is first transformed using the 'core' transform. Then DC coefficient of each 4x4 block is transformed again using a <u>4x4 Hadamard Transform</u>:

$$Y_D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \left[\quad W_D \quad \right] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

- Quantization:

$$Z_{D(i,j)} = \left(Y_{D(i,j)} \cdot MF + 2f\right) >> (\text{qbits} + 1)$$

30

Transform & Quantization

- An inverse Hadamard transform :

$$W_{QD} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} & Z_D & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

- Inverse-quantization :

If QP is greater than or equal to 12,

$$W'_{D(i,j)} = W_{QD(i,j)} \cdot V_{(0,0)} \cdot 2^{\mathrm{floor}(QP/6)-2}$$

If QP is less than or 12,

$$W'_{D(i,j)} = \left[W_{QD(i,j)} \cdot V_{(0,0)} + 2^{1-\mathrm{floor}(QP/6)} \right] >> (2 - \mathrm{floor}(QP/6)$$

31

Transform & Quantization

◆2×2 chroma DC coeff.

- Forward:

$$Y_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} W_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad Z_{D(i,j)} = \left(Y_{D(i,j)} \cdot MF + 2f \right) >> (qbits + 1)$$

- Inverse:

$$W_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} Z_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

If QP is greater than or equal to 6,

$$W'_{D(i,j)} = W_{QD(i,j)} \cdot V_{(0,0)} \cdot 2^{\mathrm{floor}(QP/6)-1}$$

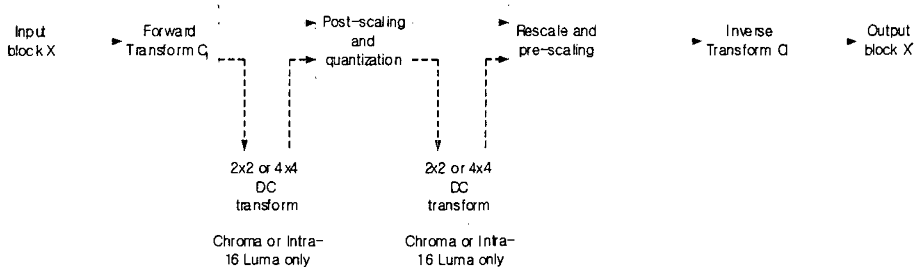
If QP is less than 6,

$$W'_{D(i,j)} = \left[W_{QD(i,j)} \cdot V_{(0,0)} \right] >> 1$$

32

Transform & Quantization

◆Diagram



< Transform, quantization, rescale and inverse transform flow diagram >

33

Reconstruction Filter

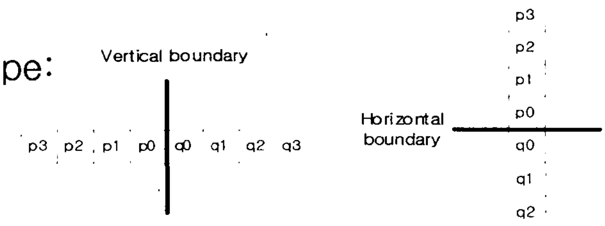
◆ Description

- A filter is applied to every decoded macroblock in order to reduce blocking distortion.
- The deblocking filter is applied
  - in the encoder : before reconstructing and storing the macroblock for future predictions
  - in the decoder : before reconstructing and displaying the macroblock
- The filter has two benefits :
  - Block edges are smoothed, improving the appearance of decoded images
  - Filtered macroblock is used for motion-compensated prediction of further frames in the encoder, resulting in a smaller residual after prediction.
- Picture edges are not filtered.

34

Reconstruction Filter

◆ Filter type:



< Pixels adjacent to vertical and horizontal boundaries >

◆ Boundary strength

P or q is intra coded and boundary is a macroblock boundary	Bs=4 (strongest filtering)
P or q is intra coded and boundary is not a macroblock boundary	Bs=3
Neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames or a different number of reference frames or different motion vector values	Bs=1
Neither p or q is intra coded; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	Bs=0 (no filtering)

35

Reconstruction Filter

◆ Filter decision

- A group of samples from the set(p2, p1, p0, q0, q1, q2) is filtered only if :
  - 1. Bs > 0
  - 2. |p0-q0|, |p1-p0| and |q1-q0| $\lesssim$ α or β
- The thresholds α and β increase with the average QP of the two blocks p and q.
- A significant change : "Switch off"
  - QP is small → low α and β
  - QP is large → high α and β, i.e. strong filtering

36

Reconstruction Filter

◆Selection α and β

	$Index_A$ (for α) or $Index_B$ (for β)																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
α	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	5	6	7	8	9	10	12	13
β	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	3	3	3	3	4	4	4

	$Index_A$ (for α) or $Index_B$ (for β)																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
α	15	17	20	22	25	28	32	36	40	45	50	56	63	71	80	90	101	113	127	144	162	182	203	226	255	255
β	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18

37

Reconstruction Filter

◆ Filter implementation

- 1. Bs ∈ { 1, 2, 3 } :
  - 4-tab filter with p1,p0,q0 and q1 to get P0 and Q0
    - If |p2-p0| < β, a 4-tap filter is applied to get P1
    - If |q2-q0| < β, a 4-tap filter is applied to get Q1
  - p1 and q1 are never filtered for chroma, only for luma
- 2. Bs = 4 :

if (|p2-p0| < β and |p0-q0| < round(α/4))
    P0 and P1 : 5-tab filtering, 4-tab filtering
    P2 : 5-tab filtering (luma only)
else
    P0 : 3-tab filtering

if (|q2-q0| < β and |p0-q0| < round(α/4))
    Q0 and Q1 : 5-tab filtering, 4-tab filtering
    Q2 : 5-tab filtering (luma only)
else
    Q0 : 3-tab filtering

38

Reconstruction Filter

◆Filtering example

Original frame (violin frame 2)

39

Reconstruction Filter

Reconstructed, QP=36 (no filter)

Reconstructed, QP=36 (with filter)

40

Entropy Codes

◆Entropy codes in H.264

- VLC : exp-Golomb code and CAVLC[1] : baseline profile
- CABAC[2] : main profile

Parameters	Description
Sequence-, picture- and slice-layer syntax elements	
Macroblock type mb_type	Prediction method for each coded macroblock
Coded block pattern	Indicates which blocks within a macroblock contain coded coefficients
Quantizer parameter	Transmitted as a delta value from the previous value of QP
Reference frame index	Identify reference frame(s) for inter prediction
Motion vector	Transmitted as a difference (mvd) from predicted motion vector
Residual data	Coefficient data for each 4x4 or 2x2 block

[1] CAVLC is an abbreviation of context-based adaptive variable length code

[2] CABAC is an abbreviation of context-based adaptive arithmetic code

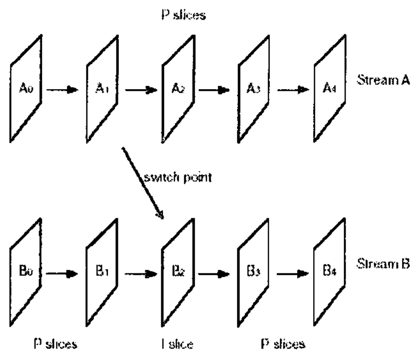
41

SP Slices

◆Extended profile

◆Meaning: switching P and I slices

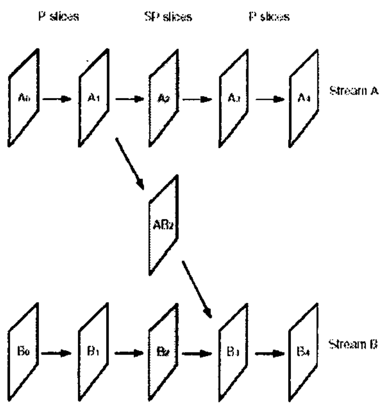
◆Case 1: switching streams using I-slices



42

SP Slices

◈Case 2: switching streams using SP-slices



43

SP Slices

◈Applications

- Bitstream switching: bandwidth scalability
- Random access
- Fast-forward
- Error resiliency/Recovery

44

The END

Thank You !!

45