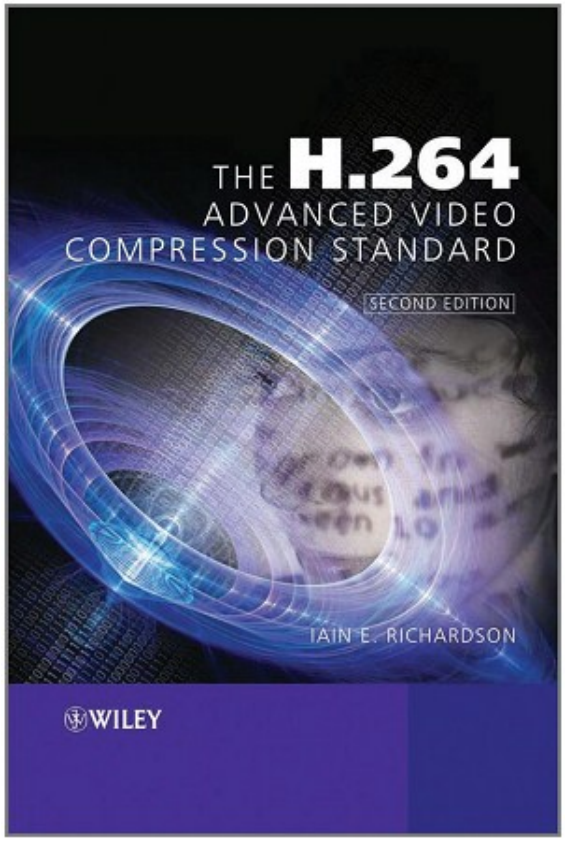
视频编解码原理

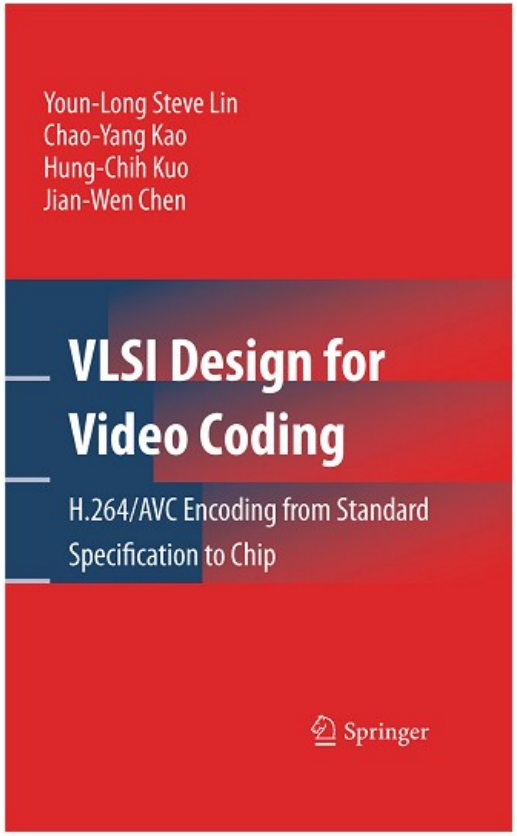
范益波

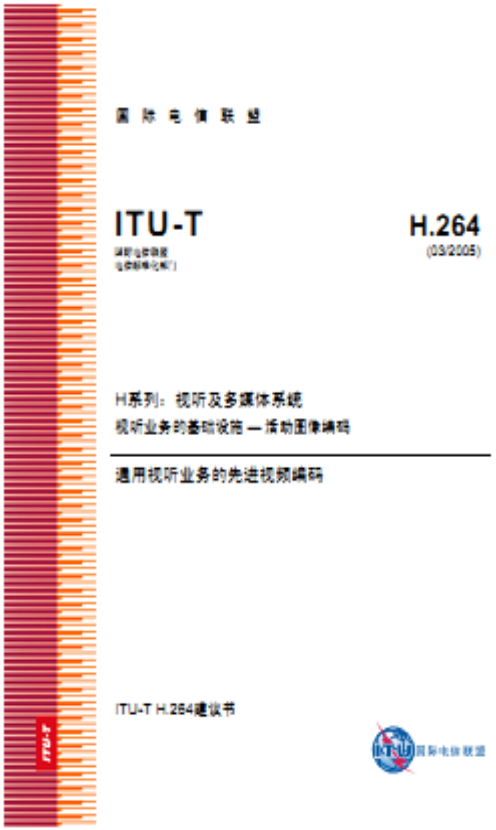
课程概要

- 视频压缩原理
- 视频编解码算法
- 视频编解码芯片

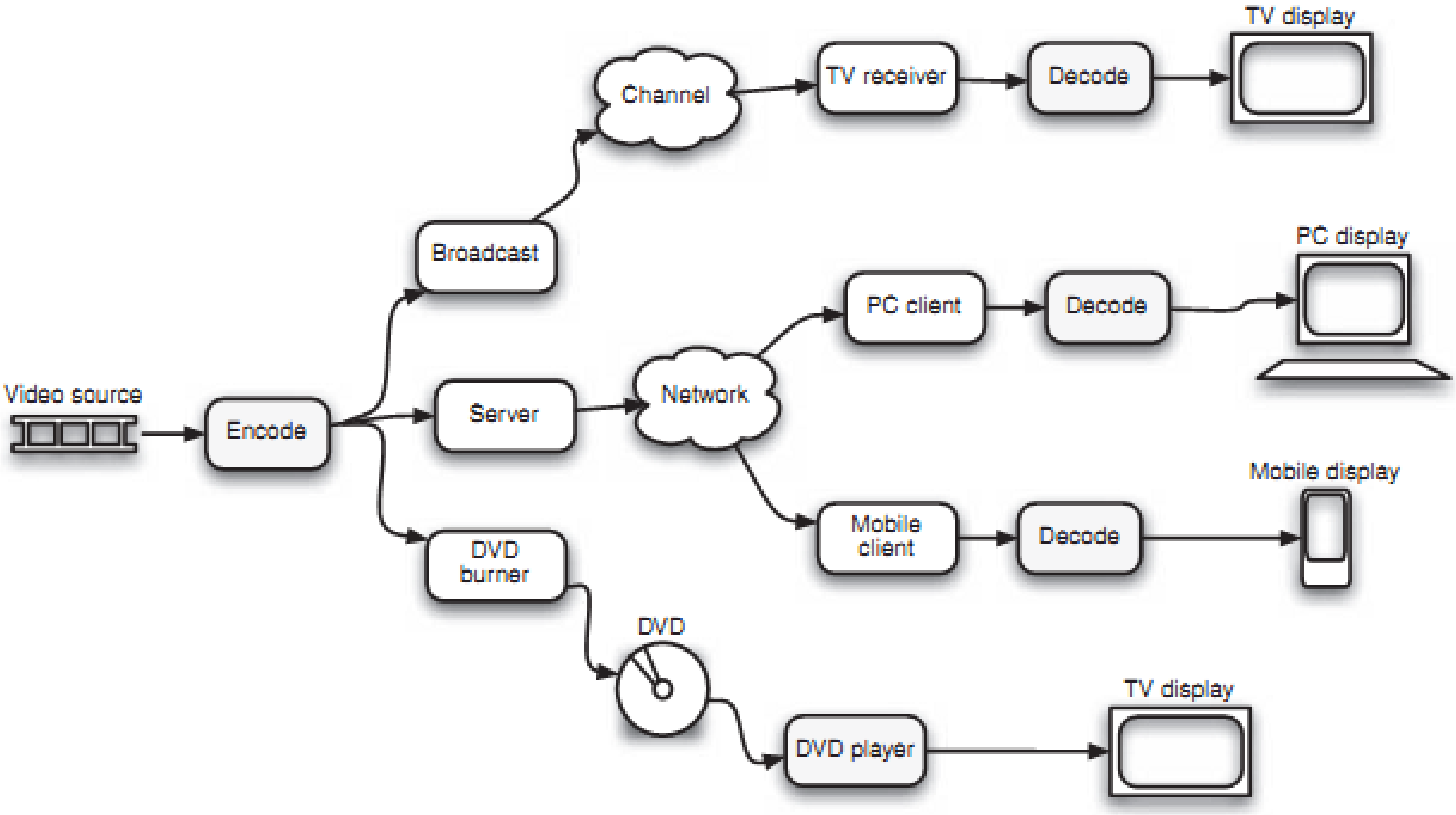
参考书目


THE H.264
ADVANCED VIDEO
COMPRESSION STANDARD
SECOND EDITION
IAIN E. RICHARDSON
WILEY

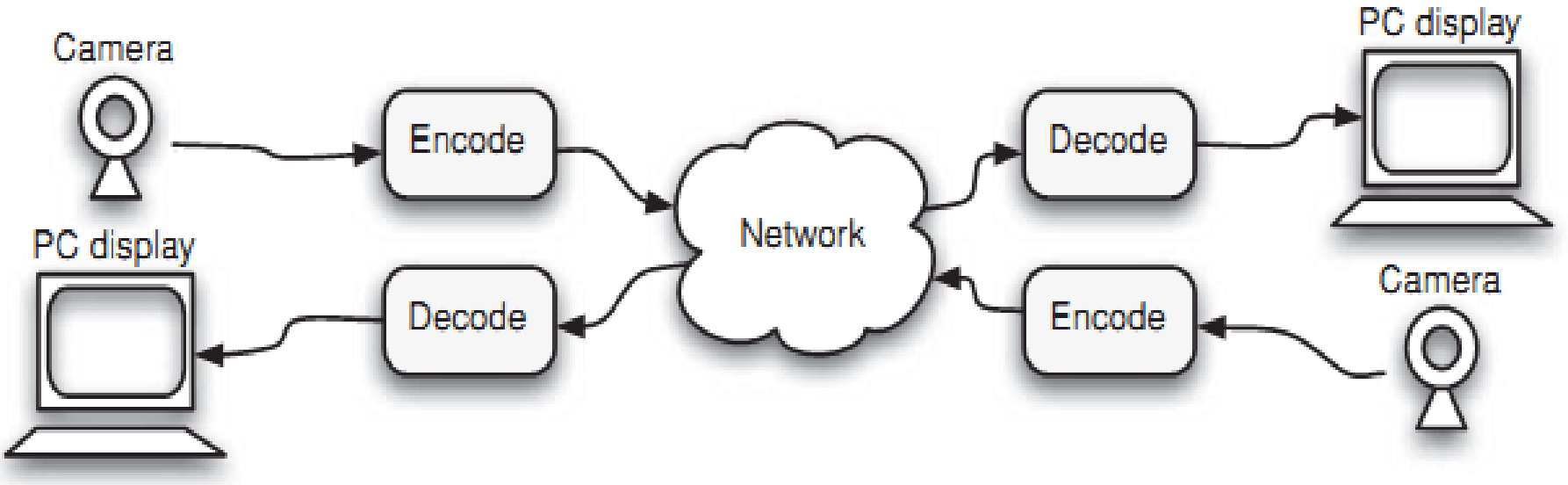

Youn-Long Steve Lin
Chao-Yang Kao
Hung-Chih Kuo
Jian-Wen Chen
VLSI Design for
Video Coding
H.264/AVC Encoding from Standard
Specification to Chip
Springer


国际电信联盟
ITU-T
H.264
(03/2005)
H系列：视听及多媒体系统
视听业务的基础设施 — 活动图像编码
通用视听业务的先进视频编码
ITU-T H.264建议书

Video Format and Quality

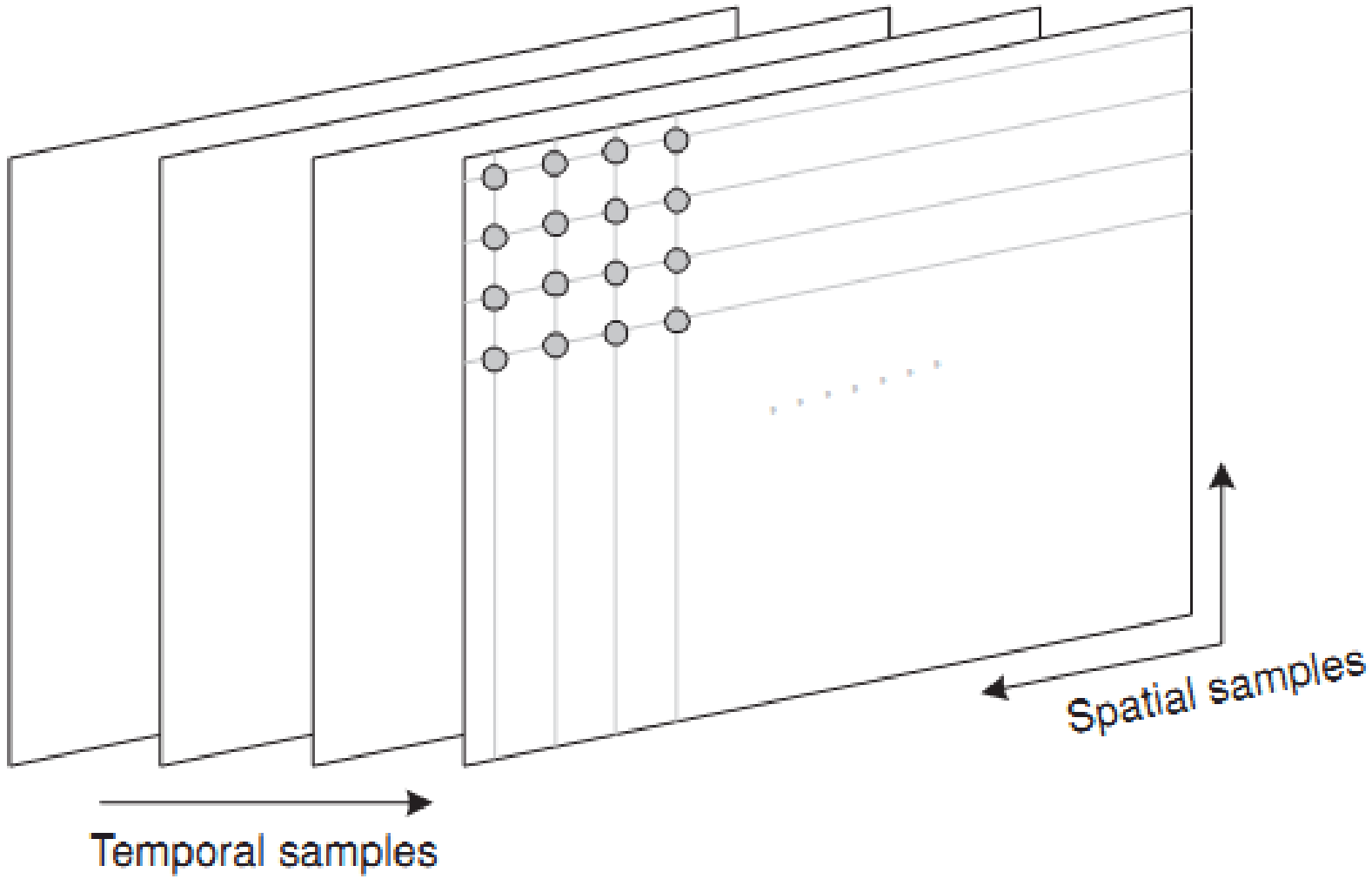


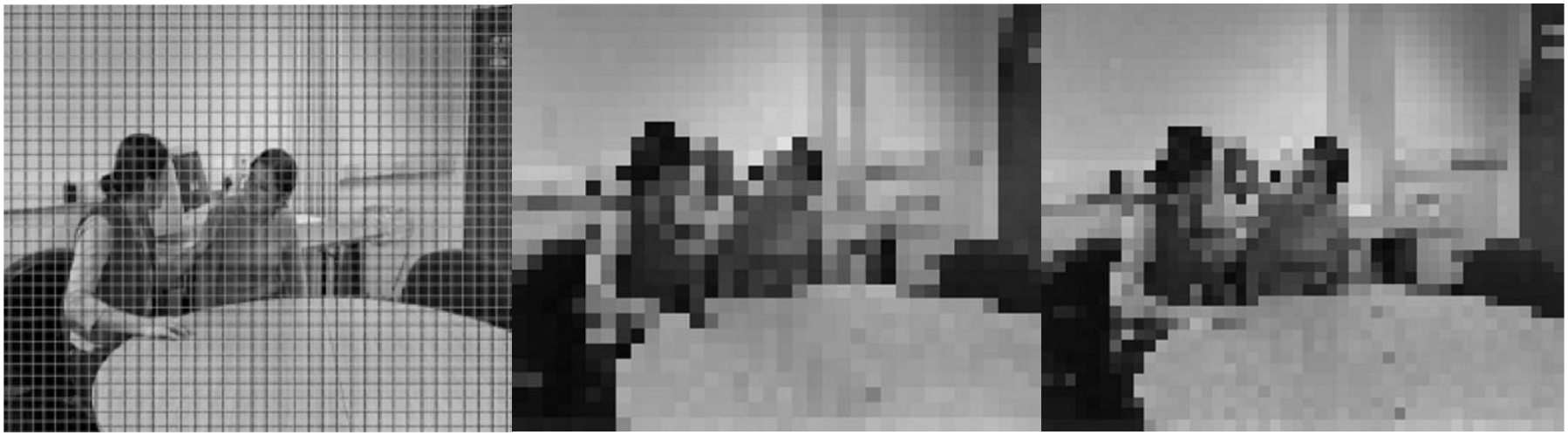
Video coding scenarios, one-way

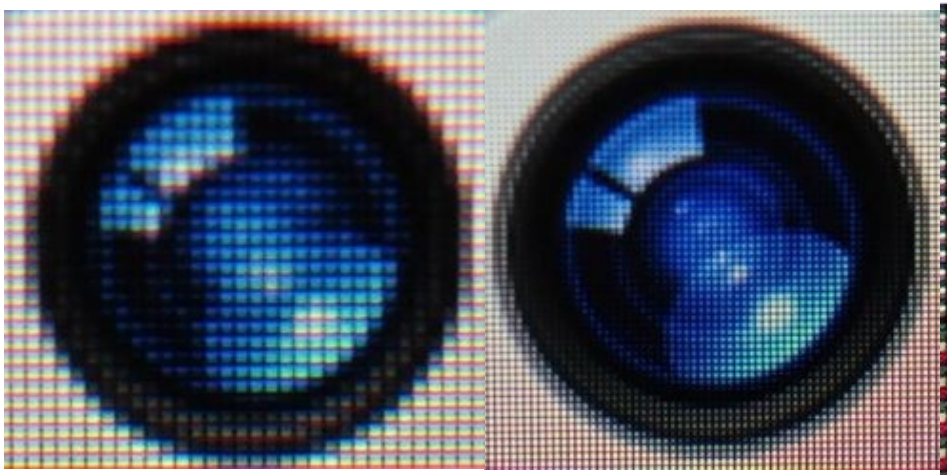


Video coding scenario, two-way

Capture

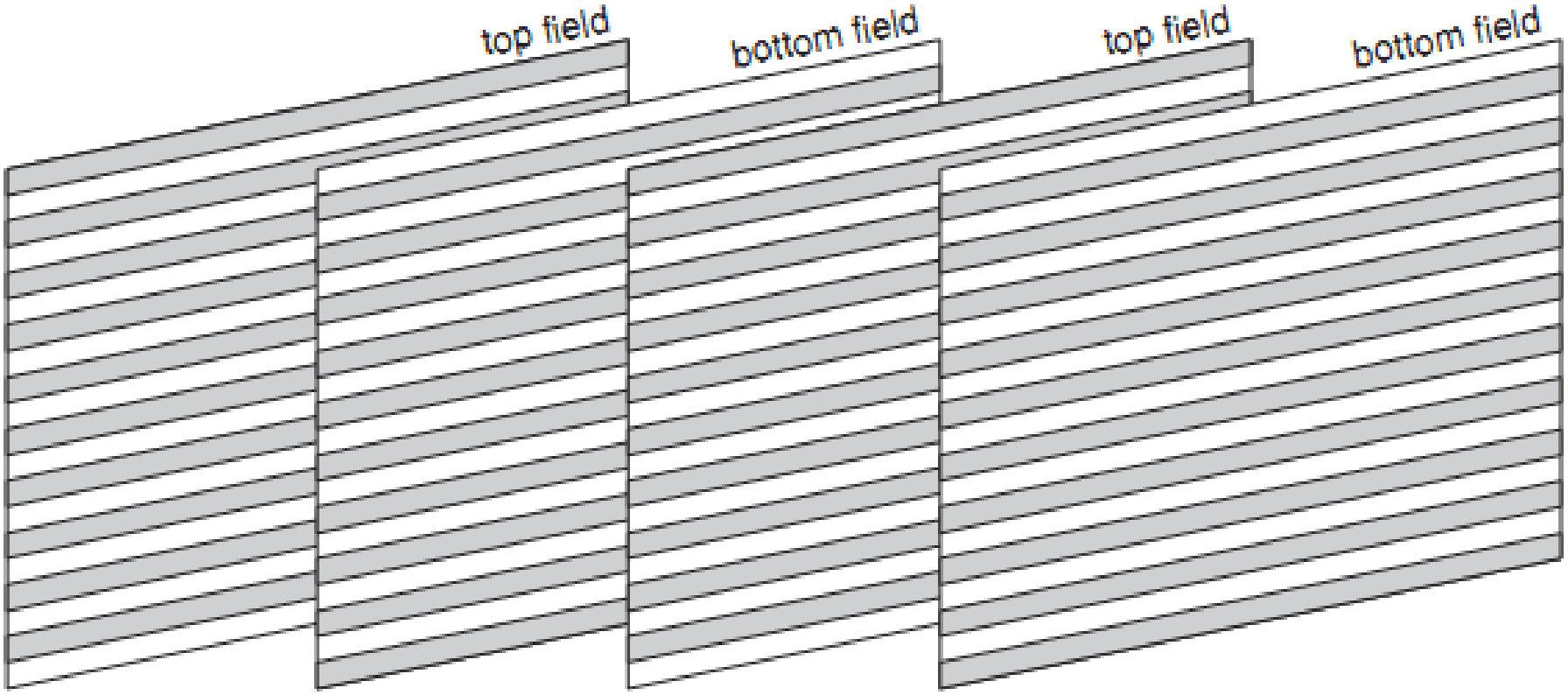






iPad2
1024×768

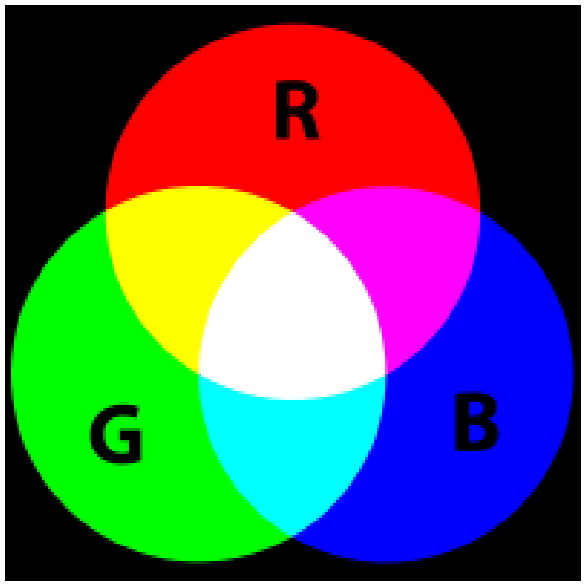
New iPad
2048x1536


top field
bottom field
top field
bottom field

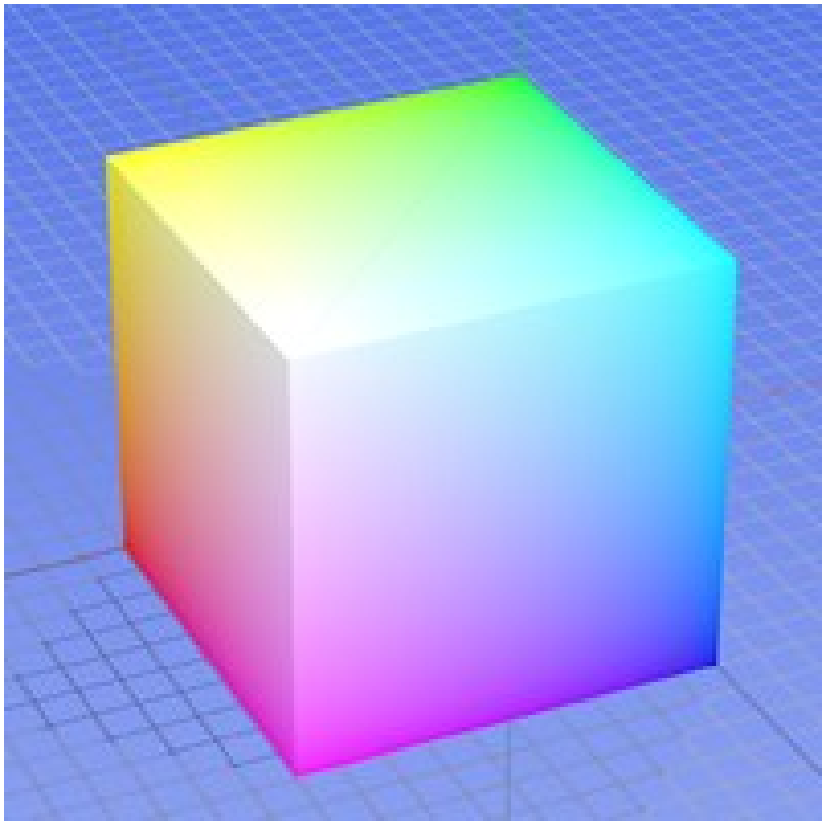
Top field

Bottom field

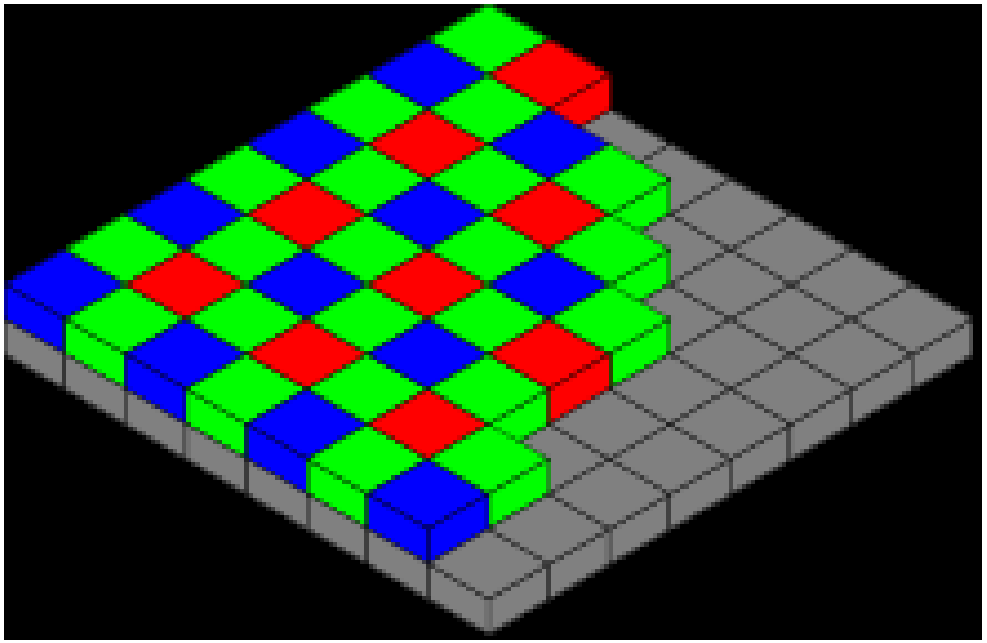
Colour Space



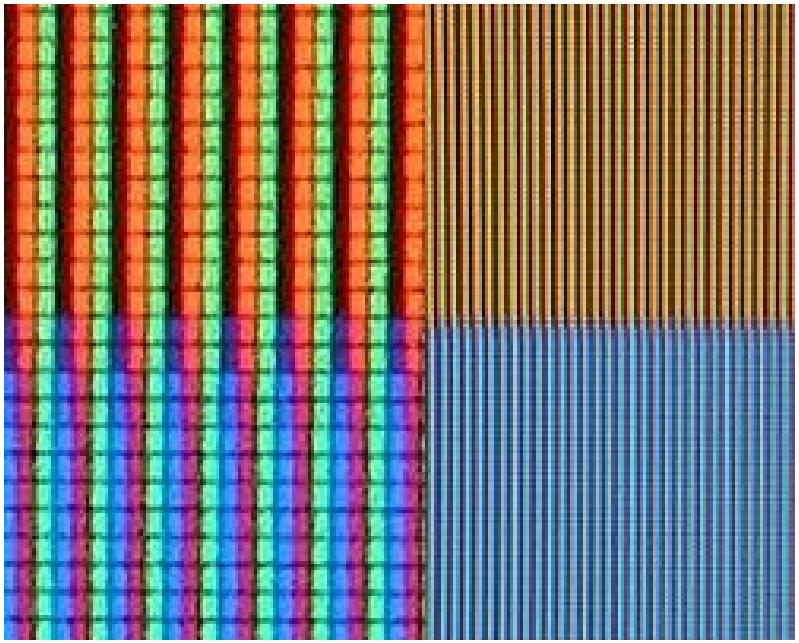
RGB



The RGB color model mapped to a cube. The horizontal x-axis as red values increasing to the left, y-axis as blue increasing to the lower right and the vertical z-axis as green increasing towards the top. The origin, black, is the vertex hidden from view.

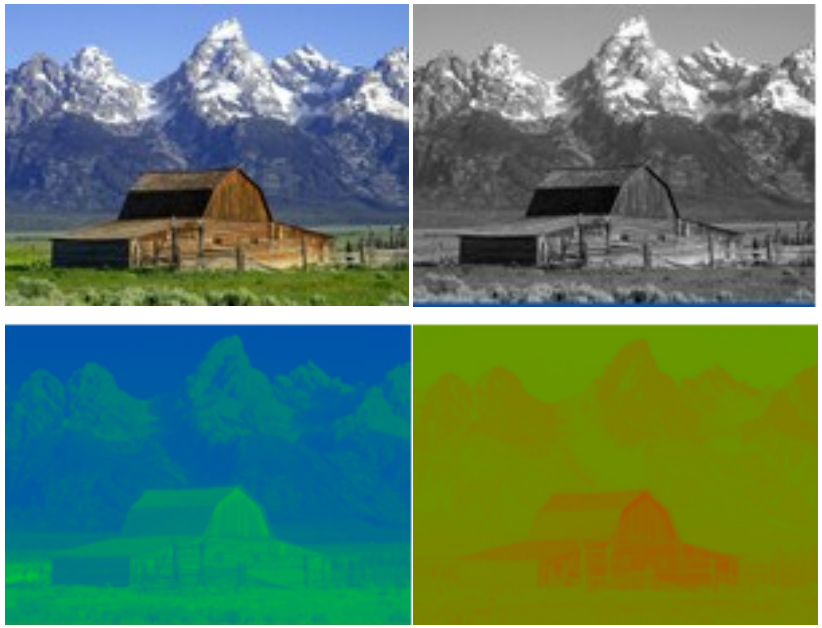


The arrangement of color filters on the pixel array of a digital image sensor

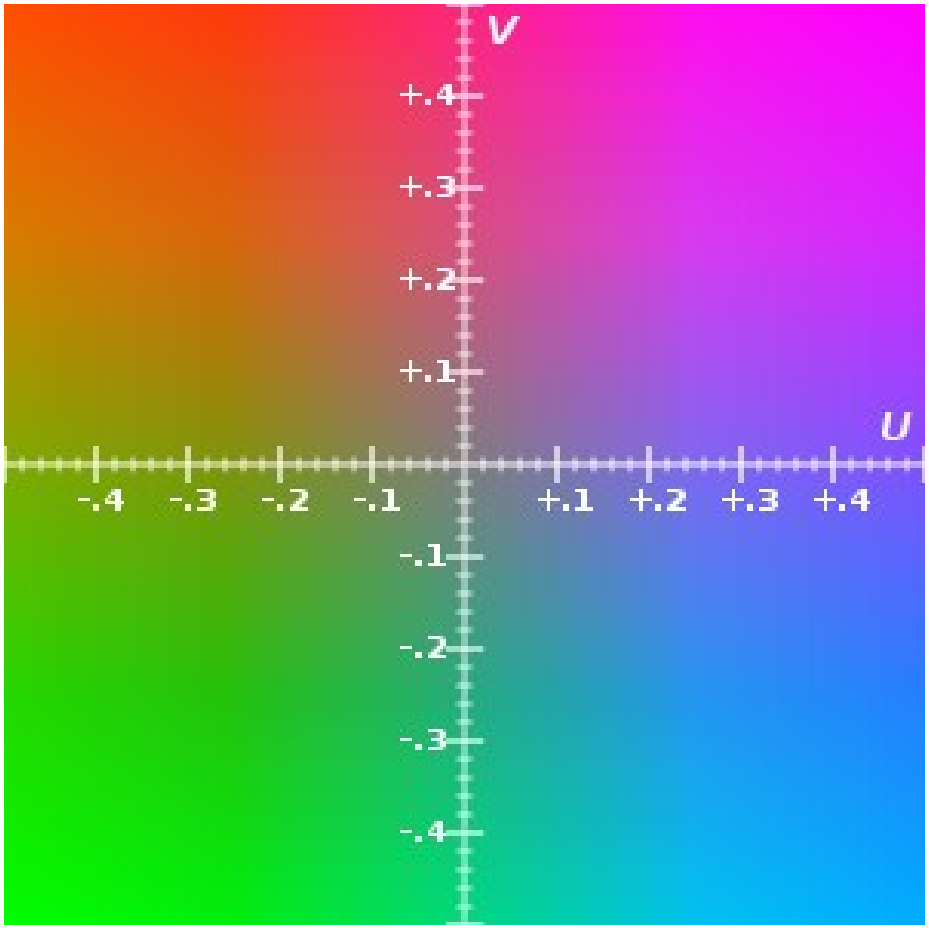


RGB sub-pixels in an LCD TV

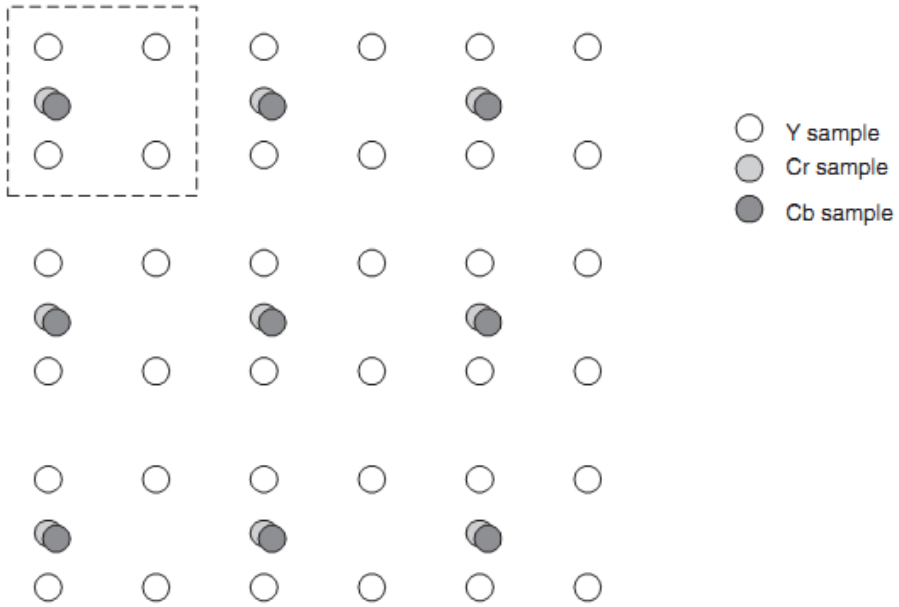
Colour Space



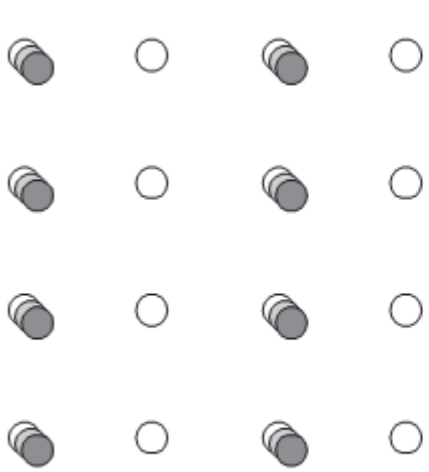
YCbCr (YUV)



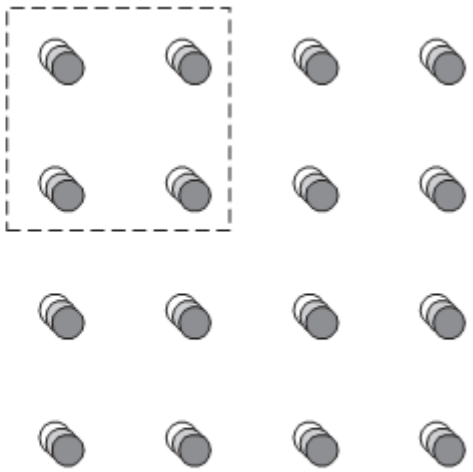
Example of U-V color plane, Y' value = 0.5, represented within RGB color gamut



4:2:0 sampling



4:2:2 sampling



4:4:4 sampling

Table 2.1 Video frame formats

Format	Luminance resolution (horiz. × vert.)	Bits per frame (4:2:0, 8 bits per sample)
Sub-QCIF	128 × 96	147456
Quarter CIF (QCIF)	176 × 144	304128
CIF	352 × 288	1216512
4CIF	704 × 576	4866048

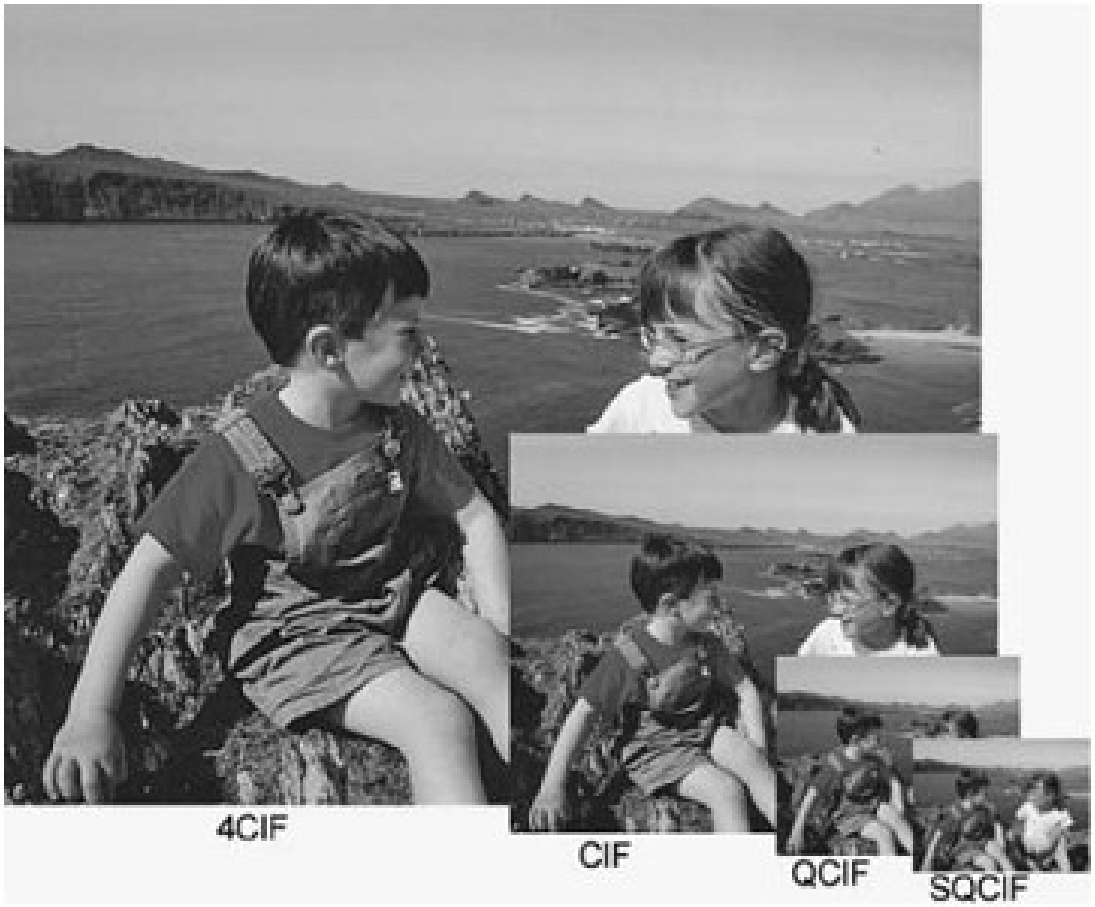


Figure 2.13 Video frame sampled at range of resolutions

Digital Coding Video For Television

Table 2.2 ITU-R BT.601-5 Parameters

	30Hz frame rate	25Hz frame rate
Fields per second	60	50
Lines per complete frame	525	625
Luminance samples per line	858	864
Chrominance samples per line	429	432
Bits per sample	8	8
Total bit rate	216 Mbps	216 Mbps
Active lines per frame	480	576
Active samples per line (Y)	720	720
Active samples per line (Cr,Cb)	360	360
	NTSC	PAL

Table 2.3 HD display formats

Format	Progressive or Interlaced	Horizontal pixels	Vertical pixels	Frames or fields per second
720p	Progressive	1280	720	25 frames
1080i	Interlaced	1920	1080	50 fields
1080p	Progressive	1920	1080	25 frames

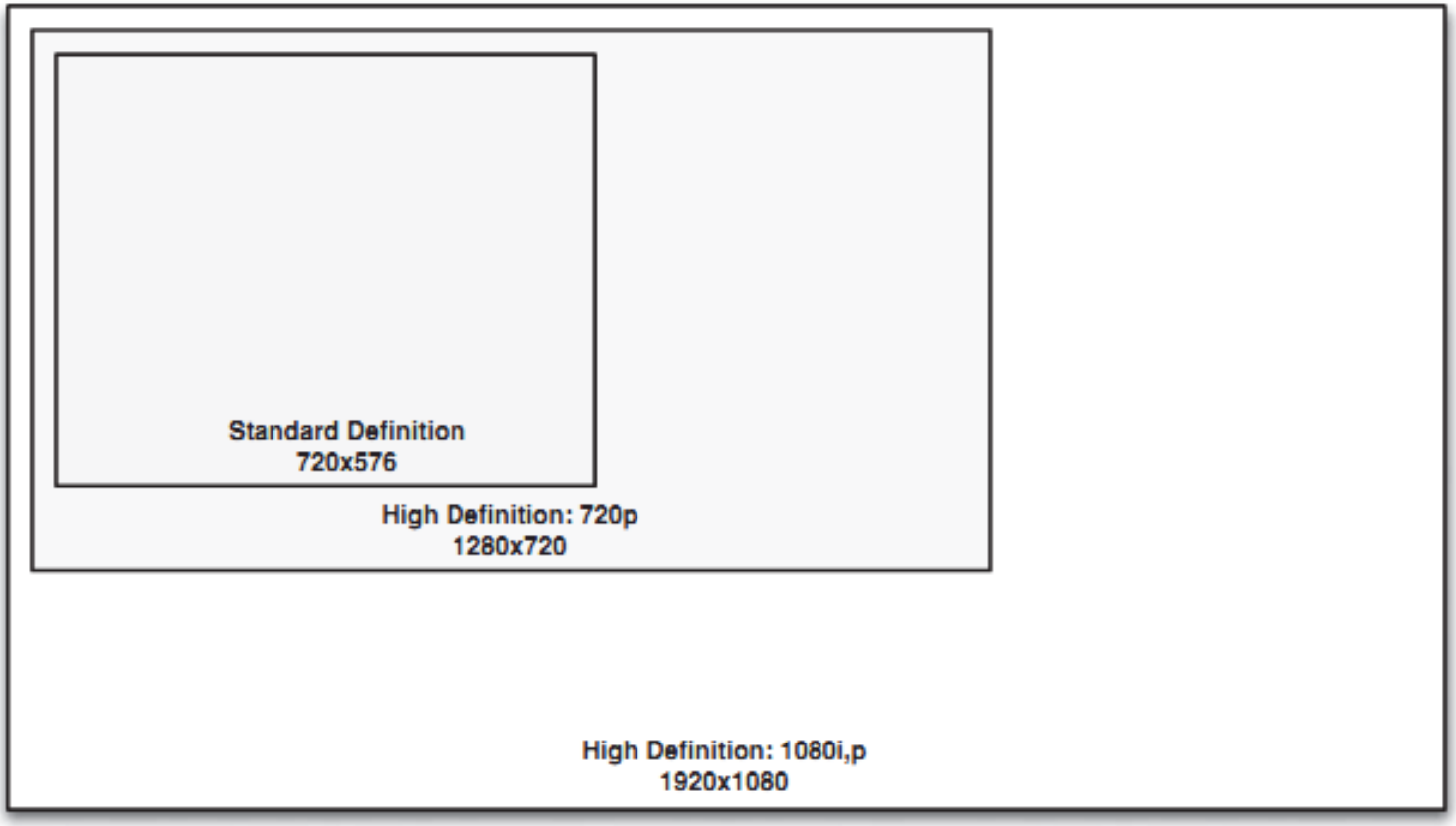


Figure 2.14 SD and HD formats

4Kx2K (4096x2160)

Video Quality Measurement

Subjective Quality Measurement

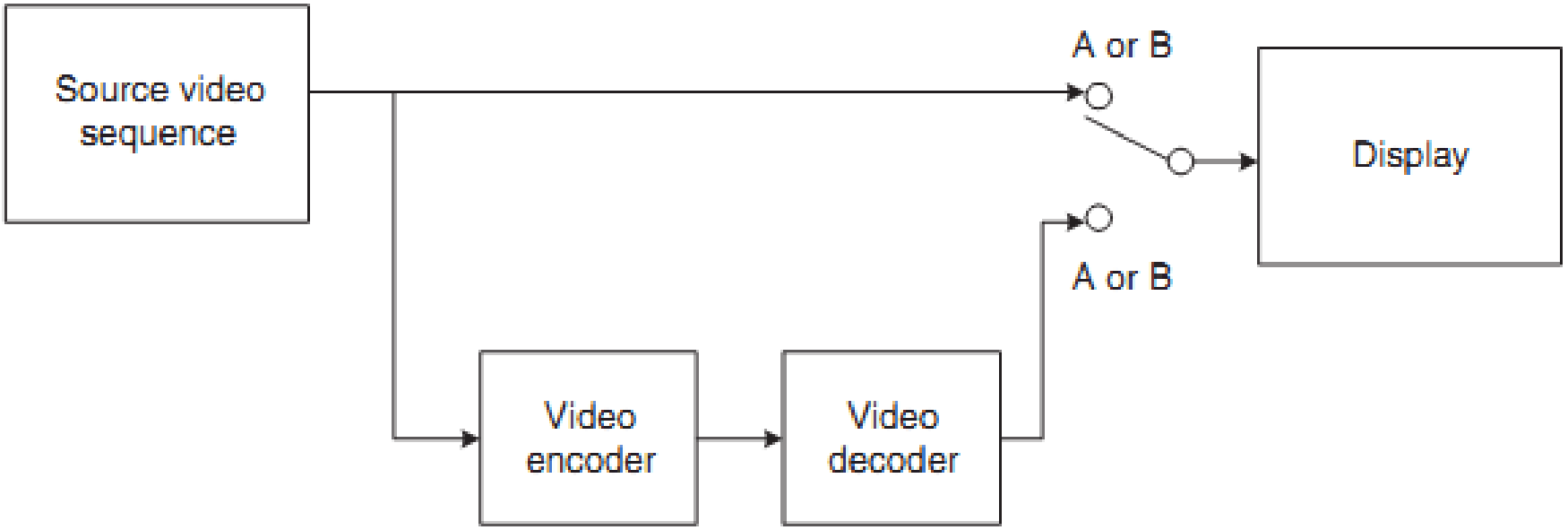


Figure 2.15 DSCQS testing system

ITU-R 500

Objective Quality Measurement

$$PSNR_{dB} = 10 \log_{10} \frac{(2^n - 1)^2}{MSE}$$

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

- MSE：Mean Squared Error between original and impaired image
- n: number of bits per image sample

Original

PSNR 45.53dB

PSNR 36.81dB

PSNR 31.45dB

Video coding concepts

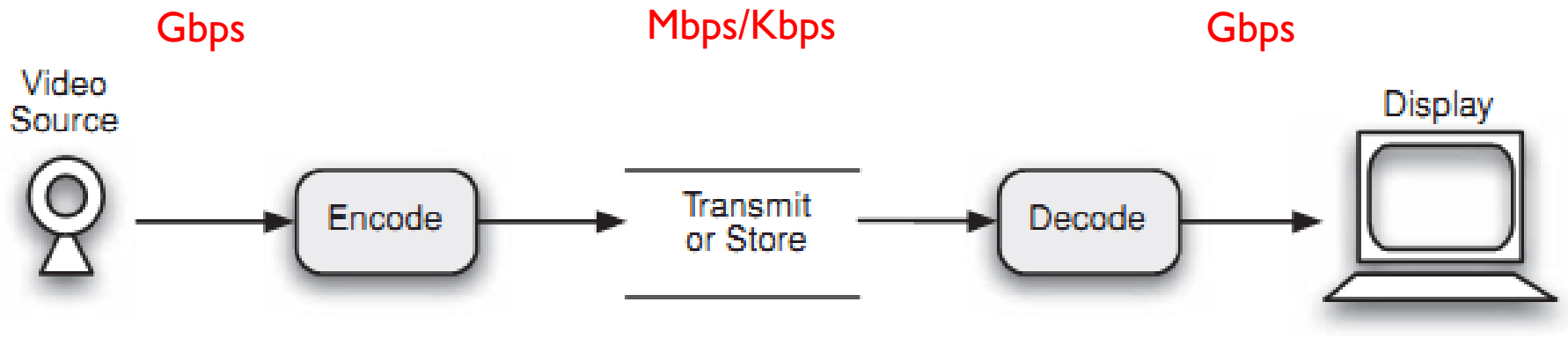
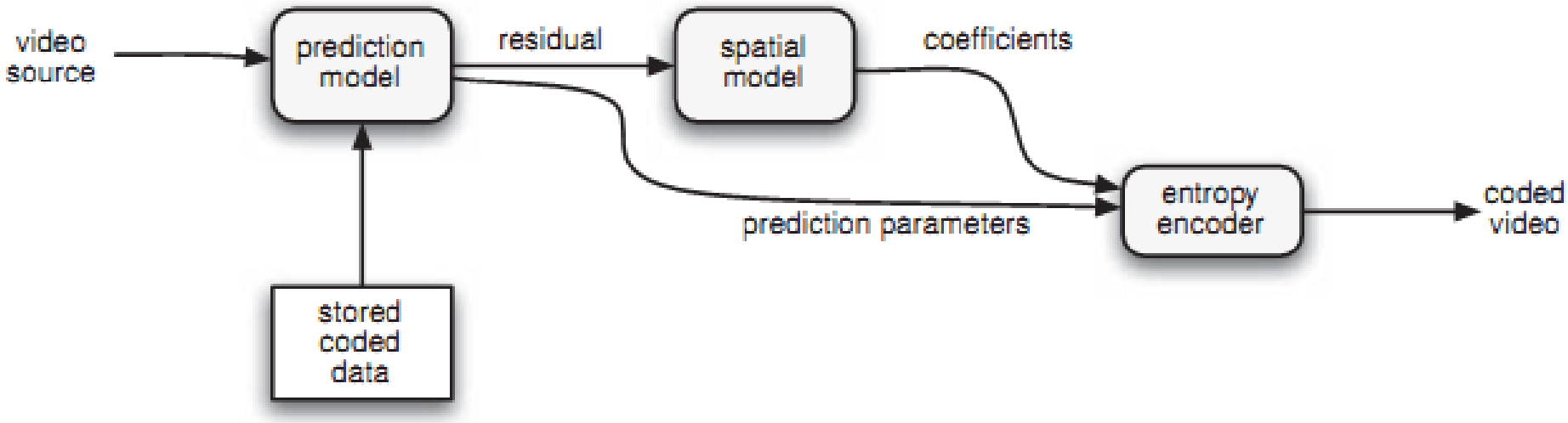


Figure 3.1 Encoder / Decoder


temporal correlation
spatial correlation


video source
prediction model
residual
spatial model
coefficients
entropy encoder
coded video
prediction parameters
stored coded data

Temporal Prediction: Inter Prediction

Figure 3.4 Frame 1

Figure 3.5 Frame 2

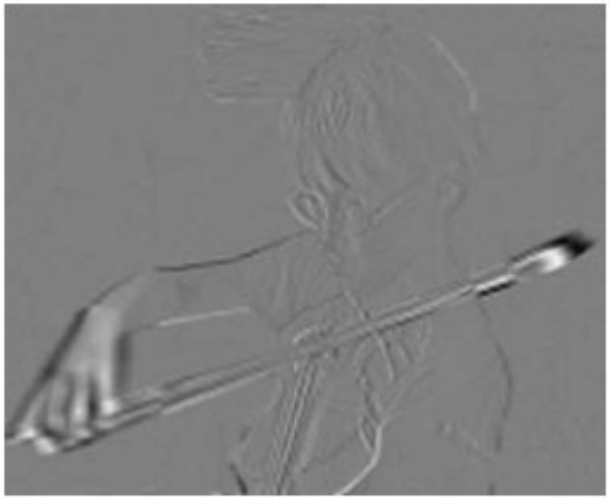


Figure 3.6 Difference

Figure 3.7 Optical flow

Block Based Motion Estimation and Compensation

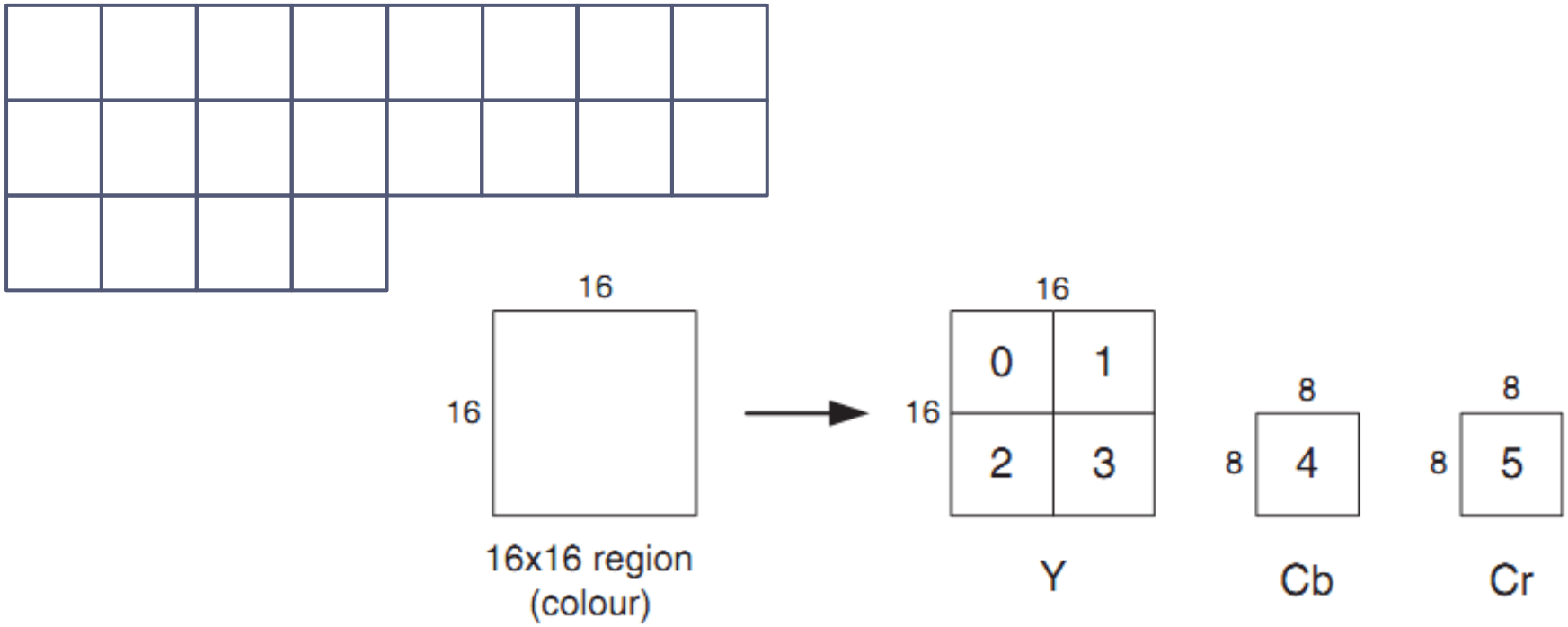


Figure 3.8 Macroblock (4:2:0)

Motion Estimation

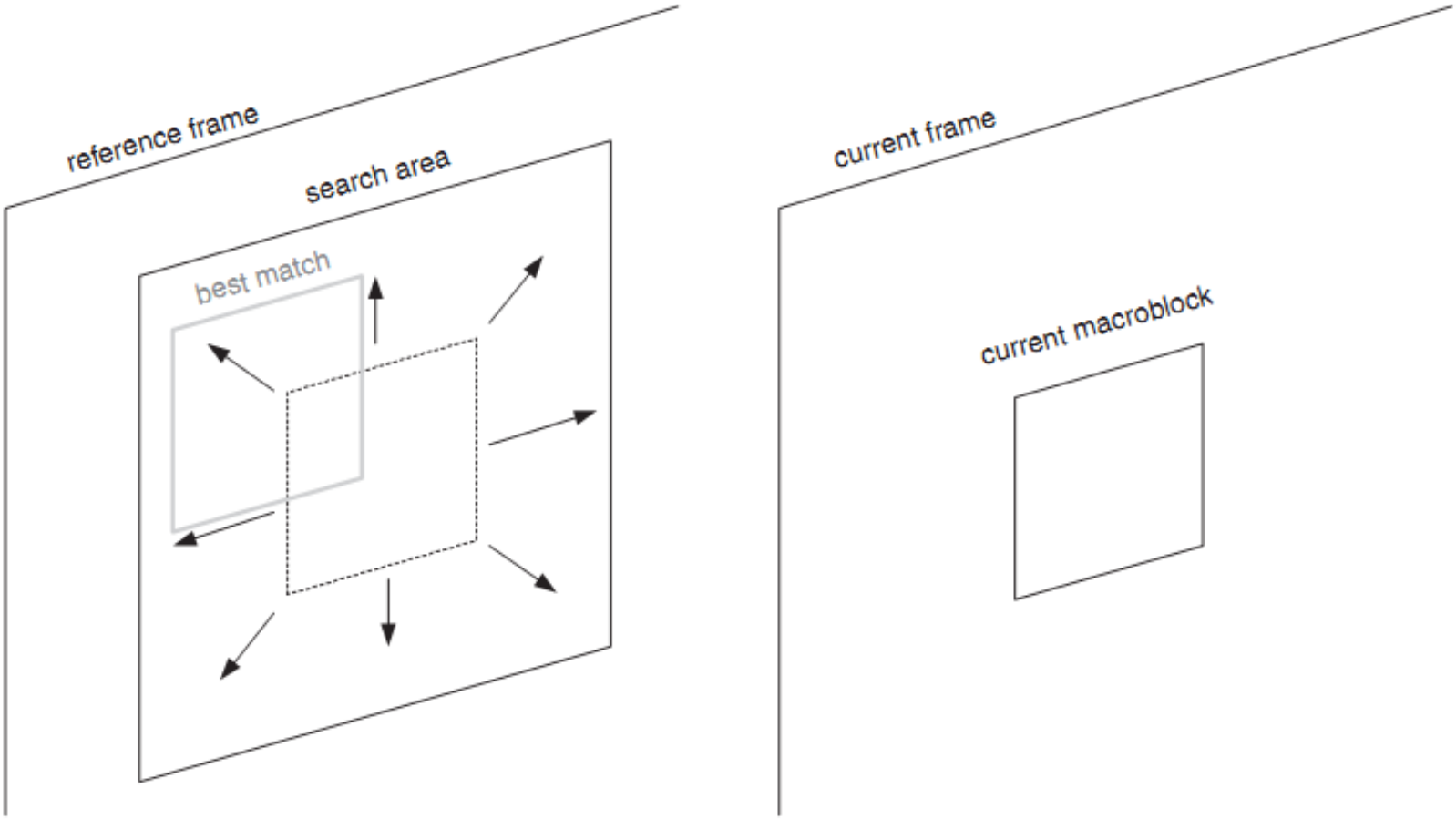


Figure 3.9 Motion estimation

Motion Compensation and Block Size

Figure 3.10 Frame 1

Figure 3.11 Frame 2

Figure 3.12 Residual : no motion compensation

Figure 3.13 Residual : 16 × 16 block size

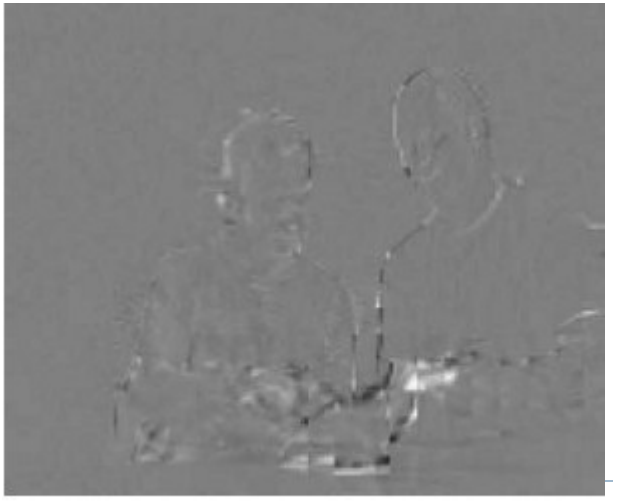


Figure 3.14 Residual : 8 × 8 block size

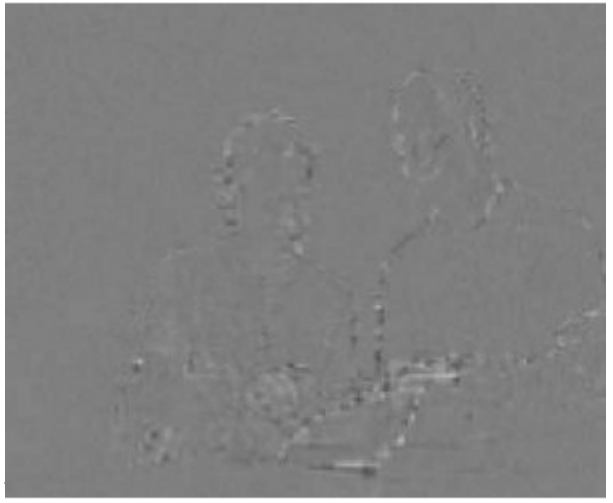


Figure 3.15 Residual : 4 × 4 block size

Sub-Pixel Motion Compensation

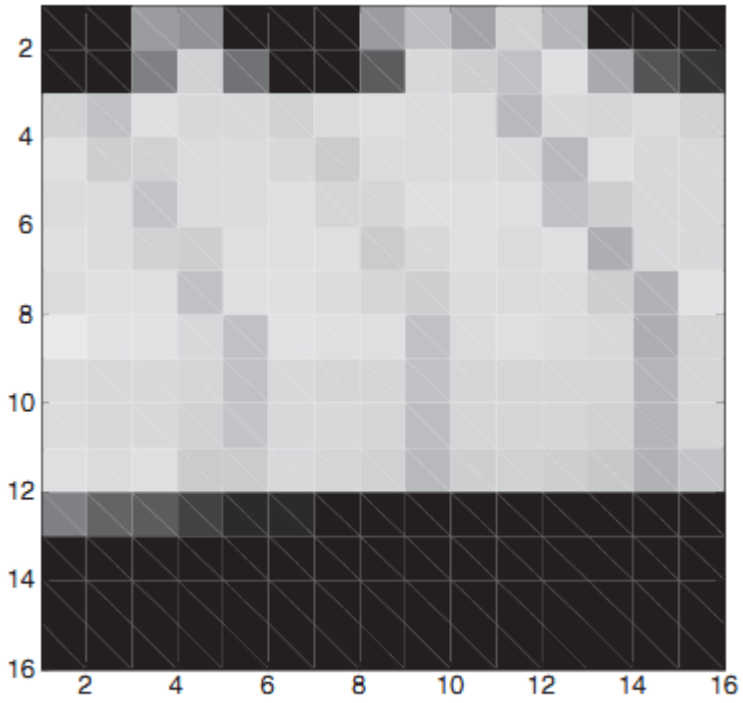


Figure 3.16 Close-up of reference region

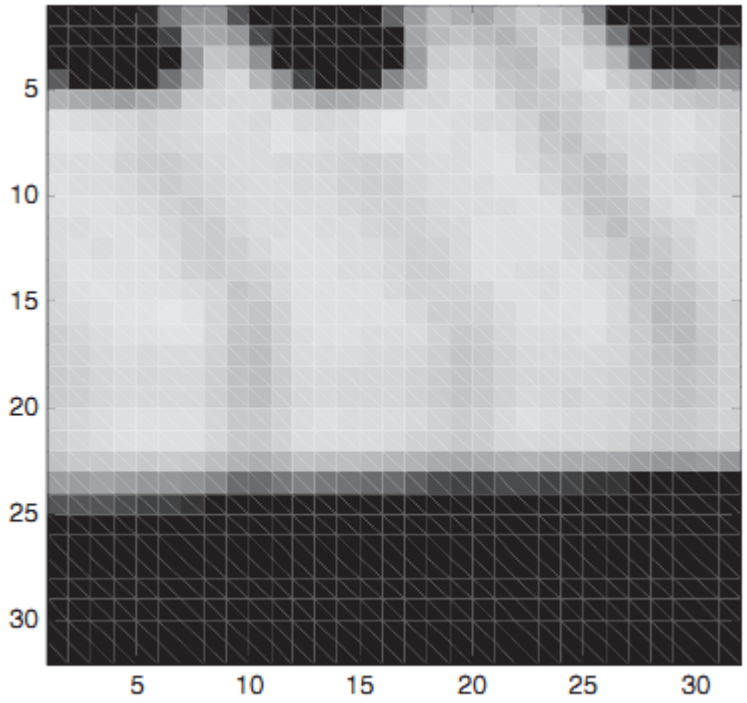


Figure 3.17 Reference region interpolated to half-pixel positions

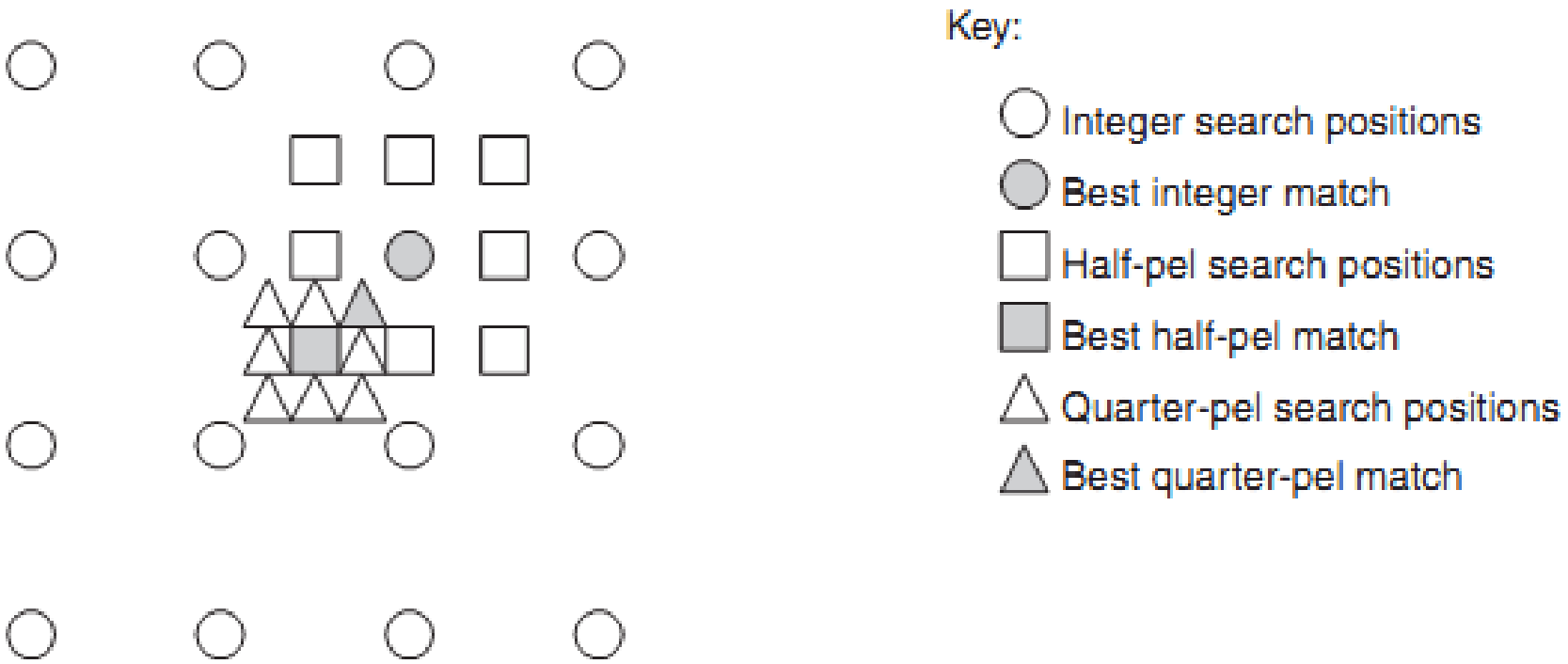


Figure 3.18 Integer, half-pixel and quarter-pixel motion estimation

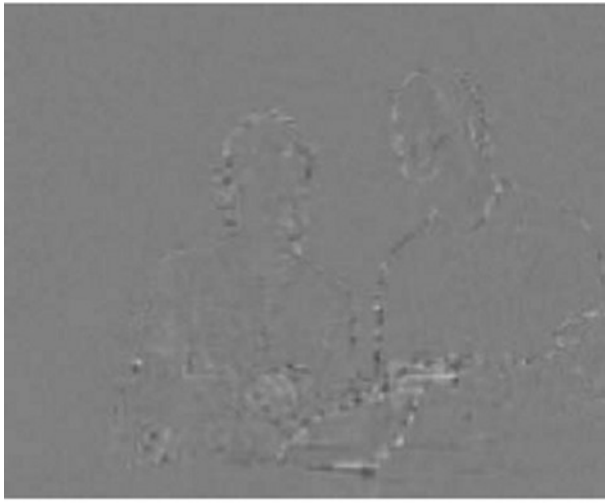


Figure 3.15 Residual : 4 × 4 block size

Table 3.1 SAE of residual frame after motion compensation, 16 × 16 block size

Sequence	No motion compensation	Integer-pel	Half-pel	Quarter-pel
'Violin', QCIF	171945	153475	128320	113744
'Grasses', QCIF	248316	245784	228952	215585
'Carphone', QCIF	102418	73952	56492	47780

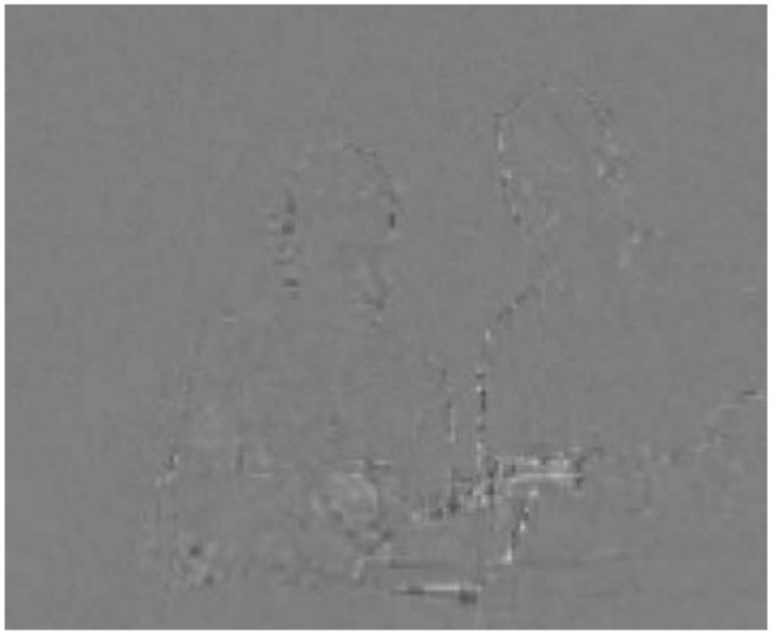


Figure 3.19 Residual : 4 × 4 blocks, 1/2-pixel compensation

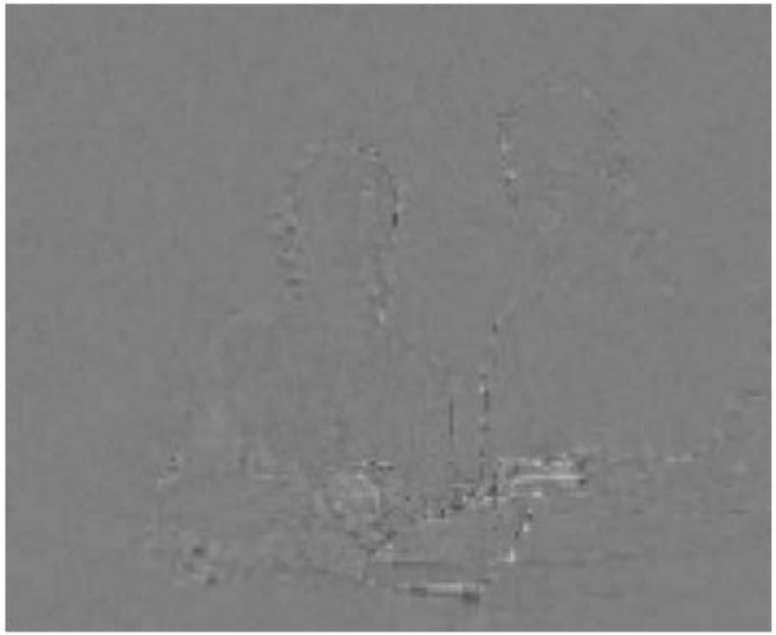


Figure 3.20 Residual : 4 × 4 blocks, 1/4-pixel compensation

Motion Vector

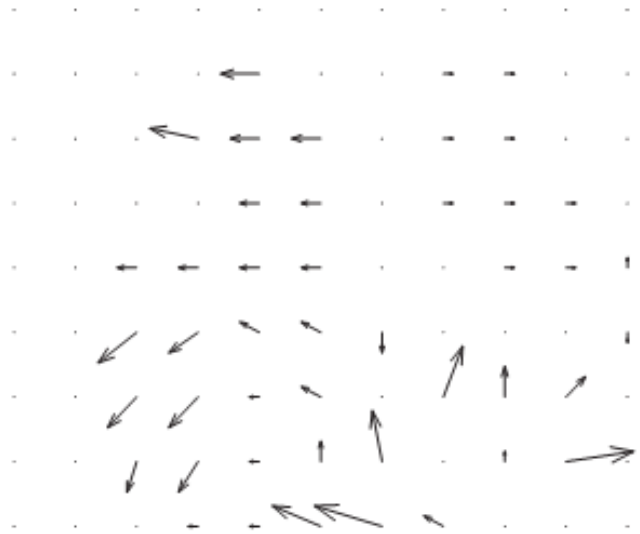


Figure 3.21 Motion vector map : 16 × 16 blocks, integer vectors

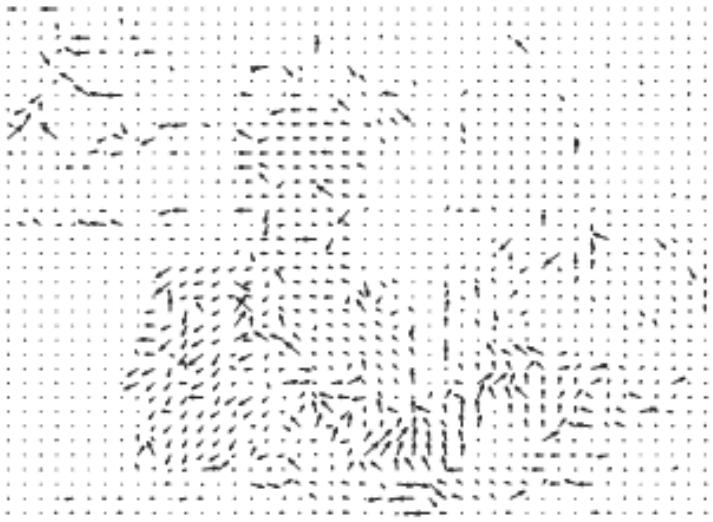


Figure 3.22 Motion vector map : 4 × 4 blocks, 1/4-pixel vectors

Spatial Prediction: Intra Prediction

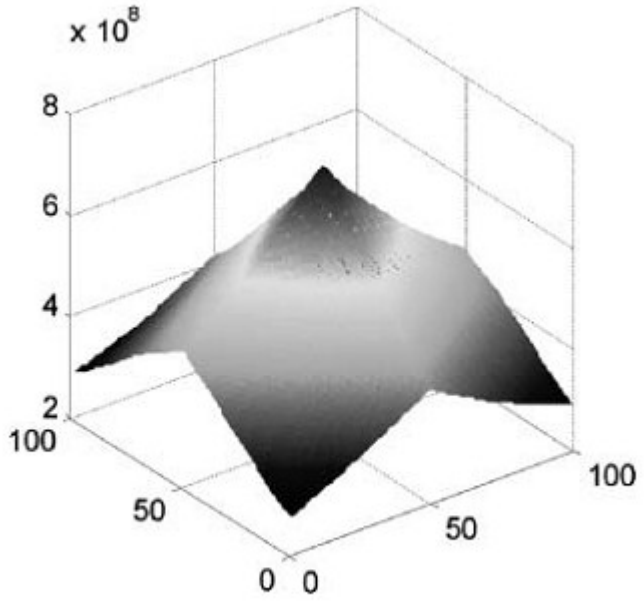


Figure 3.25 2D autocorrelation function of image

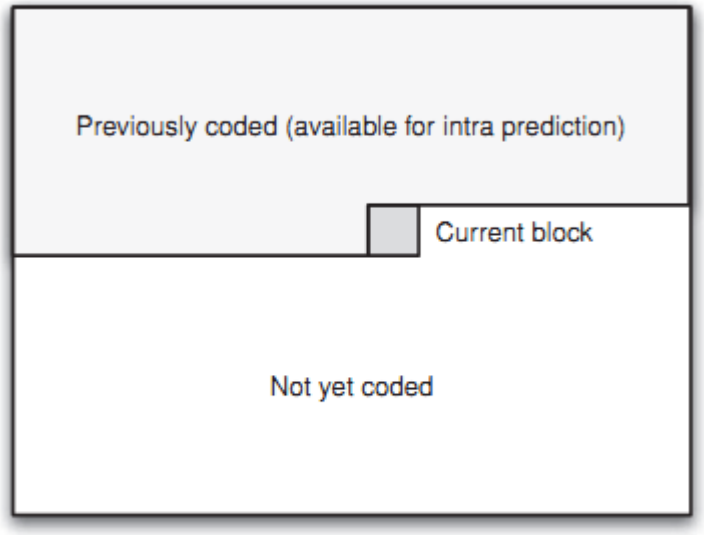


Figure 3.23 Intra prediction: available samples

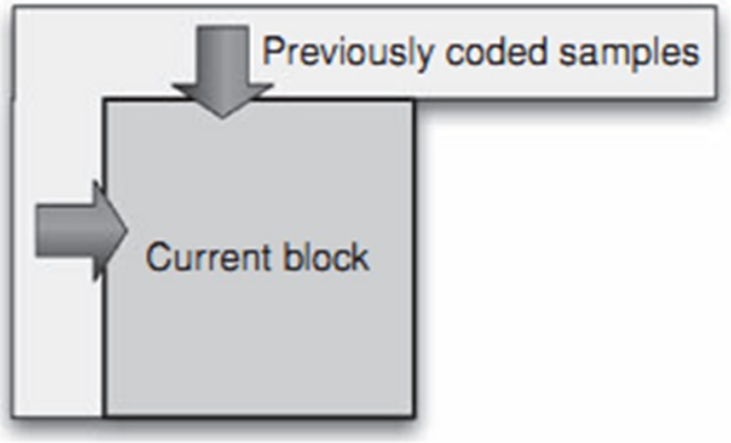
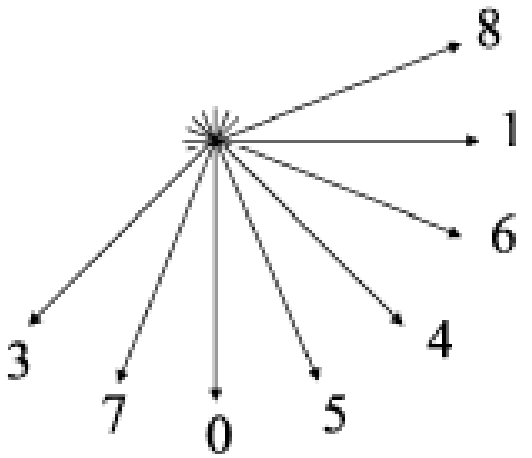


Figure 3.24 Intra prediction: spatial extrapolation

Q	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

(a)



(b)

Transform Coding

Discrete Cosine Transform (DCT)

DCT: $\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}}$

IDCT: $\mathbf{X} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}\mathbf{A}$

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \quad \text{where } C_i = \sqrt{\frac{1}{N}}\ (\mathrm{i} = 0),\ C_i = \sqrt{\frac{2}{N}}\ (\mathrm{i} > 0)$$

X is a matrix of samples.

Y is a matrix of coefficients

A is an N × N transform matrix

Example: N = 4

The transform matrix **A** for a 4 × 4 DCT is:

$$A = \begin{bmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix}$$

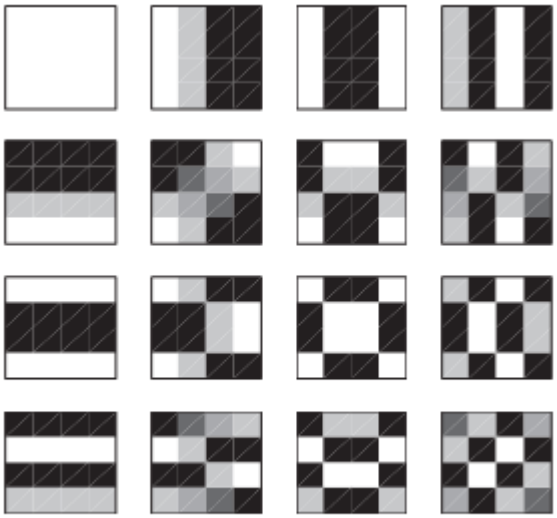


Figure 3.28 4 × 4 DCT basis patterns

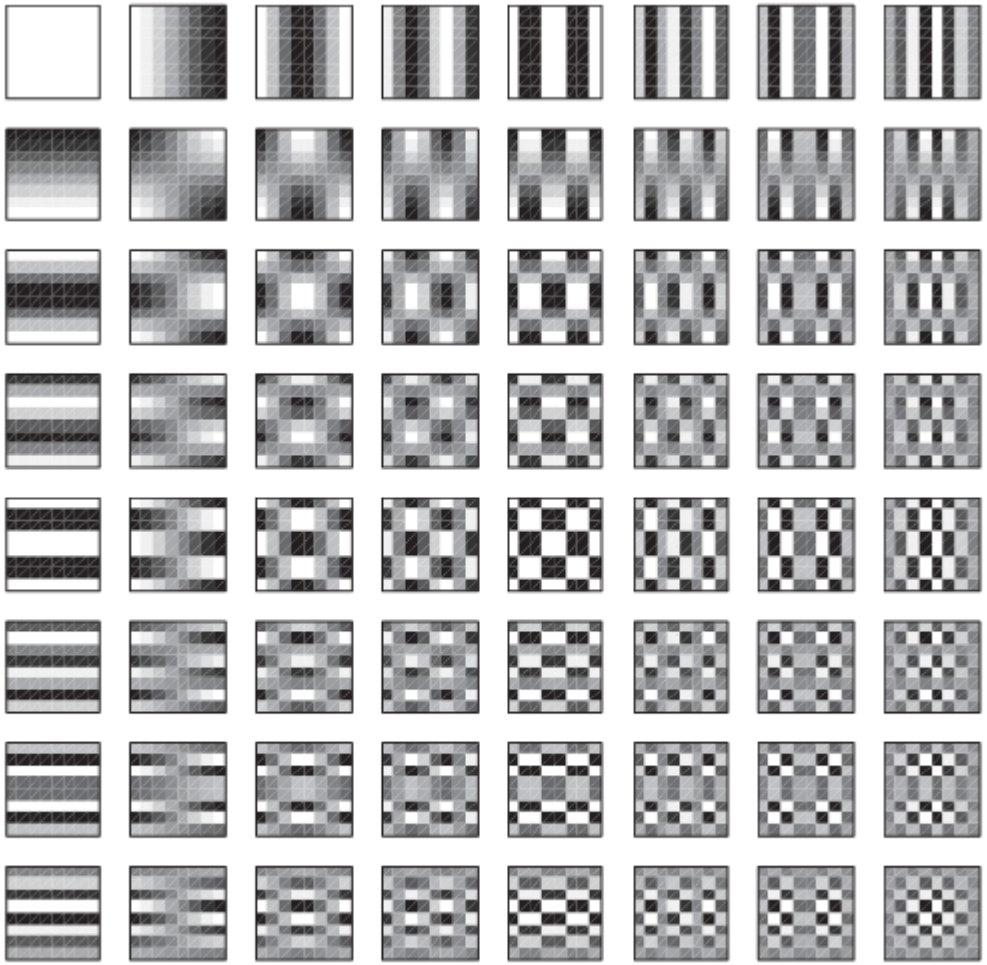


Figure 3.29 8 × 8 DCT basis patterns

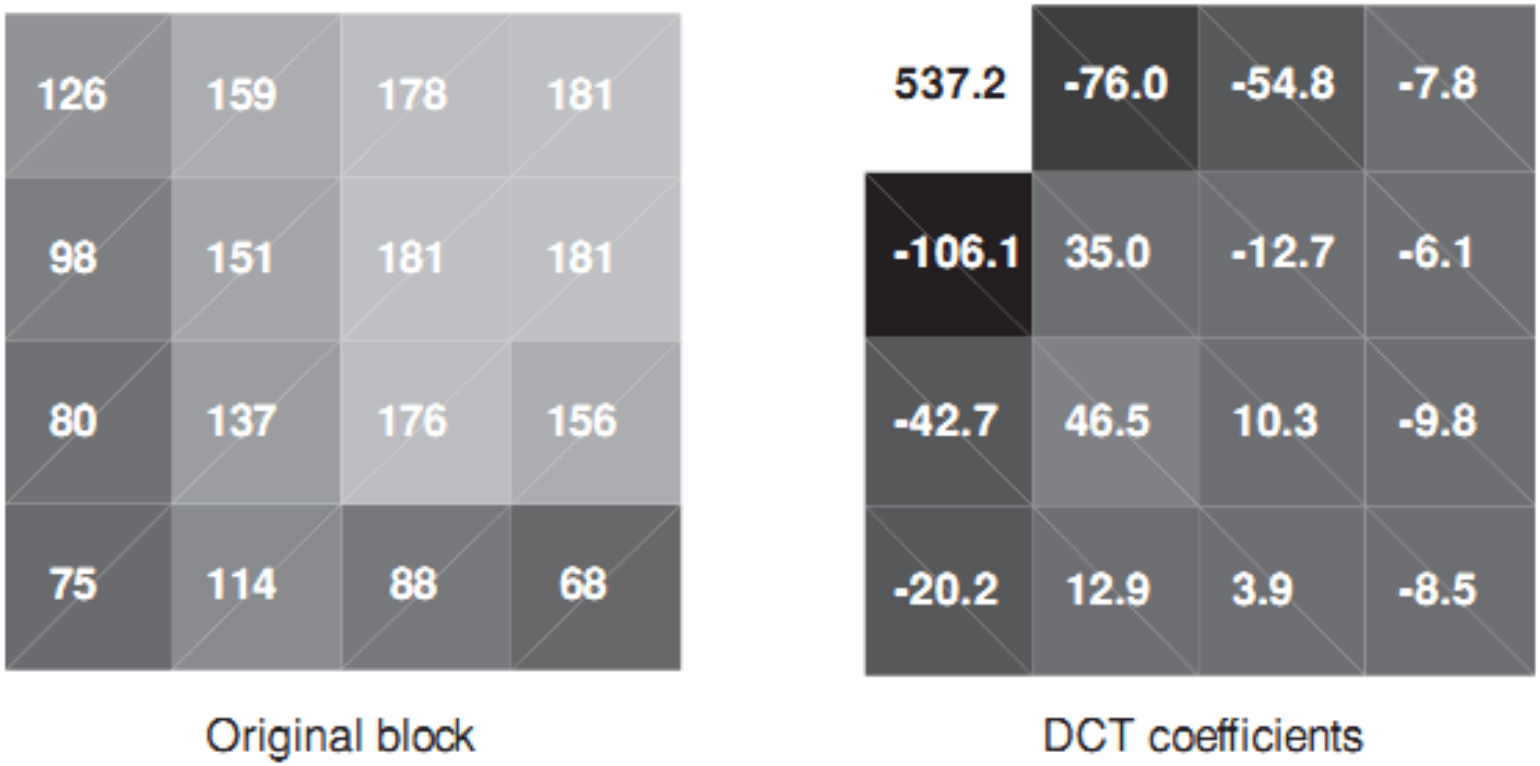


Figure 3.31 Close-up of 4 × 4 block; DCT coefficients

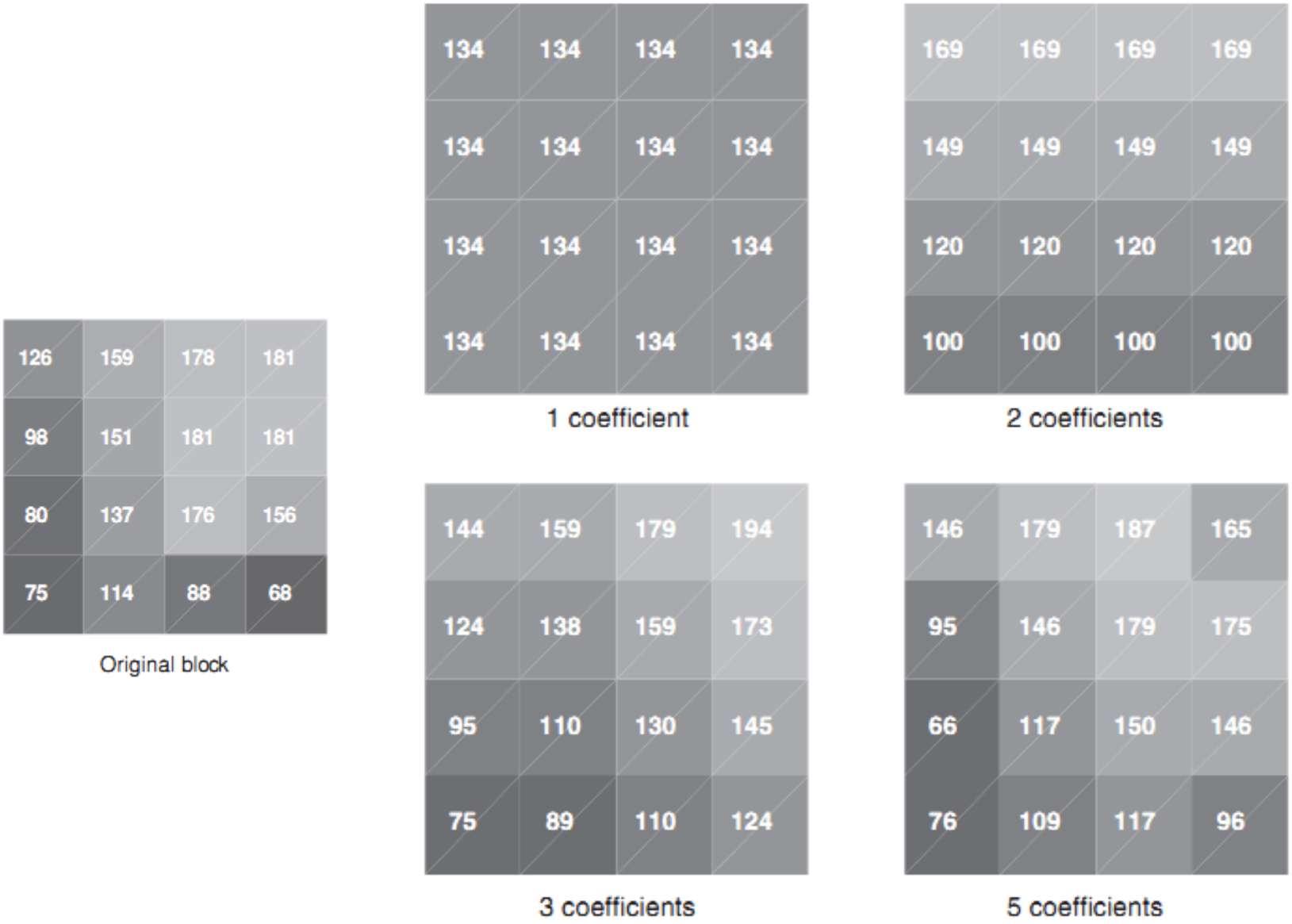


Figure 3.32 Block reconstructed from (a) 1, (b) 2, (c) 3, (d) 5 coefficients

Wavelet

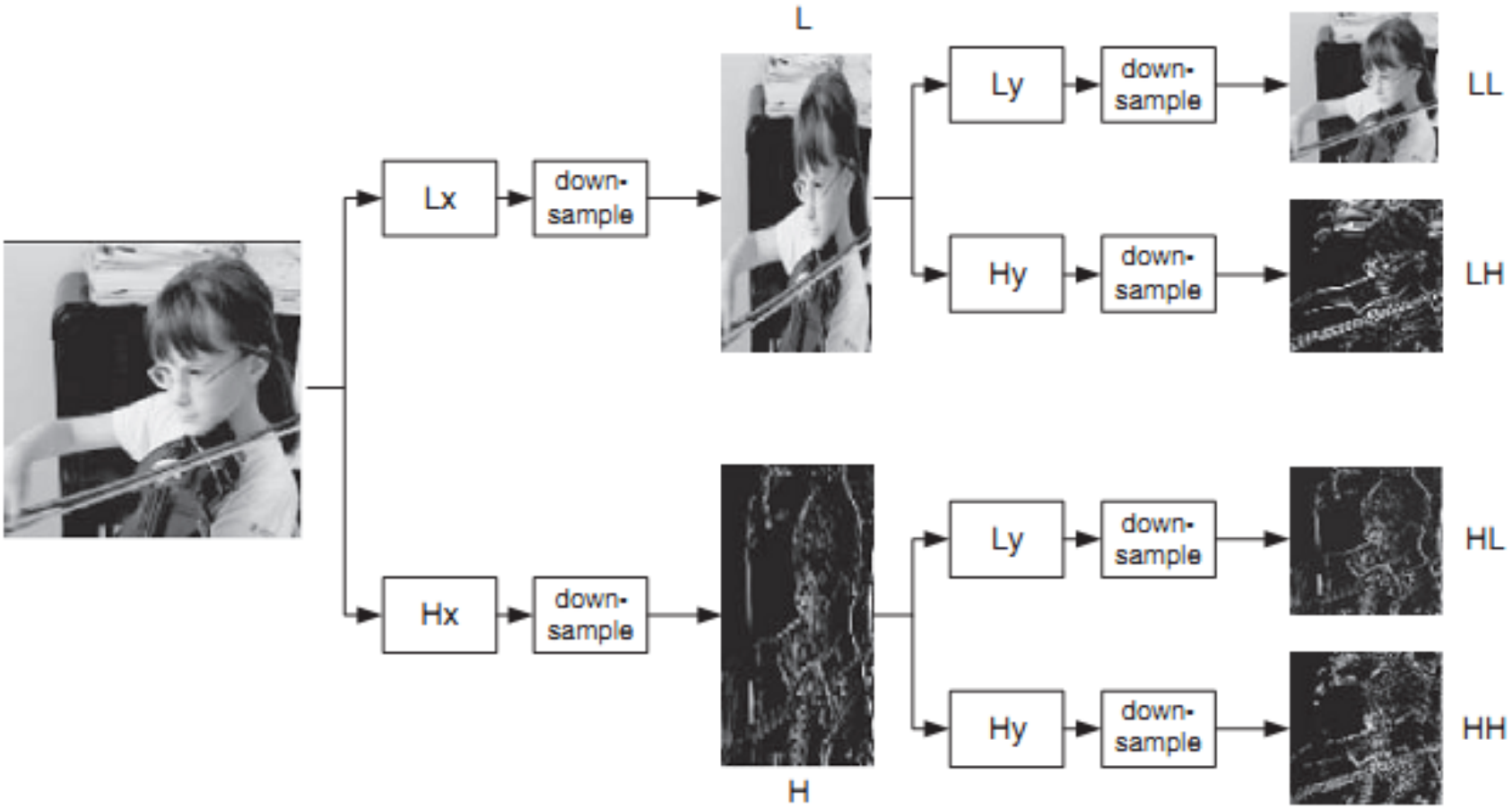


Figure 3.33 Two-dimensional wavelet decomposition process

Figure 3.34 Image after one level of decompositio

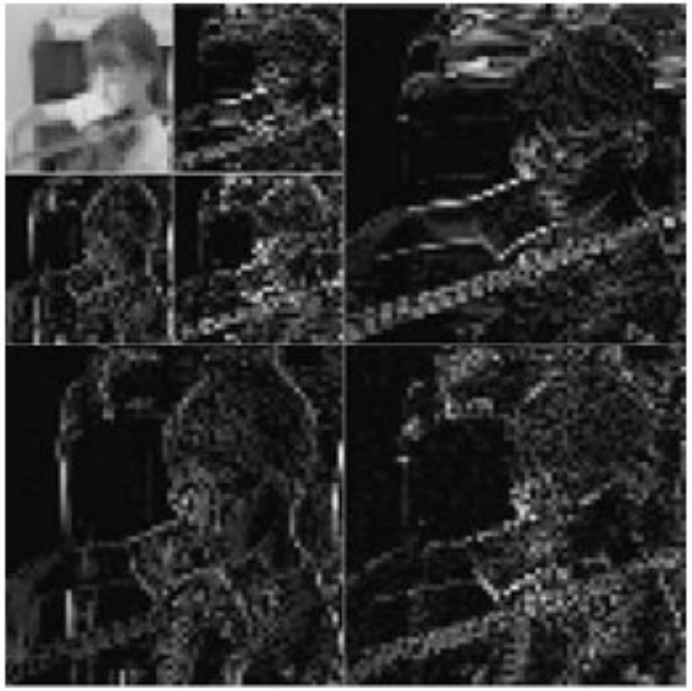


Figure 3.35 Two-stage wavelet decomposition of image

Quantization

$$FQ = round\left(\frac{X}{QP}\right)$$

$$Y = FQ.QP$$

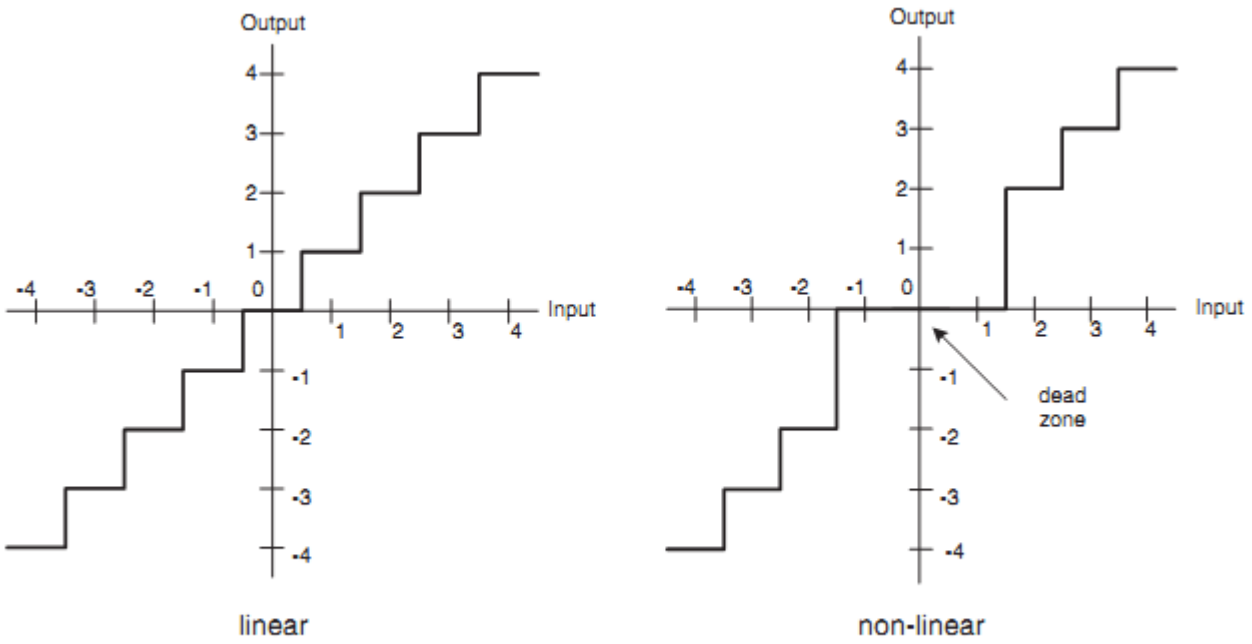


Figure 3.37 Scalar quantizers: linear; non-linear with dead zone

Example Y = QP.round(X/QP)

	Y			
X	QP=1	QP=2	QP=3	QP=5
−4	−4	−4	−3	−5
−3	−3	−2	−3	−5
−2	−2	−2	−3	0
−1	−1	0	0	0
0	0	0	0	0
1	1	0	0	0
2	2	2	3	0
3	3	2	3	5
4	4	4	3	5
5	5	4	6	5
6	6	6	6	5
7	7	6	6	5
8	8	8	9	10
9	9	8	9	10
10	10	10	9	10
11	11	10	12	10
.....				

Reordering

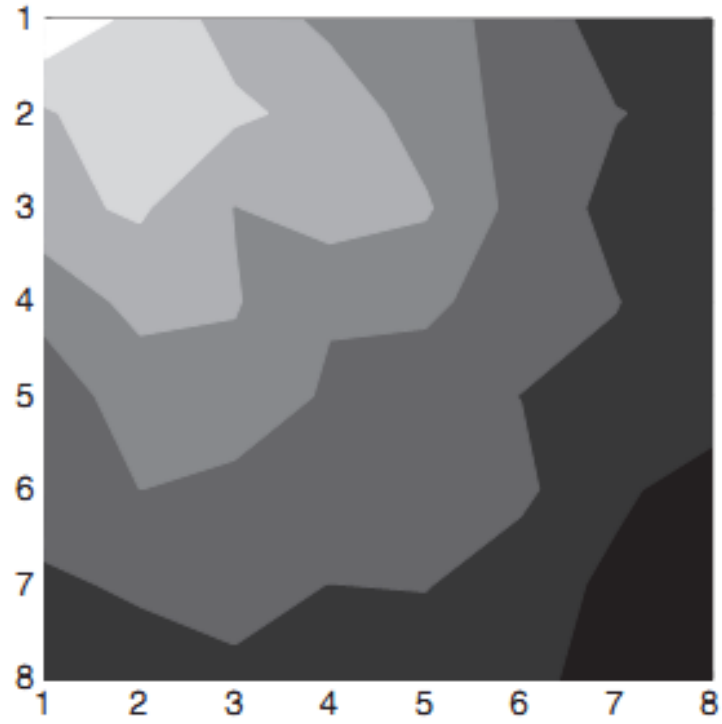


Figure 3.39 8 × 8 DCT coefficient distribution (frame)

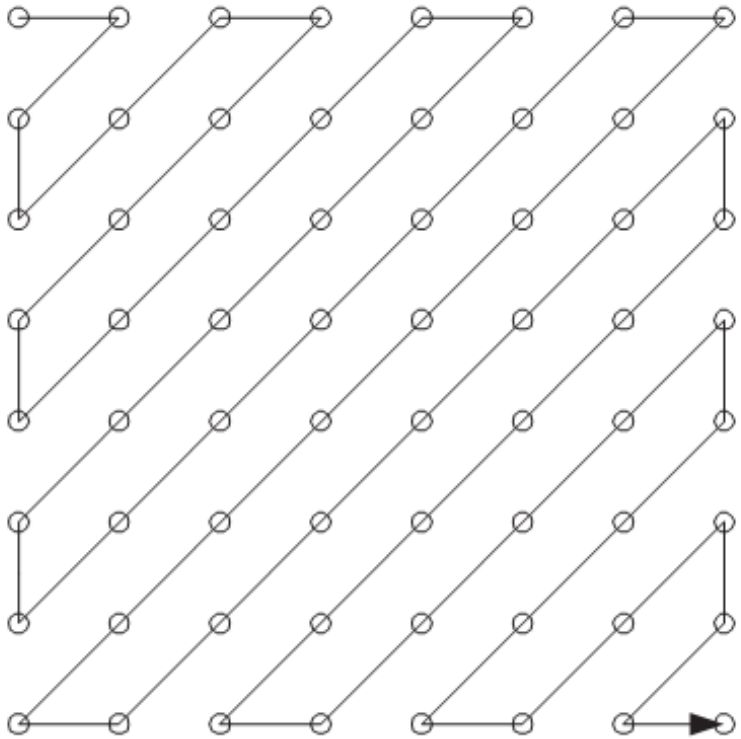
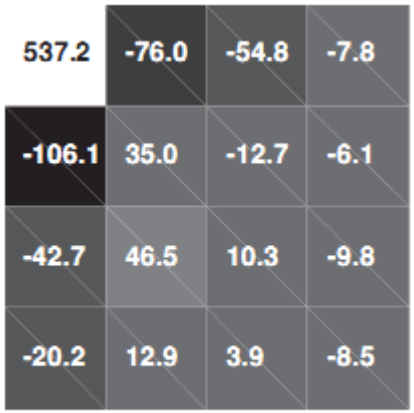
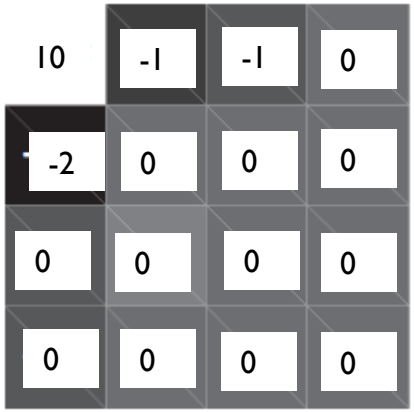


Figure 3.42 Zigzag scan example : frame block

Run-Level



DCT coefficients



DCT coefficients

Example

1. Input array: 16,0,0,−3,5,6,0,0,0,0,−7,. . .
2. Output values: (0,16),(2,−3),(0,5),(0,6),(4,−7). . .
3. Each of these output values (a run-level pair) is encoded as a separate symbol by the entropy encoder.

Entropy Coder

Variable Length Coding (VLC)

Huffman Coding

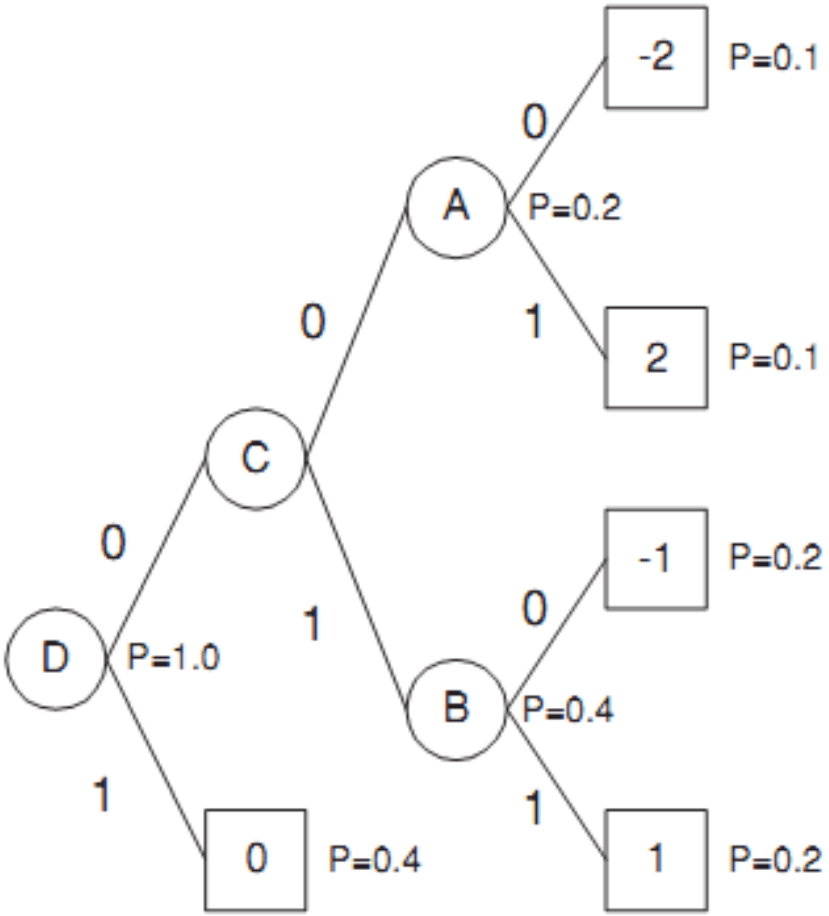
Table 3.2 Probability of occurrence of motion vectors in sequence 1

Vector	Probability p	$\log_2(1/p)$
−2	0.1	3.32
−1	0.2	2.32
0	0.4	1.32
1	0.2	2.32
2	0.1	3.32

1. Generating the Huffman code tree

To generate a Huffman code table for this set of data, the following iterative procedure is carried out:

1. Order the list of data in increasing order of probability.
2. Combine the two lowest-probability data items into a 'node' and assign the joint probability of the data items to this node.
3. Re-order the remaining data items and node(s) in increasing order of probability and repeat step 2.



Original list:	The data items are shown as square boxes. Vectors (−2) and (+2) have the lowest probability and these are the first candidates for merging to form node 'A'.
Stage 1:	The newly-created node 'A', shown as a circle, has a probability of 0.2, from the combined probabilities of (−2) and (2). There are now three items with probability 0.2. Choose vectors (−1) and (1) and merge to form node 'B'.
Stage 2:	A now has the lowest probability (0.2) followed by B and the vector 0; choose A and B as the next candidates for merging to form 'C'.
Stage 3:	Node C and vector (0) are merged to form 'D'.
Final tree:	The data items have all been incorporated into a binary 'tree' containing five data values and four nodes. Each data item is a 'leaf' of the tree.

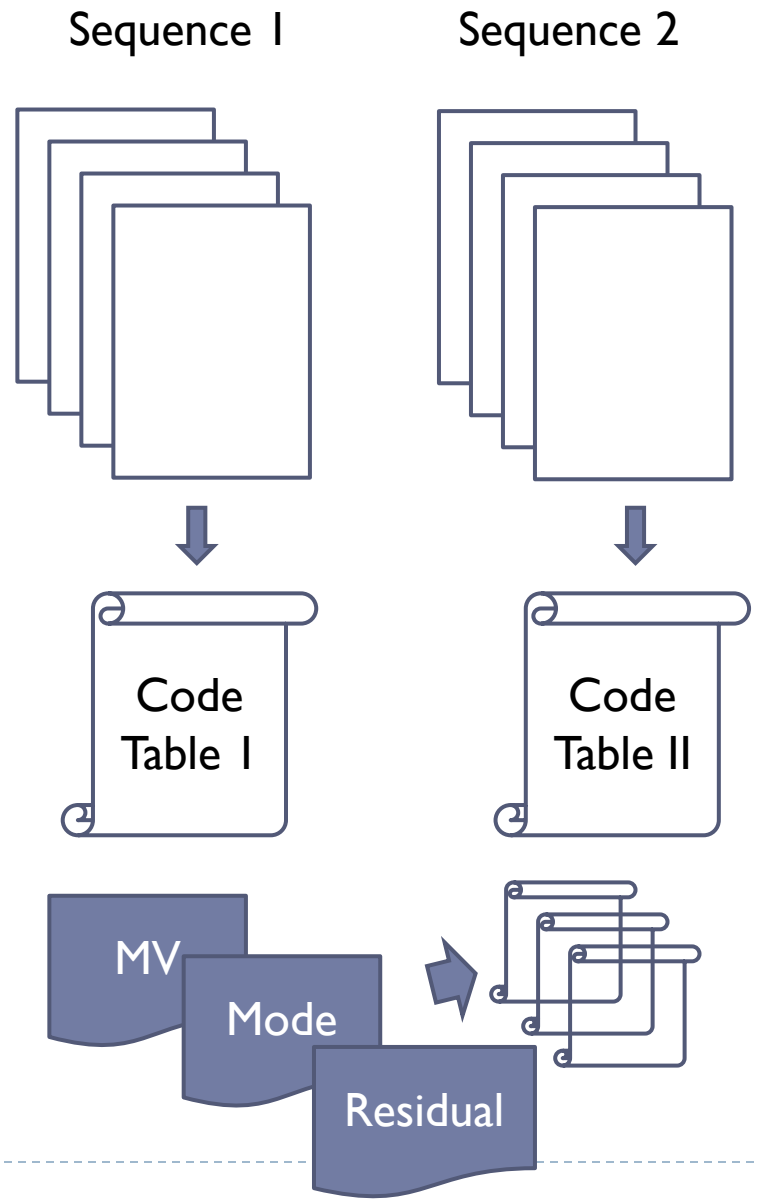
Table 3.3 Huffman codes for sequence 1 motion vectors

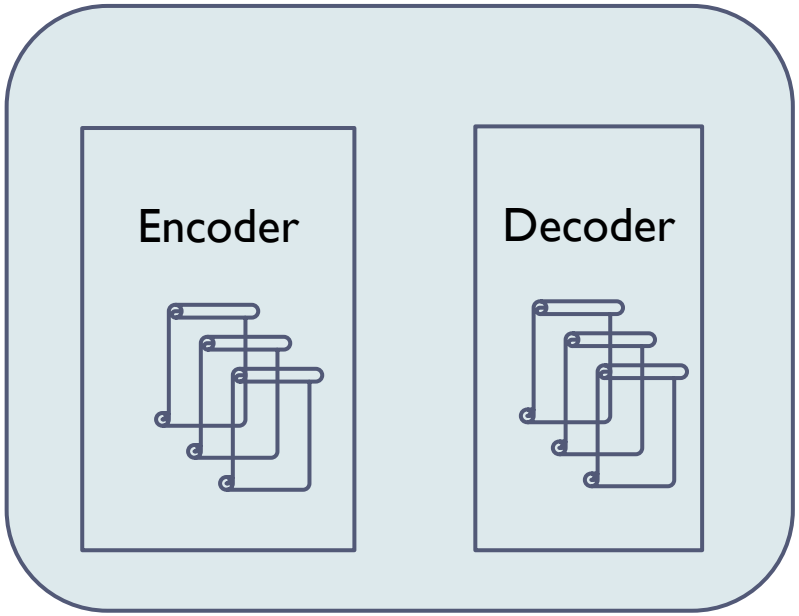
Vector	Code	Bits (actual)	Bits (ideal)
0	1	1	1.32
1	011	3	2.32
−1	010	3	2.32
2	001	3	3.32
−2	000	3	3.32

$(1, 0, -2) \rightarrow 0111000$

Decoding:

1. 011 is decoded as (1)
2. 1 is decoded as (0)
3. 000 is decoded as (−2).


Sequence 1
Sequence 2
Code Table I
Code Table II
MV
Mode
Residual


Encoder
Decoder

Pre-Calculated Huffman Coding

Table 3.6 MPEG-4 Visual Transform Coefficient (TCOEF) VLCs : partial, all codes < 9 bits

Last	Run	Level	Code
0	0	1	10s
0	1	1	110s
0	2	1	1110s
0	0	2	1111s
1	0	1	0111s
0	3	1	01101s
0	4	1	01100s
0	5	1	01011s
0	0	3	010101s
0	1	2	010100s
0	6	1	010011s
0	7	1	010010s
0	8	1	010001s
0	9	1	010000s
1	1	1	001111s
1	2	1	001110s
1	3	1	001101s
1	4	1	001100s
0	0	4	0010111s
0	10	1	0010110s
0	11	1	0010101s
0	12	1	0010100s
1	5	1	0010011s
1	6	1	0010010s
1	7	1	0010001s
1	8	1	0010000s
ESCAPE			0000011s

Table 3.7 MPEG4 Motion Vector Difference (MVD) VLCs

MVD	Code
0	1
+0.5	010
−0.5	011
+1	0010
−1	0011
+1.5	00010
−1.5	00011
+2	0000110
−2	0000111
+2.5	00001010
−2.5	00001011
+3	00001000
−3	00001001
+3.5	00000110
−3.5	00000111
...	...

Arithmetic Coding

Table 3.8 Motion vectors, sequence 1: probabilities and sub-ranges

Vector	Probability	$\log_2(1/P)$	Sub-range
−2	0.1	3.32	0–0.1
−1	0.2	2.32	0.1–0.3
0	0.4	1.32	0.3–0.7
1	0.2	2.32	0.7–0.9
2	0.1	3.32	0.9–1.0

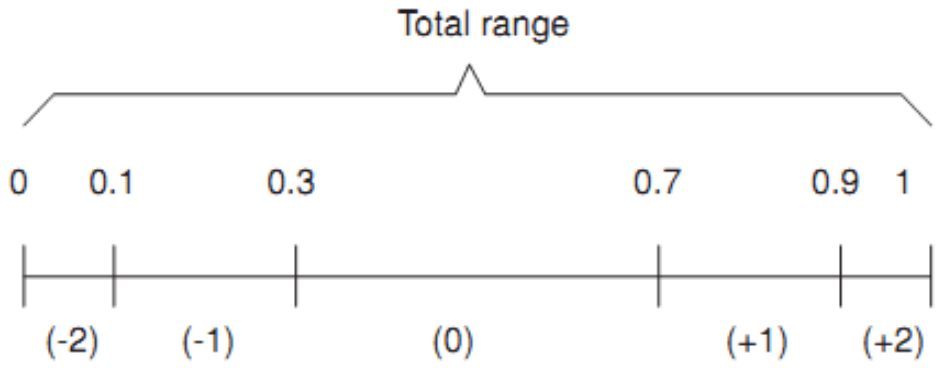
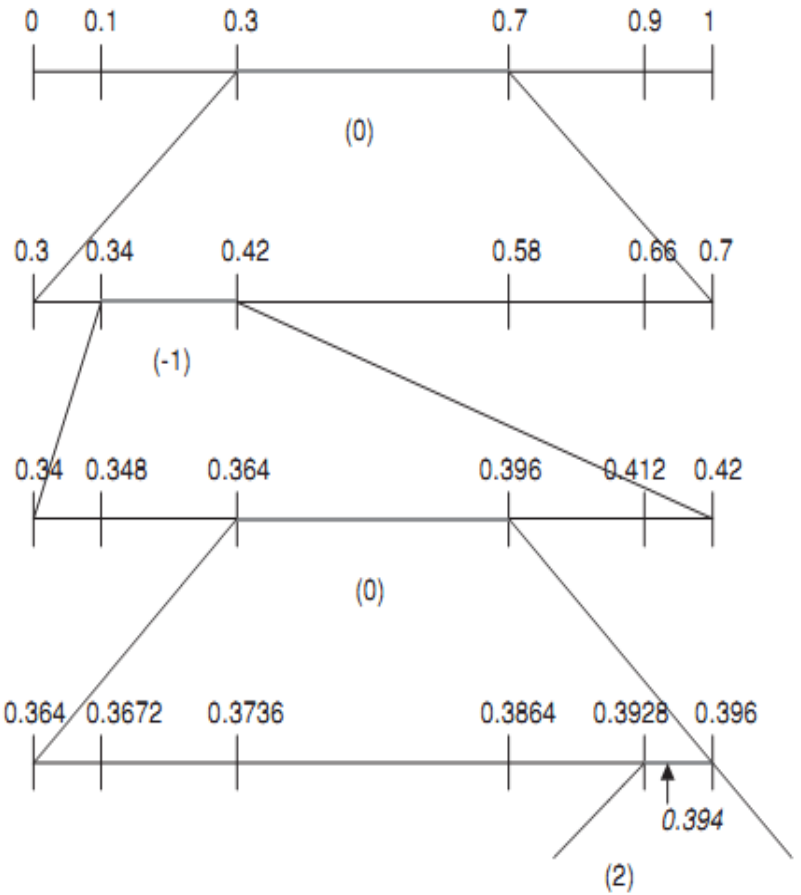


Figure 3.49 Sub-range example

Encoding procedure for vector sequence (0, −1, 0, 2).

Encoding procedure	Range (L → H)	Symbol	Sub-range (L → H)	Notes
1. Set the initial range	0 → 1.0			
2. For the first data symbol, find the corresponding sub-range (Low to High).		(0)	0.3 → 0.7	
3. Set the new range (1) to this sub-range	0.3 → 0.7			
4. For the next data symbol, find the sub-range L to H		(−1)	0.1 → 0.3	This is the sub-range within the interval 0 - 1
5. Set the new range (2) to this sub-range within the previous range	0.34 → 0.42			0.34 is 10% of the range; 0.42 is 30% of the range
6. Find the next sub-range		(0)	0.3 → 0.7	
7. Set the new range (3) within the previous range	0.364 → 0.396			0.364 is 30% of the range; 0.396 is 70% of the range
8. Find the next sub-range		(2)	0.9 → 1.0	
9. Set the new range (4) within the previous range	0.3928 → 0.396			0.3928 is 90% of the range; 0.396 is 100% of the range



0.394 (9b)

$$\log_2(1/P_0) + \log_2(1/P_{-1}) + \log_2(1/P_0) + \log_2(1/P_2) \text{ bits} = 8.28 \text{ bits}$$

Decoding procedure

Decoding procedure	Range	Sub-range	Decoded symbol
1. Set the initial range	$0 \rightarrow 1$		
2. Find the sub-range in which the received number falls. This indicates the first data symbol.		$0.3 \rightarrow 0.7$	(0)
3. Set the new range (1) to this sub-range	$0.3 \rightarrow 0.7$		
4. Find the sub-range **of the new range** in which the received number falls. This indicates the second data symbol.		$0.34 \rightarrow 0.42$	(−1)
5. Set the new range (2) to this sub-range within the previous range	$0.34 \rightarrow 0.42$		
6. Find the sub-range in which the received number falls and decode the third data symbol.		$0.364 \rightarrow 0.396$	(0)
7. Set the new range (3) to this sub-range within the previous range	$0.364 \rightarrow 0.396$		
8. Find the sub-range in which the received number falls and decode the fourth data symbol.		$0.3928 \rightarrow 0.396$	

Hybrid DPCM/DCT Video CODEC Model

Differential Pulse Code Modulation (DPCM)

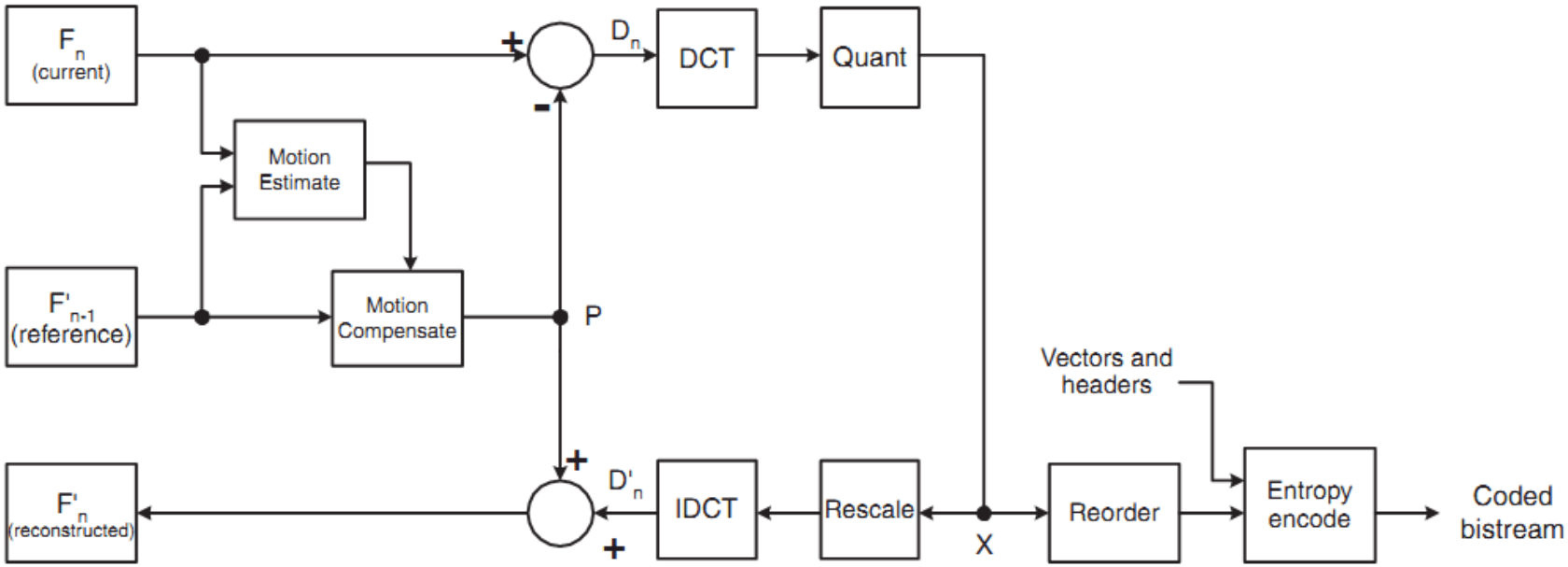


Figure 3.51 DPCM/DCT video encoder

5. D is transformed using the DCT. Typically, D is split into 8 × 8 or 4 × 4 sub-blocks and each sub-block is transformed separately.
6. Each sub-block is quantized (X).
7. The DCT coefficients of each sub-block are reordered and run-level coded.
8. Finally, the coefficients, motion vector and associated header information for each macroblock are entropy encoded to produce the compressed bitstream.

The reconstruction data flow is as follows:

1. Each quantized macroblock X is rescaled and inverse transformed to produce a decoded residual D'. Note that the non-reversible quantization process means that D' is not identical to D, i.e. distortion has been introduced.
2. The motion compensated prediction P is added to the residual D' to produce a reconstructed macroblock. The reconstructed macroblocks are combined to produce reconstructed frame F'_n.

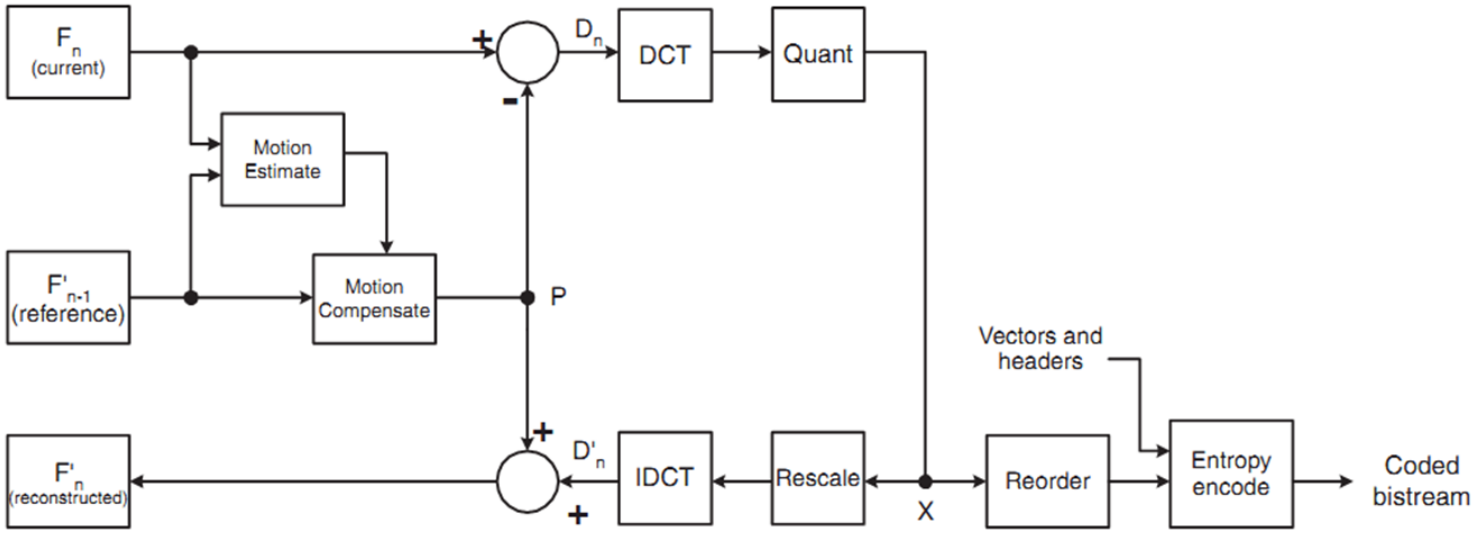


Figure 3.51 DPCM/DCT video encoder

Decoder data flow

1. A compressed bitstream is entropy decoded to extract coefficients, motion vector and header for each macroblock.
2. Run-level coding and reordering are reversed to produce a quantized, transformed macroblock X.
3. X is rescaled and inverse transformed to produce a decoded residual D'.
4. The decoded motion vector is used to locate a 16 × 16 region in the decoder's copy of the previous (reference) frame F'_{n-1}. This region becomes the motion compensated prediction P.
5. P is added to D' to produce a reconstructed macroblock. The reconstructed macroblocks are saved to produce decoded frame F'_n.

After a complete frame is decoded, F'_n is ready to be displayed and may also be stored as a reference frame for the next decoded frame F'_{n+1}.

It is clear from the figures and from the above explanation that the encoder includes a decoding path : rescale, IDCT, reconstruct. This is necessary to ensure that the encoder and decoder use identical reference frames F'_{n-1} for motion compensated prediction.

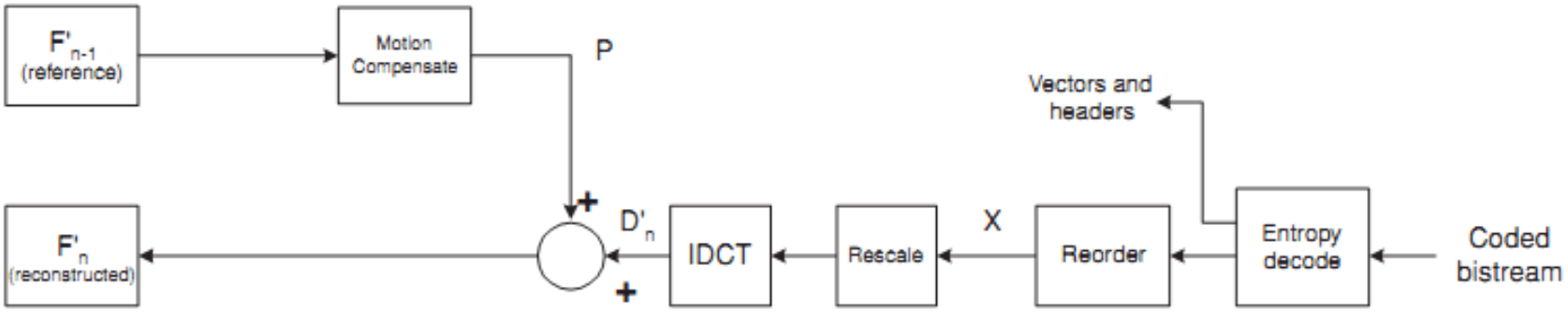
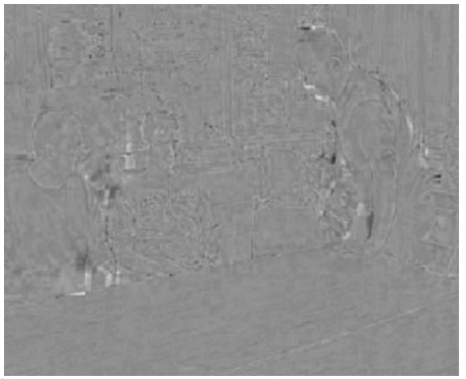


Figure 3.52 DPCM/DCT video decoder



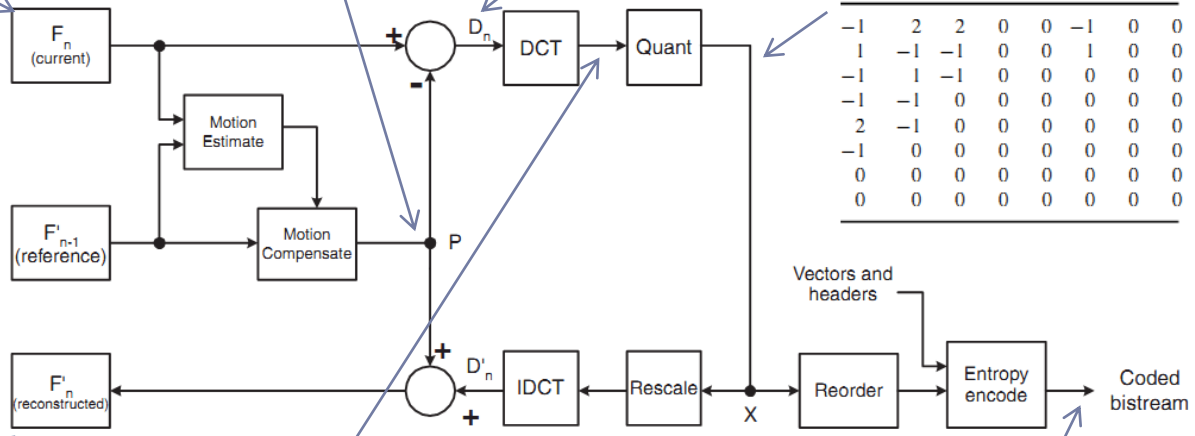


Figure 3.51 DPCM/DCT video encoder

Table 3.11 Quantized coefficients

−1	2	2	0	0	−1	0	0
1	−1	−1	0	0	1	0	0
−1	1	−1	0	0	0	0	0
−1	−1	0	0	0	0	0	0
2	−1	0	0	0	0	0	0
−1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

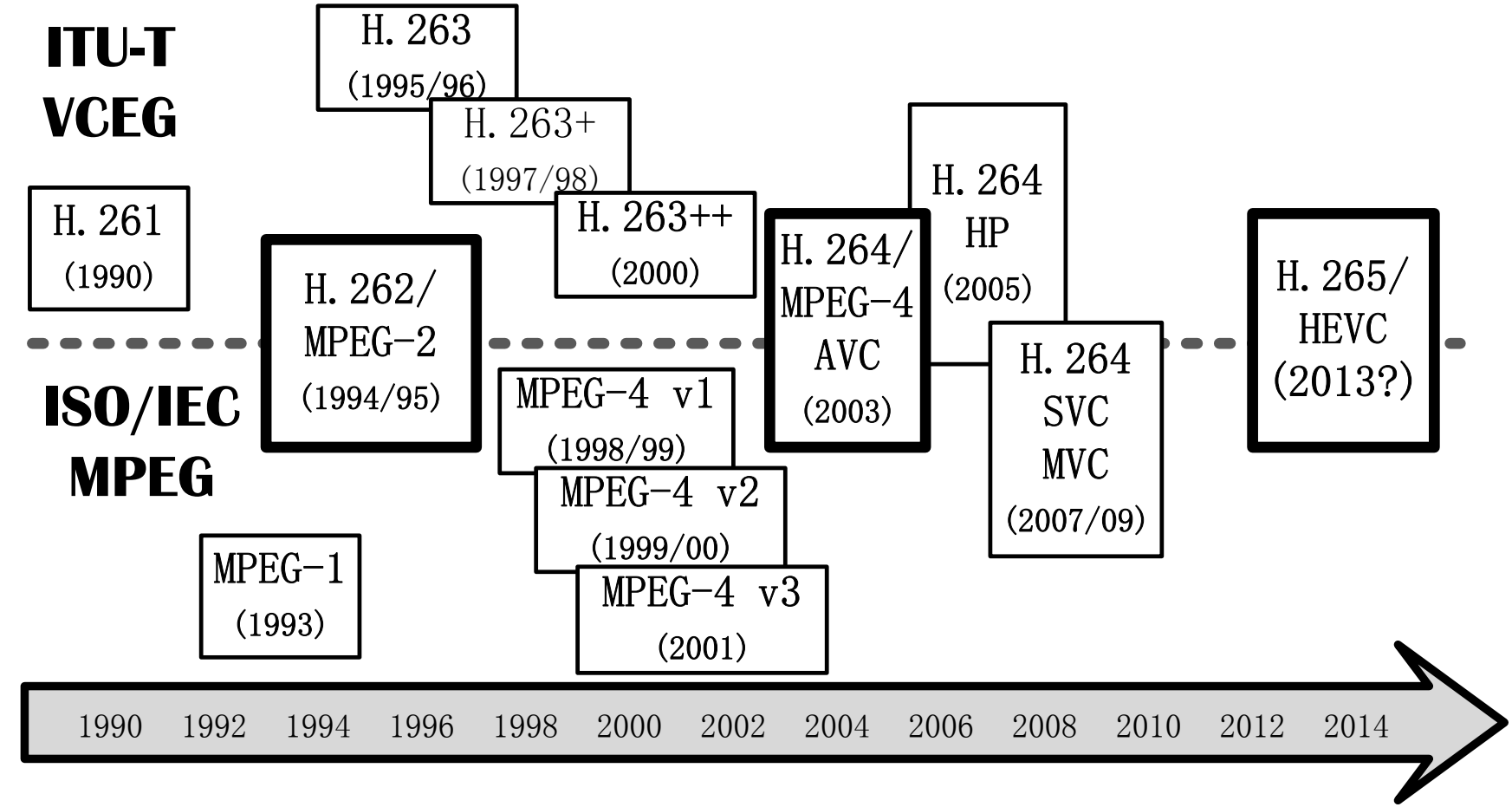
Table 3.10 DCT coefficients

−13.50	20.47	20.20	2.14	−0.50	−10.48	−3.50	−0.62
10.93	−11.58	−10.29	−5.17	−2.96	10.44	4.96	−1.26
−8.75	9.22	−17.19	2.26	3.83	−2.45	1.77	1.89
−7.10	−17.54	1.24	−0.91	0.47	−0.37	−3.55	0.88
19.00	−7.20	4.08	5.31	0.50	0.18	−0.61	0.40
−13.06	3.12	−2.04	−0.17	−1.19	1.57	−0.08	−0.51
1.73	−0.69	1.77	0.78	−1.86	1.47	1.19	0.42
−1.99	−0.05	1.24	−0.48	−1.86	−1.17	−0.21	0.92

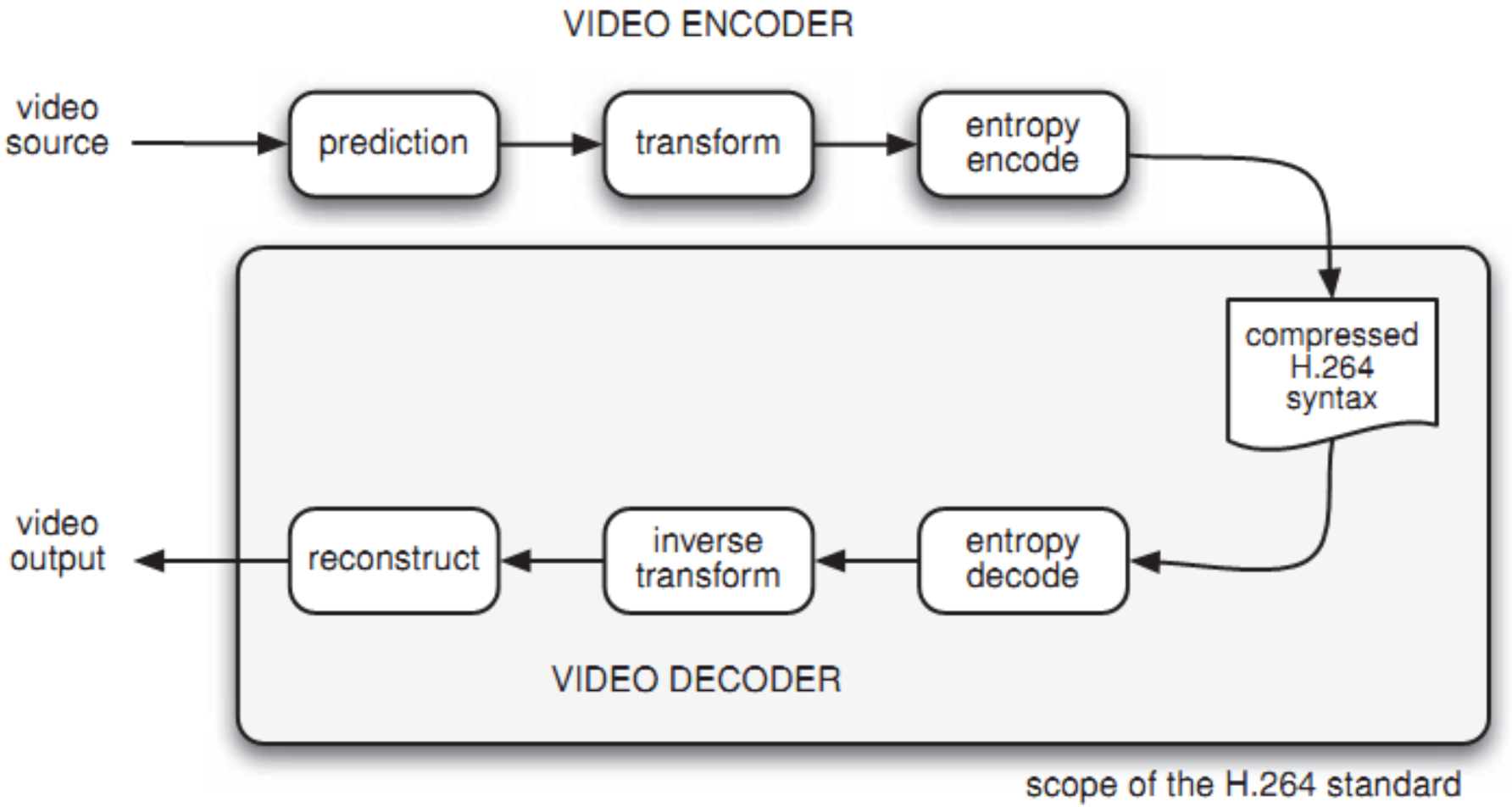
Table 3.12 Variable length coding example

Run, Level, Last	VLC including sign bit
(0,−1, 0)	101
(0,2, 0)	11100
(0,1, 0)	100
(0,−1, 0)	101
(0,−1, 0)	101
(0,2, 0)	11100
(1,−1, 0)	1101
...	...
(5,1, 1)	00100110

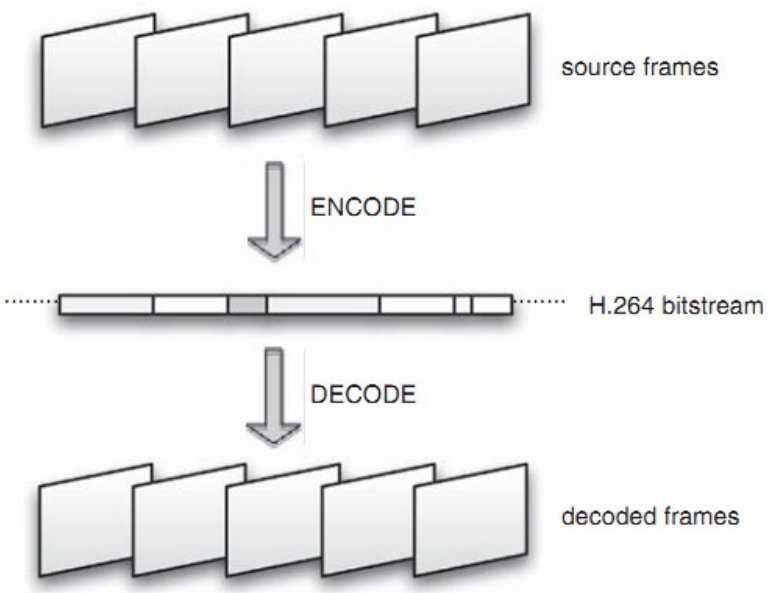
H.264/AVC Overview

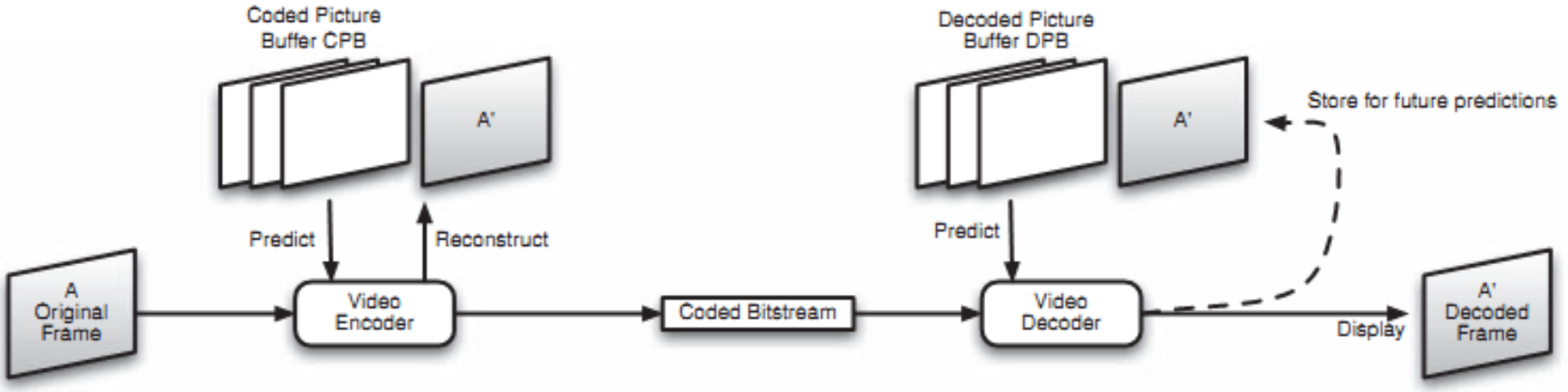


电视信号	模拟电视	数字电视	
电视分辨率	PAL、NTSC 720*576	SD、HD 1280x720 1920x1080	Super Hi-vision 4Kx2K 8Kx4K


VIDEO ENCODER
video source
prediction
transform
entropy encode
compressed H.264 syntax
video output
reconstruct
inverse transform
entropy decode
VIDEO DECODER
scope of the H.264 standard

Working Flow





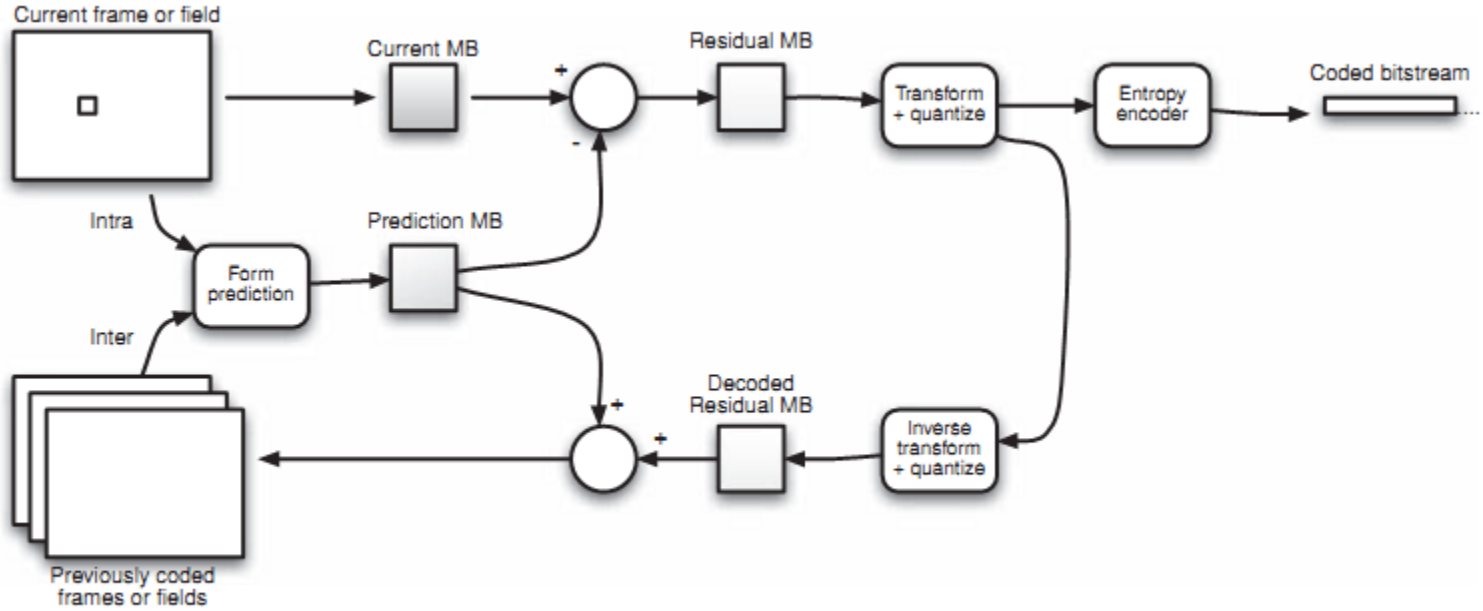


Figure 4.4 Typical H.264 encoder

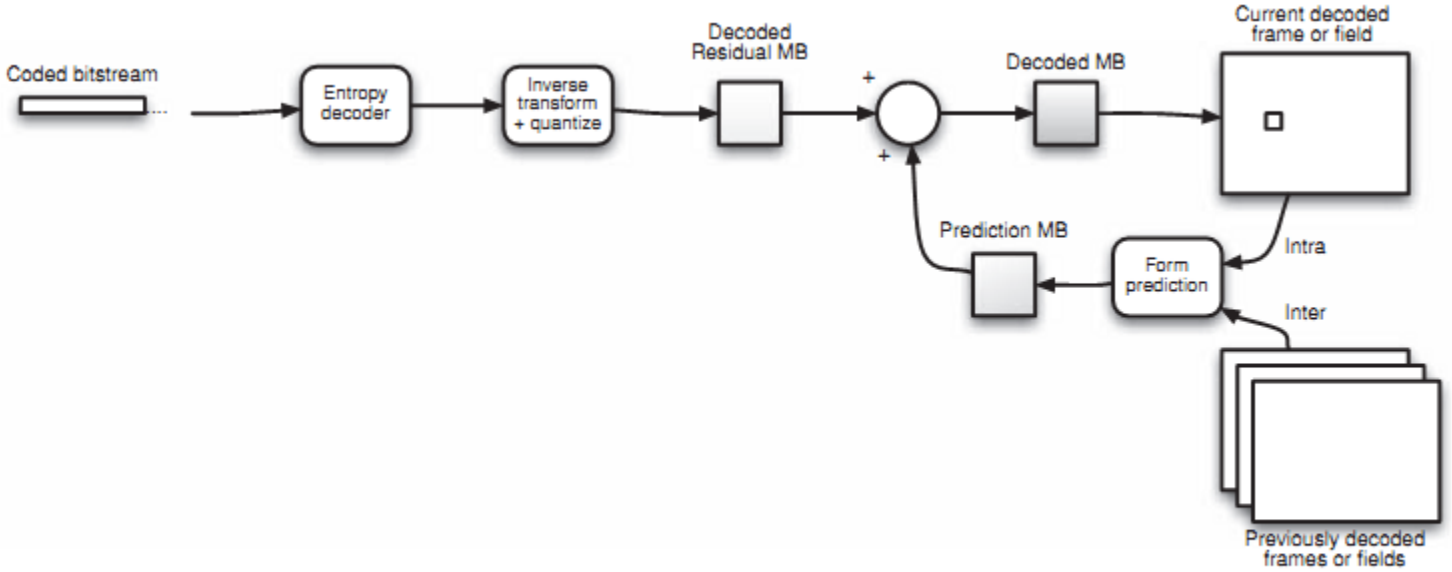
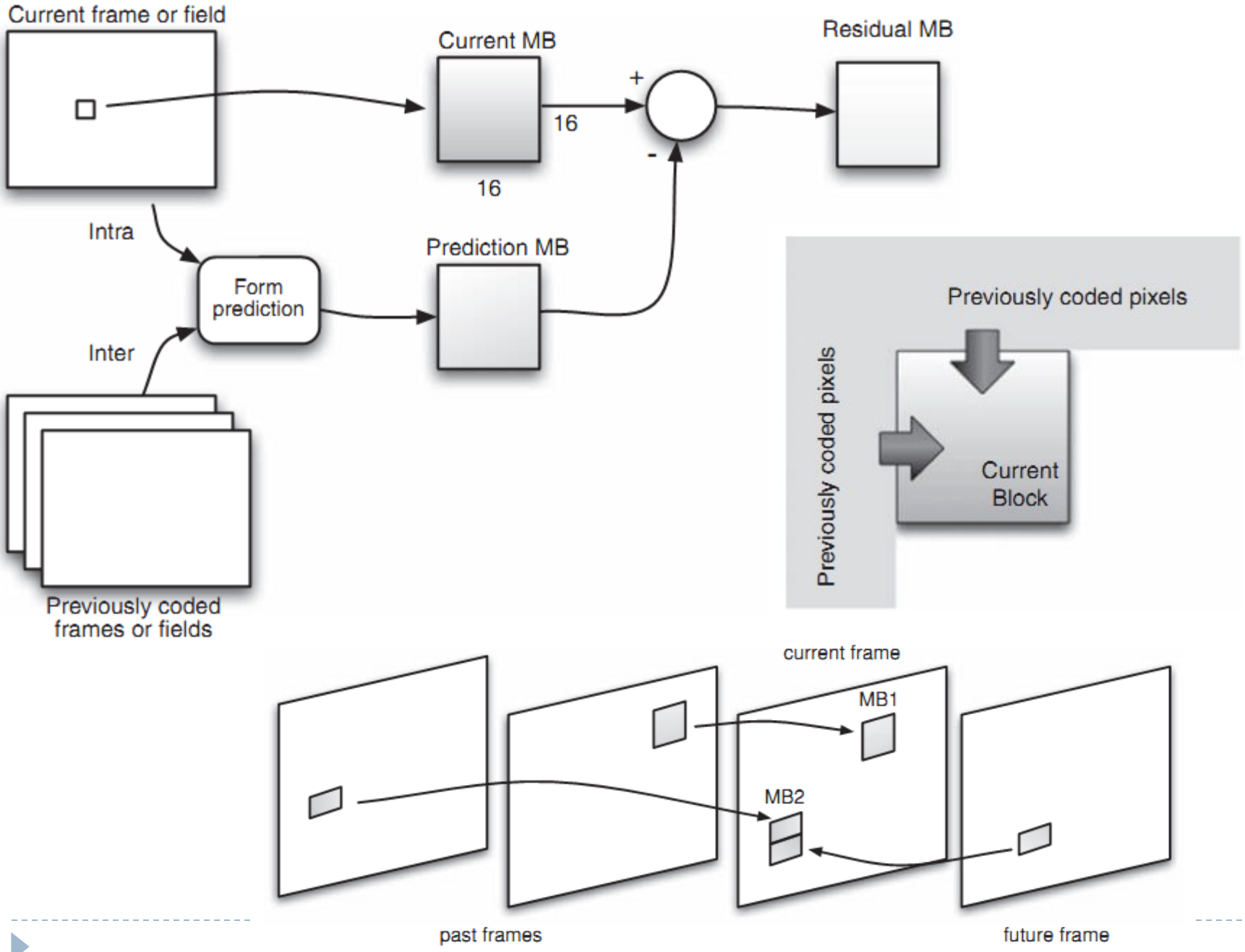


Figure 4.5 Typical H.264 decoder


Current frame or field
Current MB
16
16
+
-
Residual MB
Intra
Inter
Form prediction
Prediction MB
Previously coded frames or fields
Previously coded pixels
Previously coded pixels
Current Block
current frame
MB1
MB2
past frames
future frame

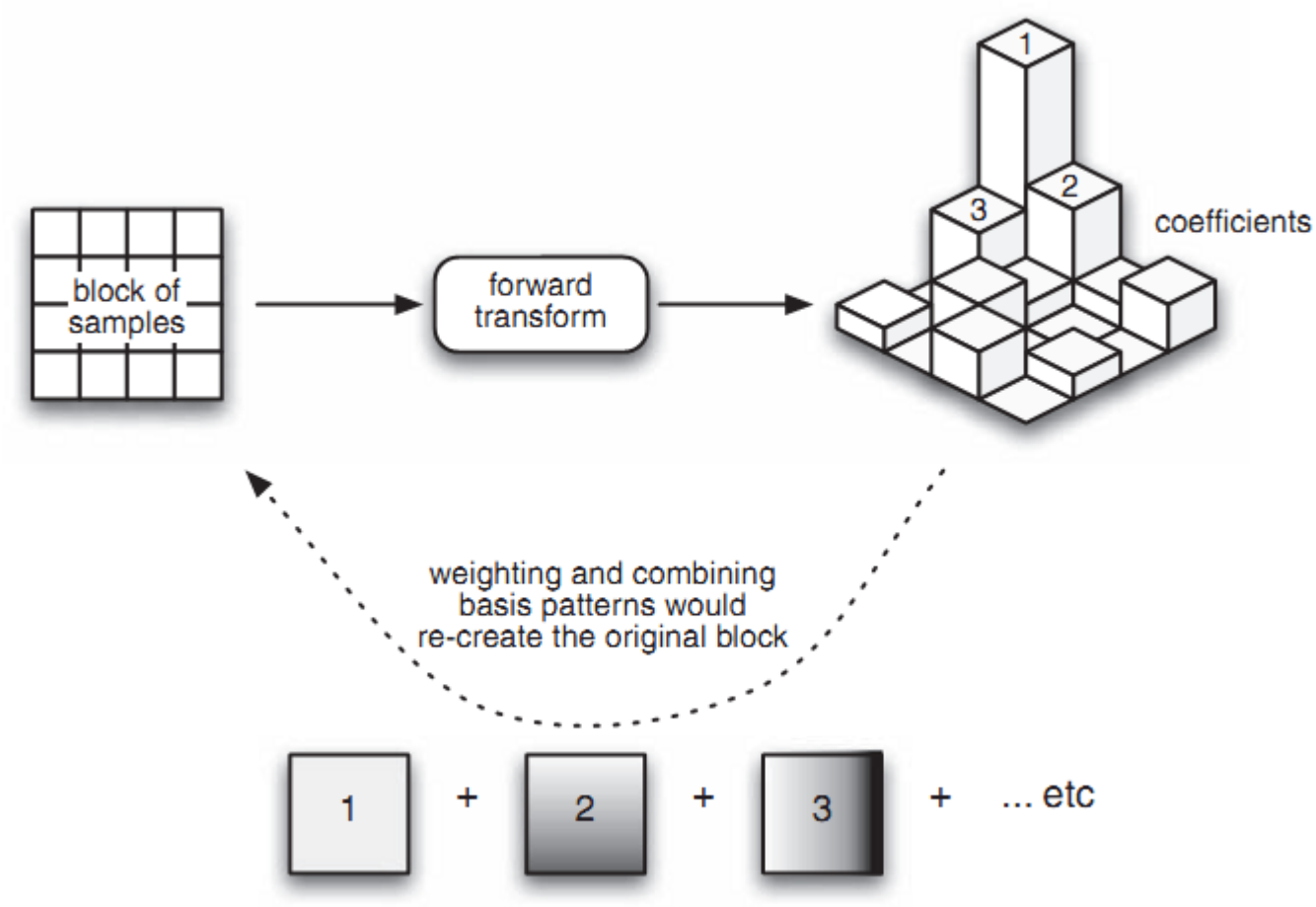


Figure 4.11 Forward transform

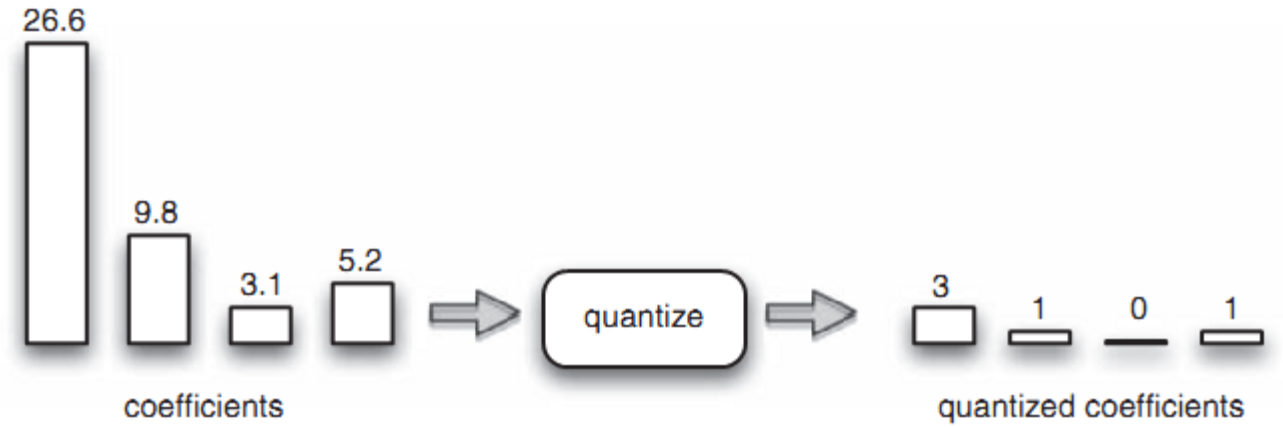
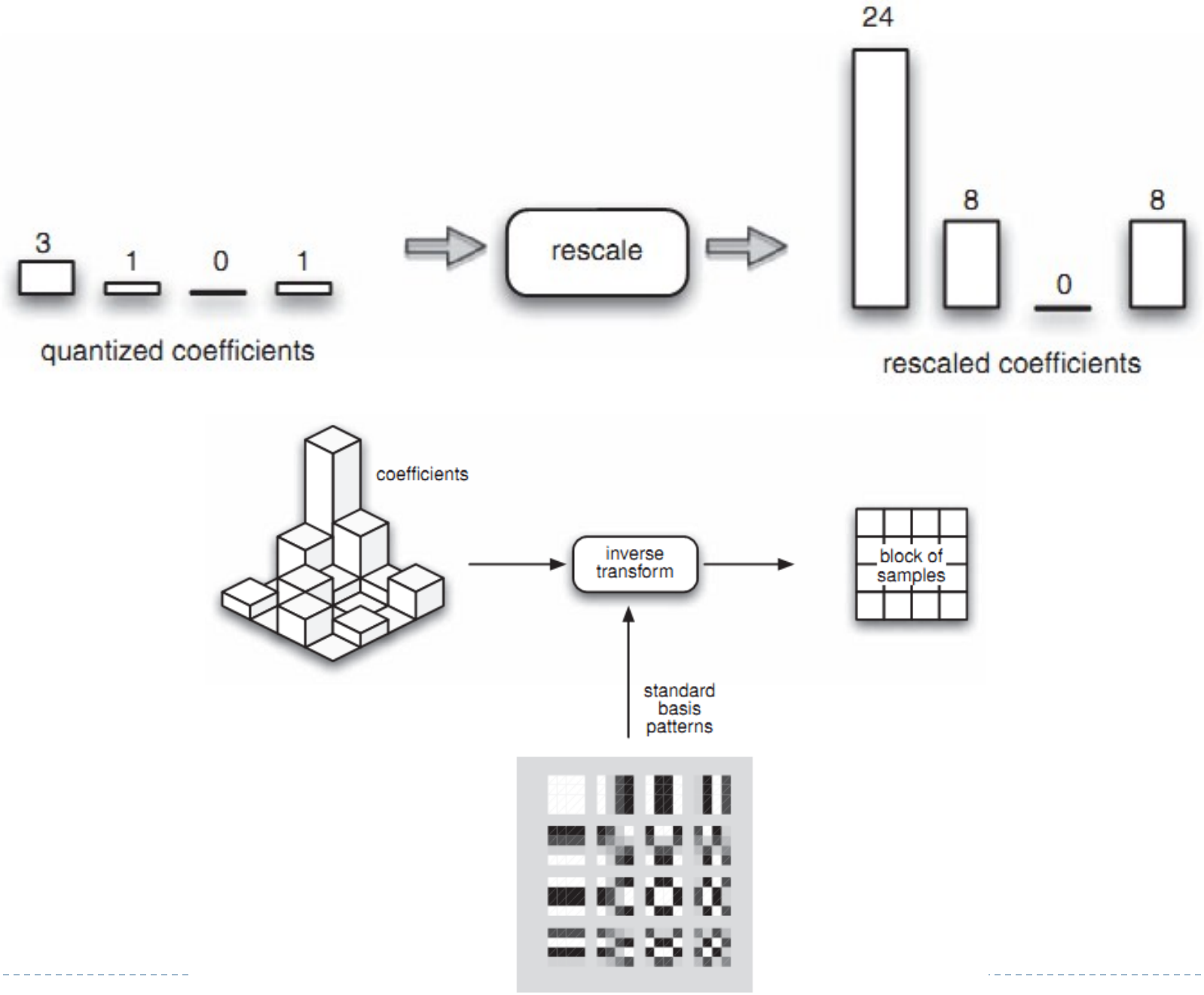


Figure 4.12 Quantization example

73 87 64 13 40 63 23 2 36 24 68 26 29 98 67 12 **Original block (4x4)**	181.3 47.4 –65.8 4.0 5.7 29.6 16.4 –2.2 40.3 18.3 –28.8 –13.8 13.1 –20.2 13.8 33.9 **Forward transform coefficients**
18 5 –7 0 1 3 2 0 4 2 –3 –1 1 –2 1 3 **Quantized coefficients (step size = 10)**	9 2 –3 0 0 1 1 0 2 1 –1 –1 1 –1 1 2 **Quantized coefficients (step size = 20)**

Figure 4.13 Example: Block, transform coefficients, quantized coefficients


3
1
0
1
quantized coefficients
rescale
24
8
0
8
rescaled coefficients
coefficients
inverse transform
block of samples
standard basis patterns

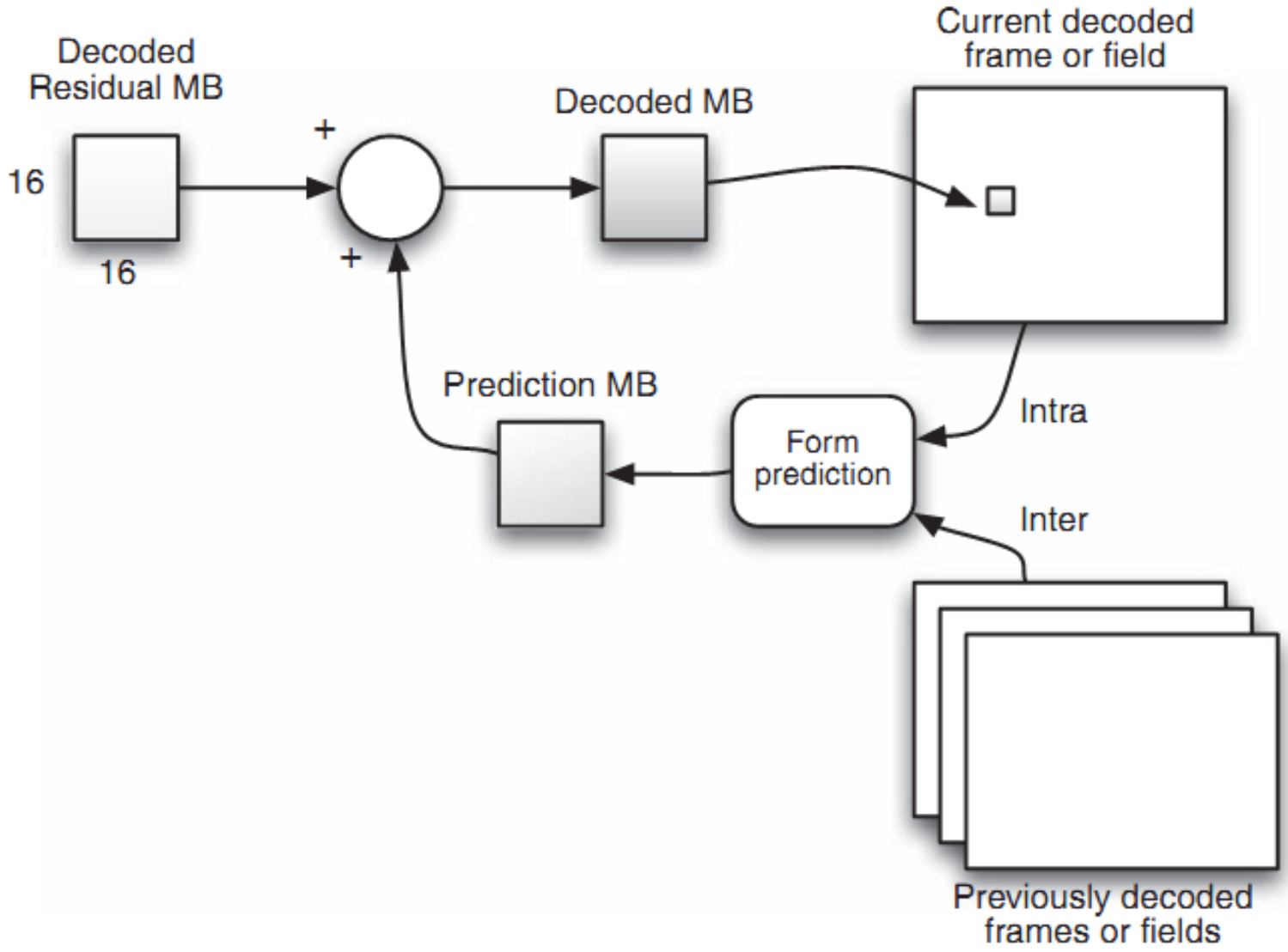
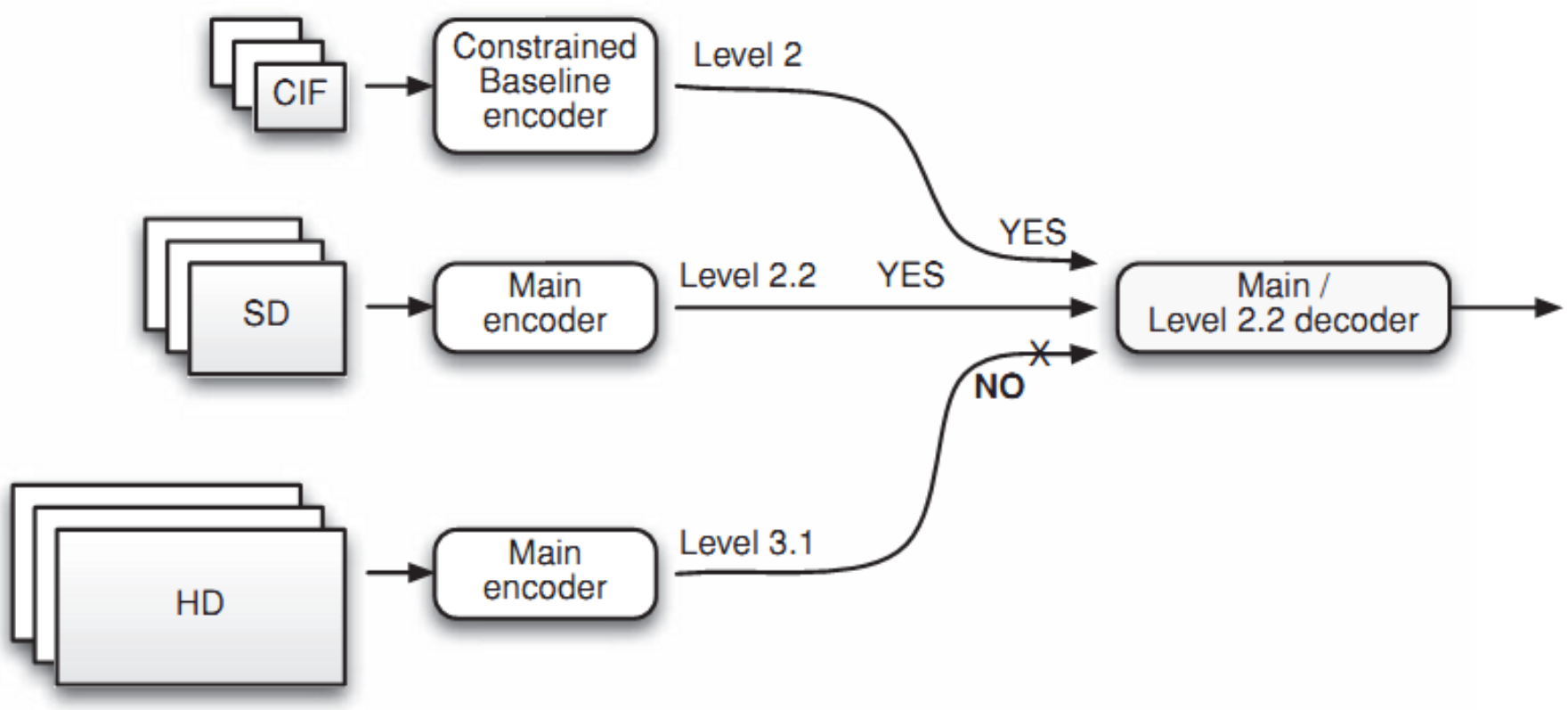


Figure 4.17 Reconstruction flow diagram

Profiles and Levels



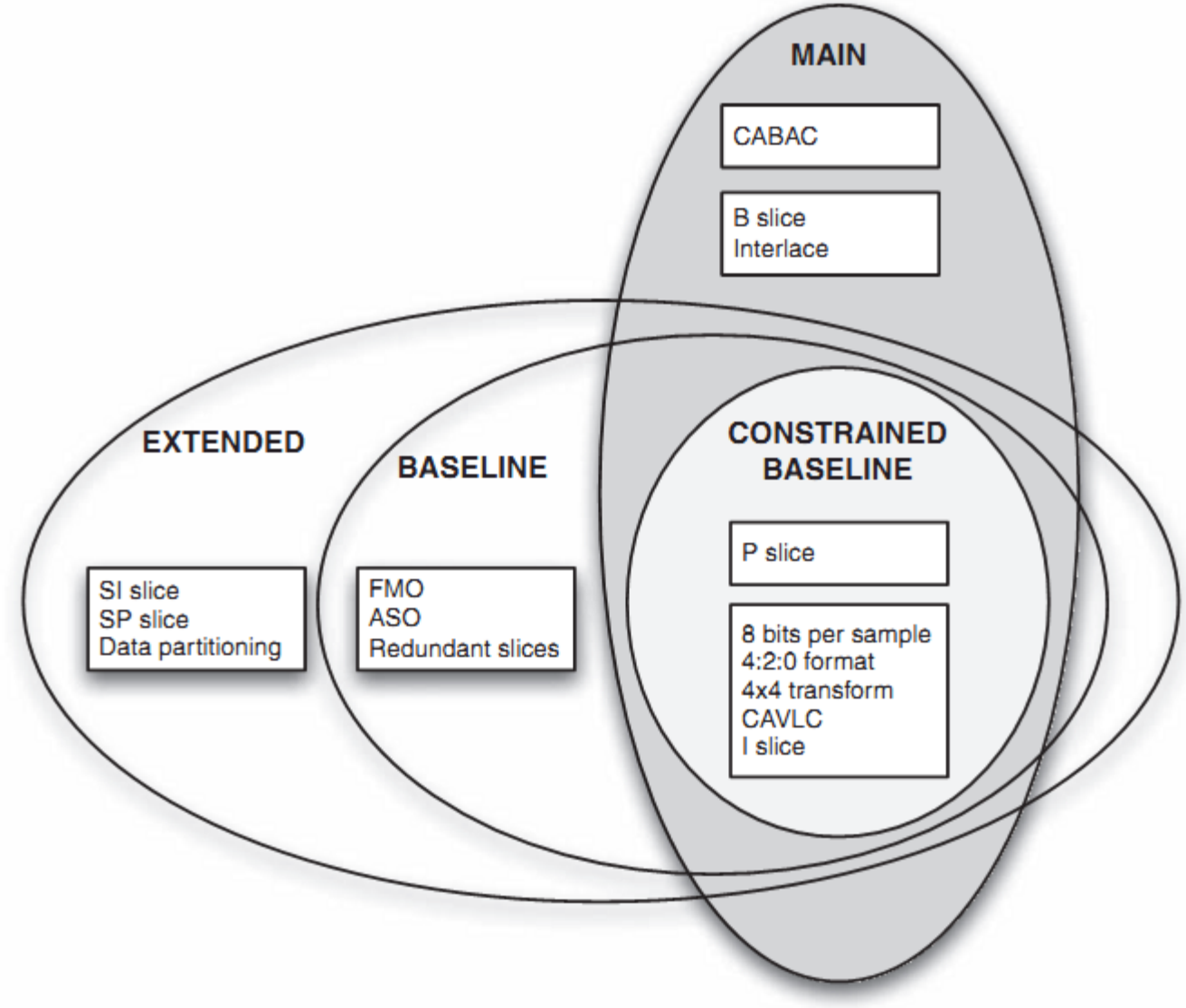


Figure 8.1 Baseline, Constrained Baseline, Extended and Main Profiles

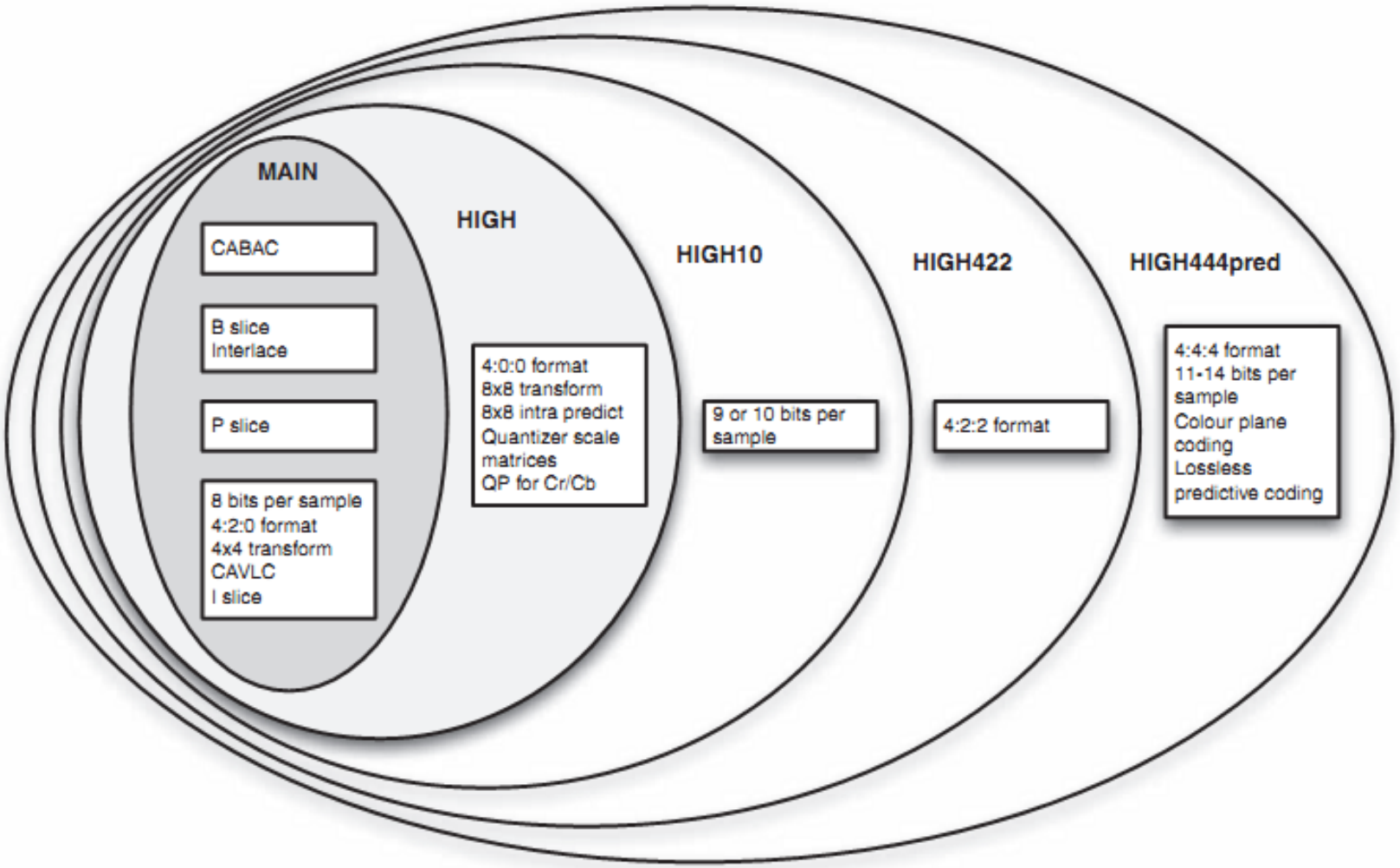


Figure 8.2 Main and High Profiles

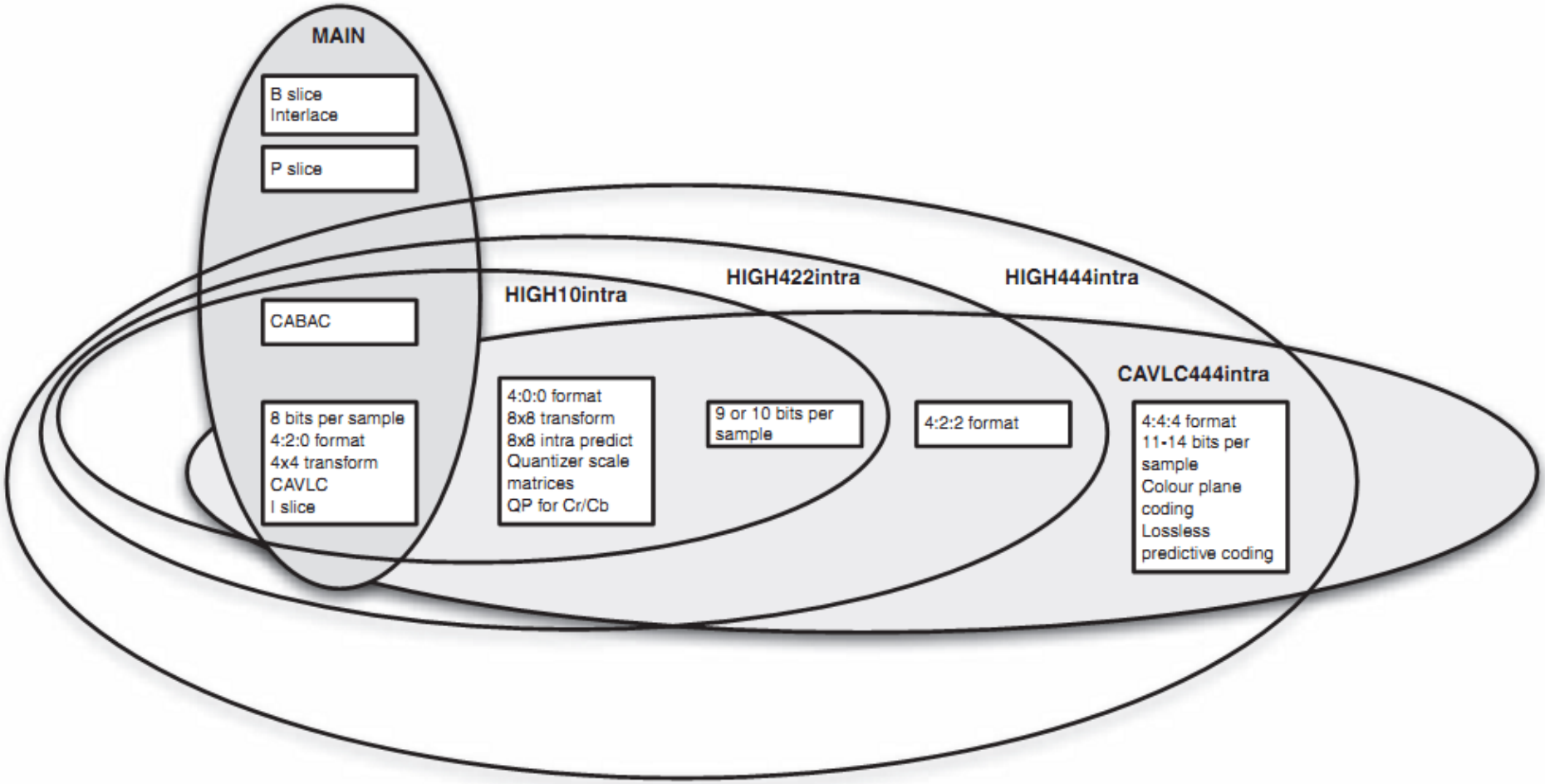
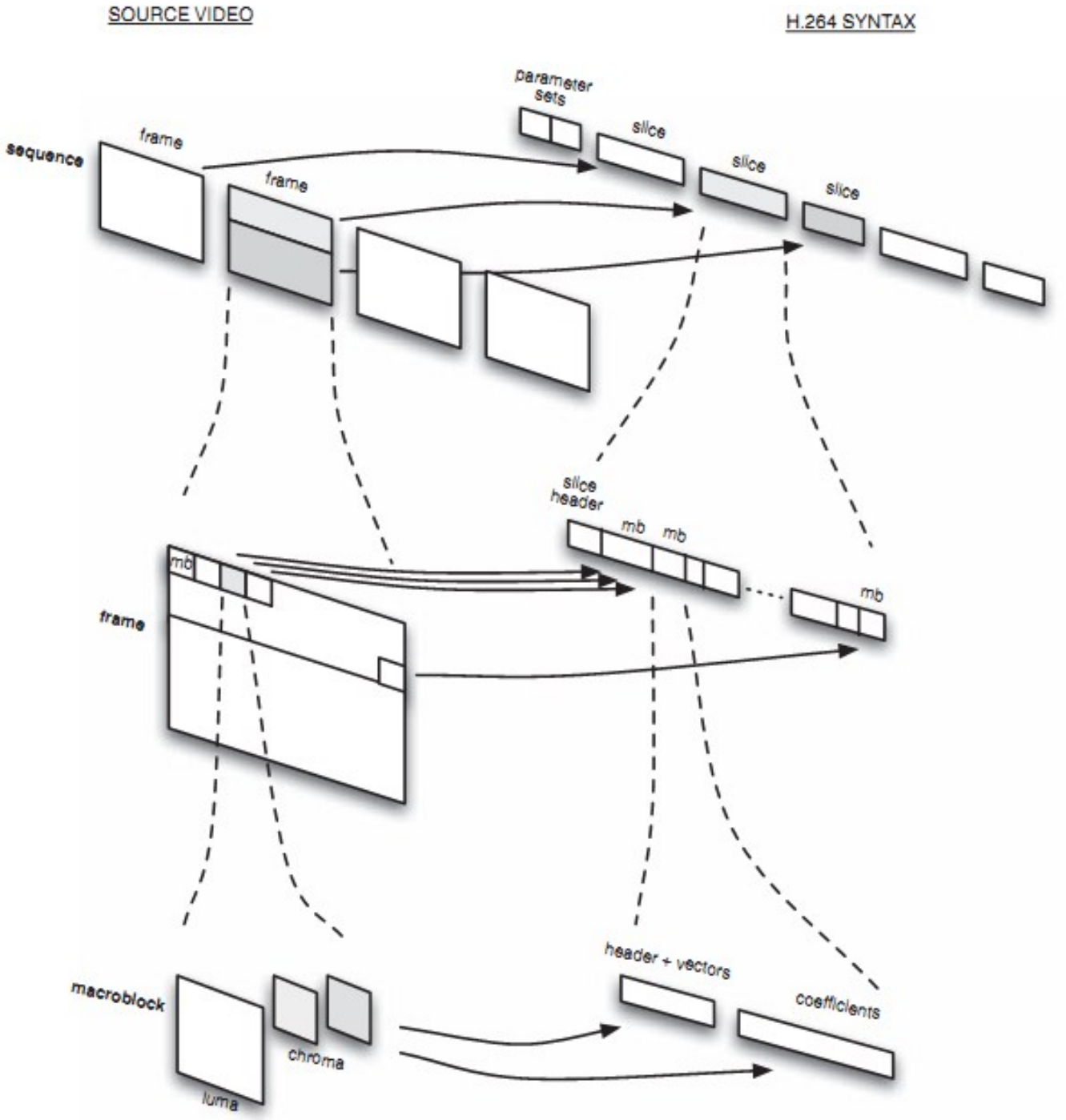


Figure 8.3 Main and Intra Profiles

Syntax

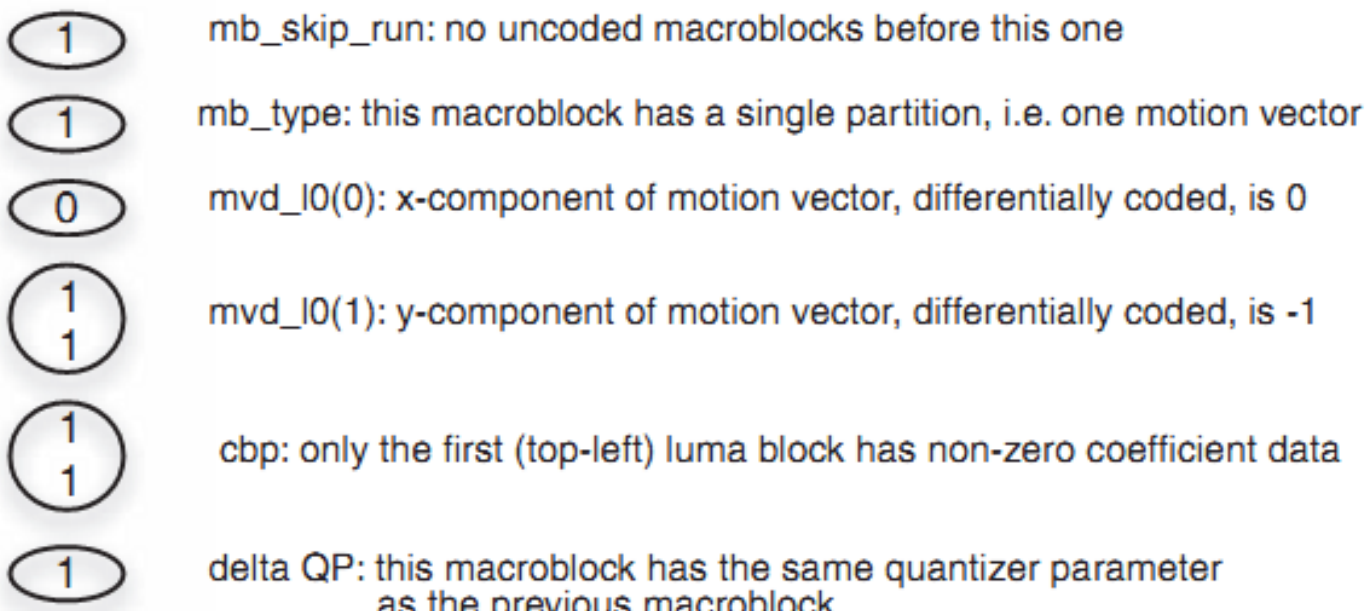

SOURCE VIDEO
H.264 SYNTAX
parameter sets
slice
slice
slice
sequence
frame
frame
slice header
mb
mb
mb
frame
mb
header + vectors
coefficients
macroblock
chroma
luma

Example

An H.264 decoder receives the following series of bits, describing an inter-coded macroblock or P MB coded using context-adaptive VLCs:

1101111100011100011100 10 (alternate syntax elements highlighted).

Figure 4.20 shows the syntax elements in this sequence of bits. The header elements, the first eight bits and six syntax elements, indicate the macroblock type (P, one motion vector),



Performance

Figure 4.23 A video frame compressed at the same bitrate using MPEG-2 (left), MPEG-4 Visual (centre) and H.264 compression (right)

盗版组织抛弃Xvid/avi格式惹恼用户

blackhat 发表于 2012年3月05日 11时42分 星期一

来自中国还流行RMVB吗部门

知名的TV盗版组织联合作出了一项决定：他们不再发布Xvid/avi 格式的视频，改用MP4/x264格式。他们做出的这一决定被部分BitTorrent用户视为暗箱操作缺乏民主，一些人甚至威胁要抵制他们发布的视频。

盗版组织在视频发布标准《The SD x264 TV Releasing Standards 2012》中指出，x264已在过去几年成为最先进的编解码器，与Xvid相比，在标清（SD）分辨率上它能提供更高的视频质量和更高的压缩率。自2月22日起，视频发布组织如ASAP、BAJSKORV、C4TV、D2V、DiVERGE、FTP、KYR、LMAO、LOL、 MOMENTUM、SYS、TLA和YesTV开始采用新格式发布视频。一些BitTorrent用户抱怨他们的DVD播放器不支持MP4格式，还有一些人则对发布组织爆粗口，另一些人则把发布的MP4格式转制成AVI格式重新上传到BT网站。

Frame, Field & Picture

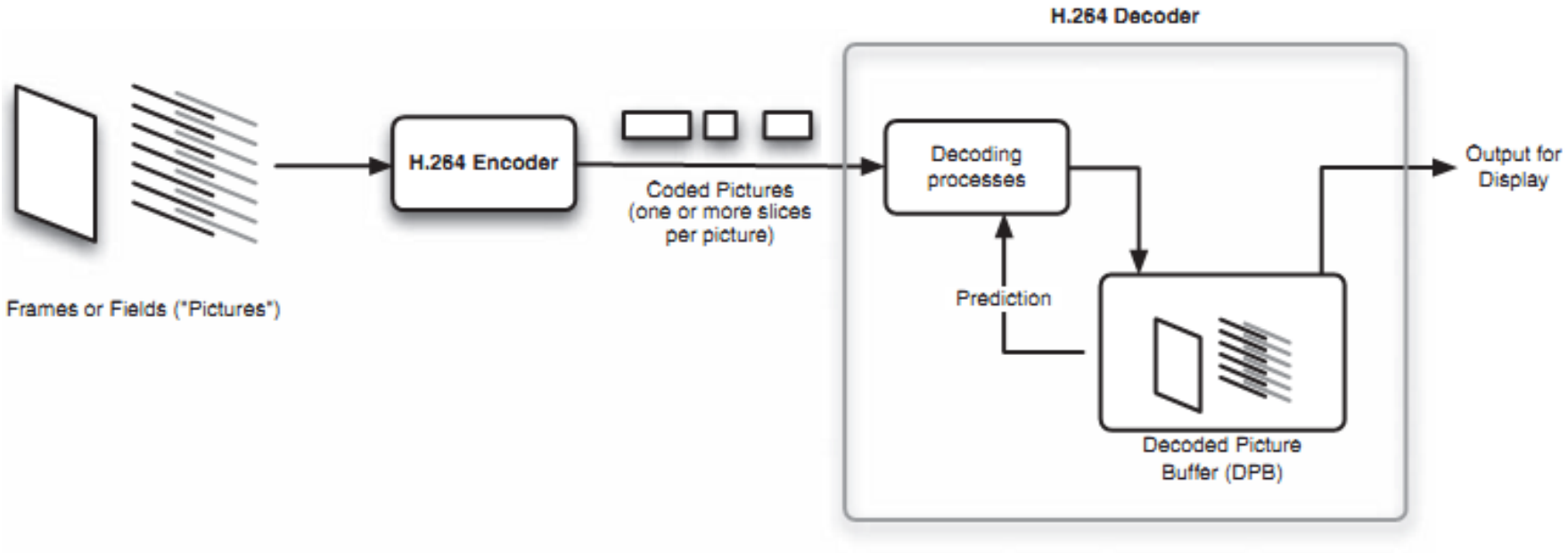


Figure 5.2 Picture handling in H.264, overview

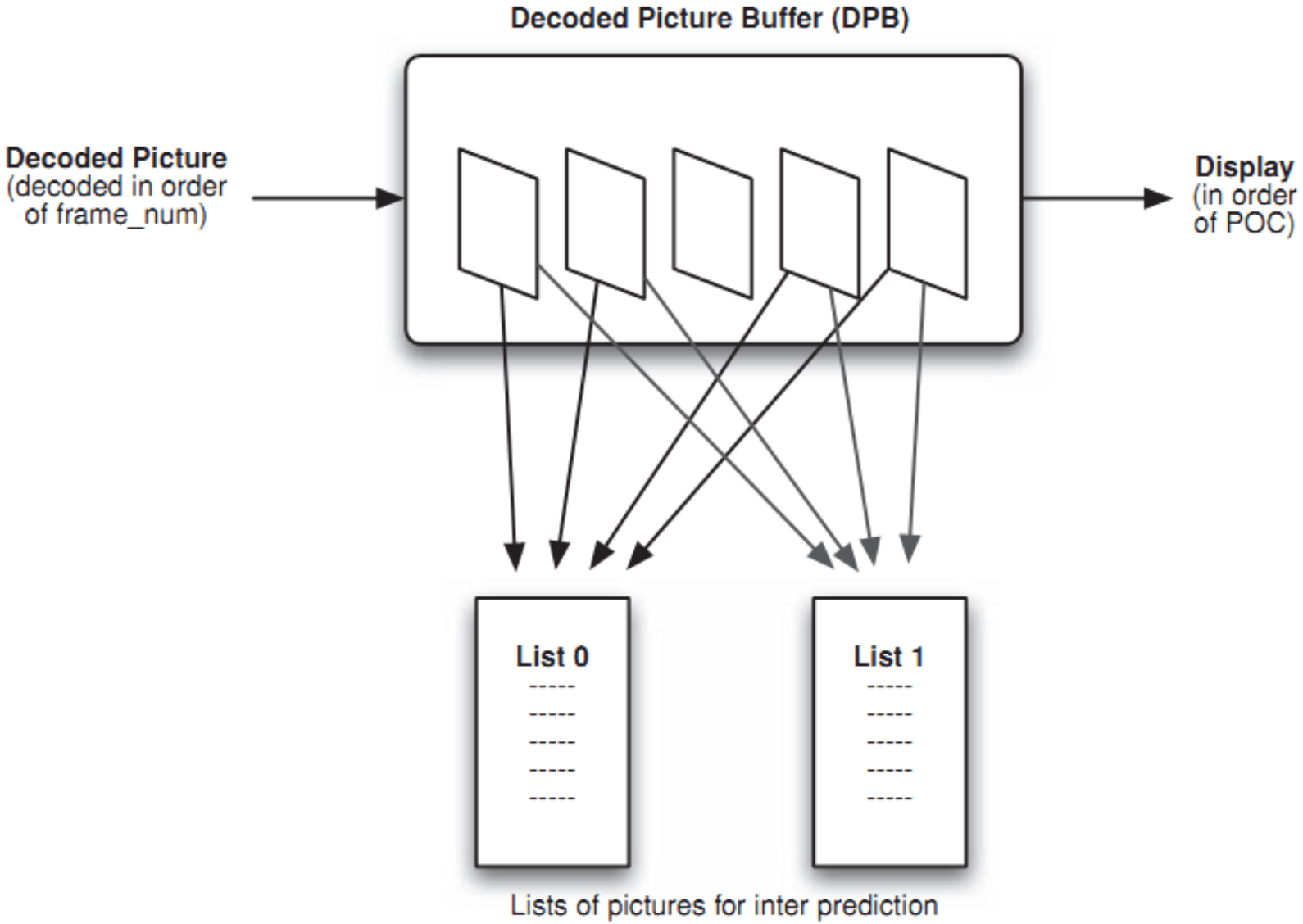


Figure 5.3 Decoded Picture Buffer and picture orders

Decoding Order

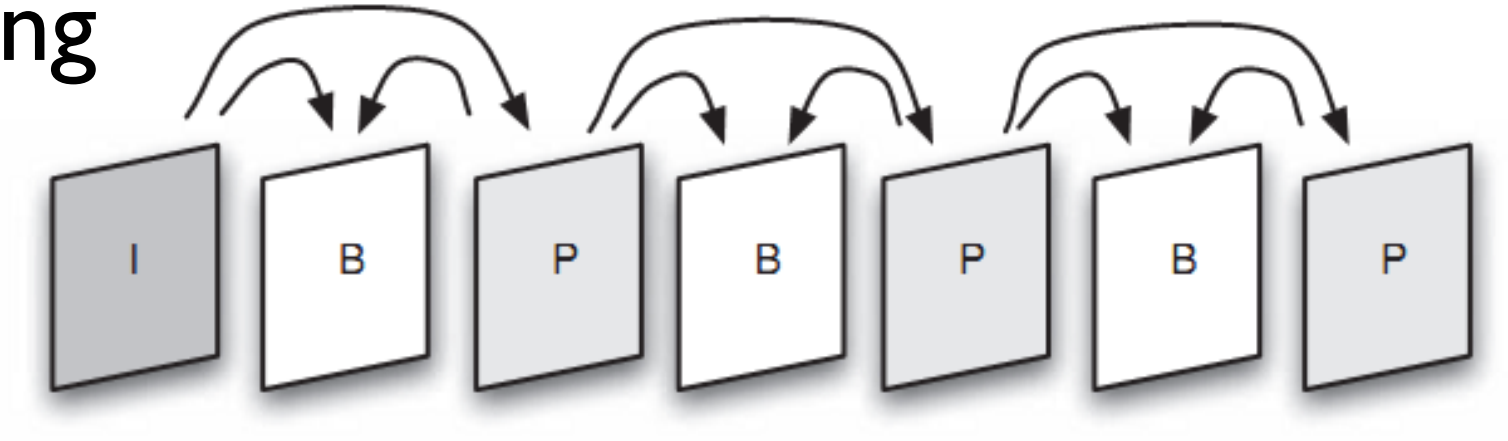


Figure 5.4 Display order: Type 0 example

- Type 0: The least significant bits of POC are sent in every slice header. This allows maximum flexibility but typically requires more bits than the other methods.
- Type 1: A 'cycle' of POC increments is set up in the sequence parameter set and POC changes according to this cycle unless otherwise signalled in the slice header using a Delta offset. The cycle defines the interval between frames used for reference, plus a POC offset to frames not used for reference.
- Type 2: POC is derived directly from *frame_num* and display order is the same as decoding order.

Display Order Example, Type 0:

Frame pictures, one frame per slice, display order IBPBPBPBPB. . . (Figure 5.4).

In this example, the B slices, bipredicted slices in which each macroblock or partition can be predicted from up to two reference pictures, are not used for reference prediction of any other pictures. POC increments by two for every complete frame, i.e. every two fields. Note that *frame_num* increments **after** each reference picture is transmitted. In this example, only the I and P pictures are reference pictures. Increments in *frame_num* are indicated in bold type.

Slice	Type	Used for reference	*frame_num*	POC LSBs	Display order
1st	I	Yes	**0**	0	0
2nd	P	Yes	**1**	4	2
3rd	B	No	**2**	2	1
4th	P	Yes	2	8	4
5th	B	No	**3**	6	3
6th	P	Yes	3	12	6
7th	B	No	**4**	10	**5**
8th	P	Yes	4	16	8
. . .	. . .				

Display Order Example, Type 1:

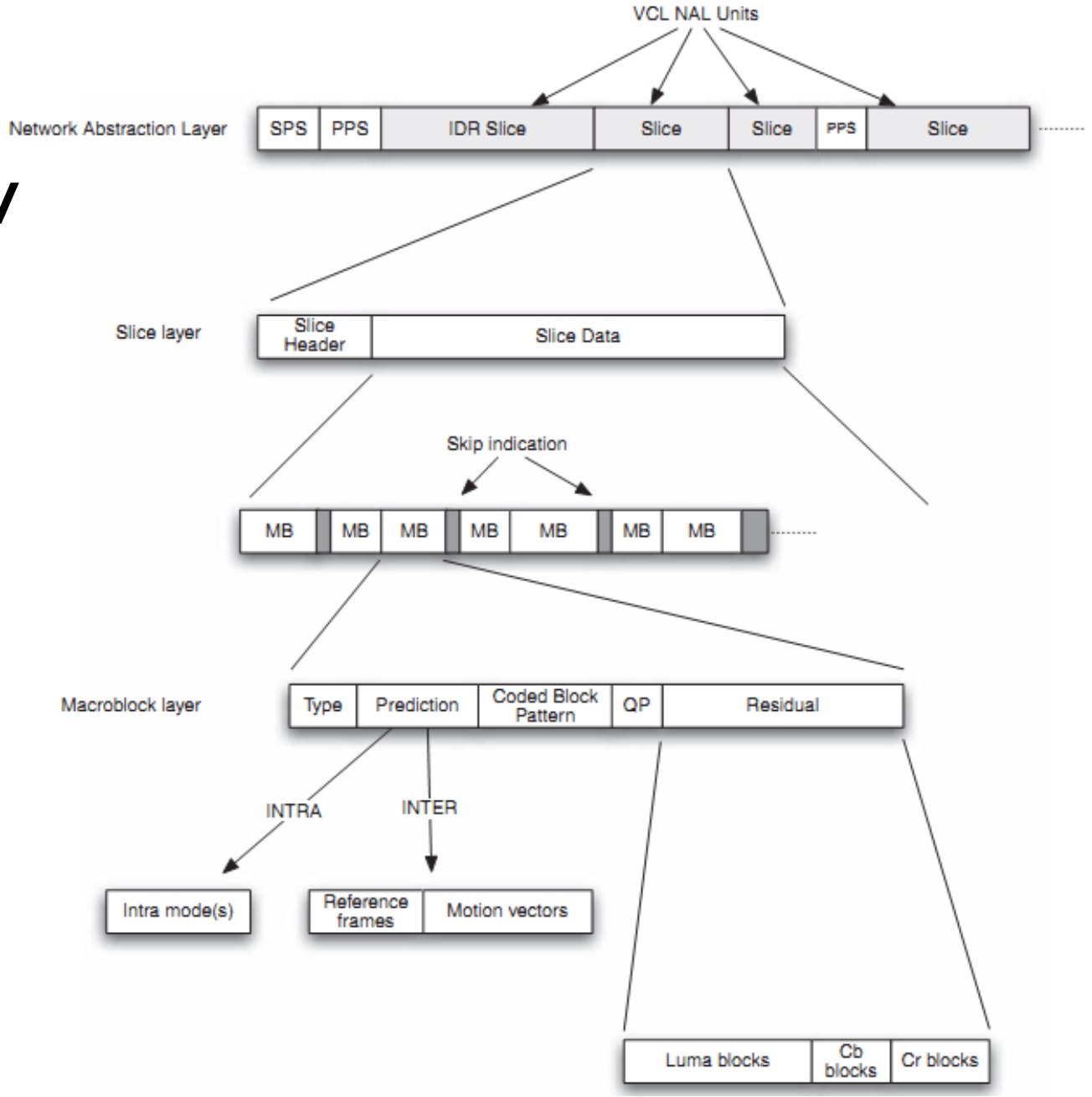
Frame pictures, one frame per slice, display order: IBBPBBPBBPB...

In this example, the B slices are not used for reference. The POC cycle consists of one reference frame, with an offset from the previous reference frame to the next reference frame of +6 and an offset from the previous reference frame to the next non-reference frame of −4.

Slice	Type	Used for reference	*frame_num*	Delta POC	Offset	POC	Display order
1st	I	Yes	**0**	0	0	0	0
2nd	P	Yes	**1**	0	+6	6	3
3rd	B	No	**2**	0	−4	2	1
4th	B	No	2	2	−4+2	4	2
5th	P	Yes	2	0	+6	12	6
6th	B	No	**3**	0	−4	8	4
7th	B	No	3	2	−4+2	10	5
8th	P	Yes	3	0	+6	18	9
...	...						

The 4th and 7th slices have a delta POC, signalled in the slice header, of +2. The POC for these slices is calculated as: POC = expected POC + 2.

Syntax Overview



Parameter Sets

SPS

(Sequence~)

Table 5.7 Sequence Parameter Set example

Parameter	Binary code	Symbol	Discussion
profile_idc	1000010	66	Baseline Profile
constrained_set0_flag	0	0	Bistream might not obey all the constraints of the Baseline Profile
constrained_set1_flag	0	0	As above, Main Profile
constrained_set2_flag	0	0	As above, Extended Profile
constrained_set3_flag	0	0	Used to specify the special case of Level 1b
reserved_zero_4bits	0	0	Not used
level_idc	11110	30	Level 3
seq_parameter_set_id	1	0	Sequence Parameter Set 0
log2_max_frame_num_minus4	1	0	frame_num will not exceed 16.
pic_order_cnt_type	1	0	Default POC
log2_max_pic_order_cnt_lsb_minus4	1	0	LSB of POC will not exceed 16.
num_ref_frames	1011	10	Up to 10 reference frames.
gaps_in_frame_num_value_allowed_flag	0	0	No gaps in frame_num.
pic_width_in_mbs_minus1	1011	10	11 macroblocks wide = QCIF
pic_height_in_map_units_minus1	1001	8	9 MBs high = QCIF
frame_mbs_only_flag	1	1	No field slices or field MBs
direct_8 × 8_inference_flag	1	1	Specifies how certain B macroblock motion vectors are derived
frame_cropping_flag	0	0	Frames are not cropped
vui_parameters_present_flag	0	0	VUI parameters not present

PPS (Picture ~)

Table 5.8 Picture Parameter Set example

Parameter	Binary code	Symbol	Discussion
pic_parameter_set_id	1	0	Picture Parameter Set 0
seq_parameter_set_id	1	0	Use SPS 0
entropy_coding_mode_flag	0	0	CAVLC entropy coding
pic_order_present_flag	0	0	POC not present
num_slice_groups_minus1	1	0	One slice group
num_ref_idx_l0_active_minus1	1010	9	10 reference pictures in list0
num_ref_idx_l1_active_minus1	1010	9	10 reference pictures in list1
weighted_pred_flag	0	0	Weighted prediction not used
weighted_bipred_idc	0	0	Weighted biprediction not used
pic_init_qp_minus26	1	0	Initial QP (luma) = 26
pic_init_qs_minus26	1	0	Initial SI/SP QP=26
chroma_qp_index_offset	1	0	No chroma QP offset
deblocking_filter_control_present_flag	0	0	Use default filter parameters
constrained_intra_pred_flag	0	0	Intra prediction is not constrained
redundant_pic_cnt_present_flag	0	0	Redundant picture count parameter is not used

Using parameter sets: Example

SPS0 is sent at the start of a sequence, followed by PPS0 and PPS1, both of which 'inherit' SPS0 (Figure 5.12). PPS0 is activated by IDR slice 0, which means that SPS0 becomes active at the same time. Slices 1 and 2 use the parameters of PPS0 and SPS0. PPS1 is activated by slice 3, making PPS0 inactive, hence slice 3 uses the parameters of PPS1 and SPS0; PPS0 is activated again by slice 4, making PPS1 inactive.

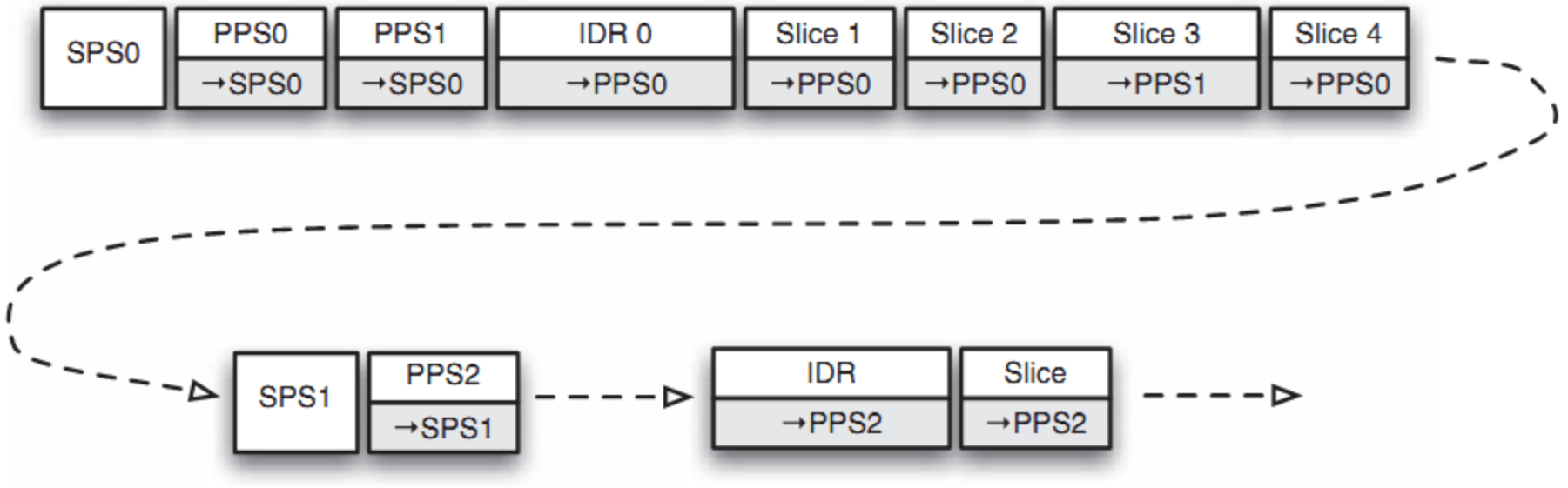


Figure 5.12 Example: Sequence and Picture Parameter Sets

Slice Layer
Slice Header + Slice Data

Table 5.9 Slice types in H.264

Slice type	Contains macroblock types	Notes
I (including IDR)	I only	Intra prediction only.
P	I and/or P	Intra prediction (I) and/or prediction from one reference per macroblock partition (P).
B	I, P and/or B	Intra prediction (I), prediction from one reference (P) or biprediction, i.e. prediction from two references (B)
SP	P and/or I	Switching P slice, see Chapter 8
SI	SI	Switching I slice, see Chapter 8

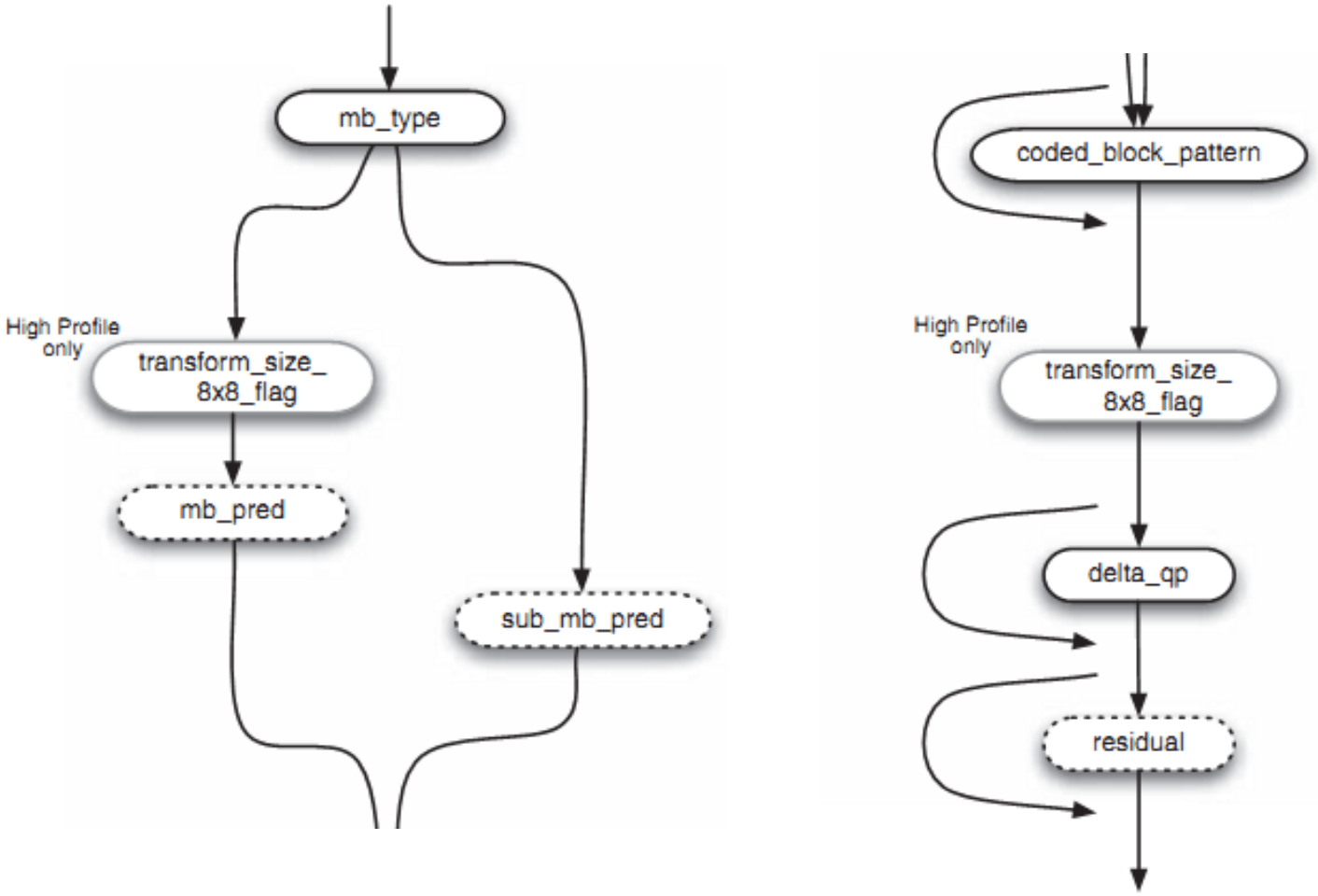
Table 5.10 Slice Header, IDR/Intra, Frame 0

Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB is at position 0, the top-left position in the slice.
slice_type	1000	7	I slice, contains only I MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	0	0	Slice is in frame 0
idr_pic_id	1	0	IDR #0 : only present in IDR picture
pic_order_cnt_lsb	0	0	Picture order count = 0
no_output_of_prior_pics_flag	0	0	Not used
long_term_reference_flag	0	0	Not used
slice_qp_delta	1000	4	QP = initial QP + 4 = 30

Table 5.11 Slice Header, Inter, Frame 1

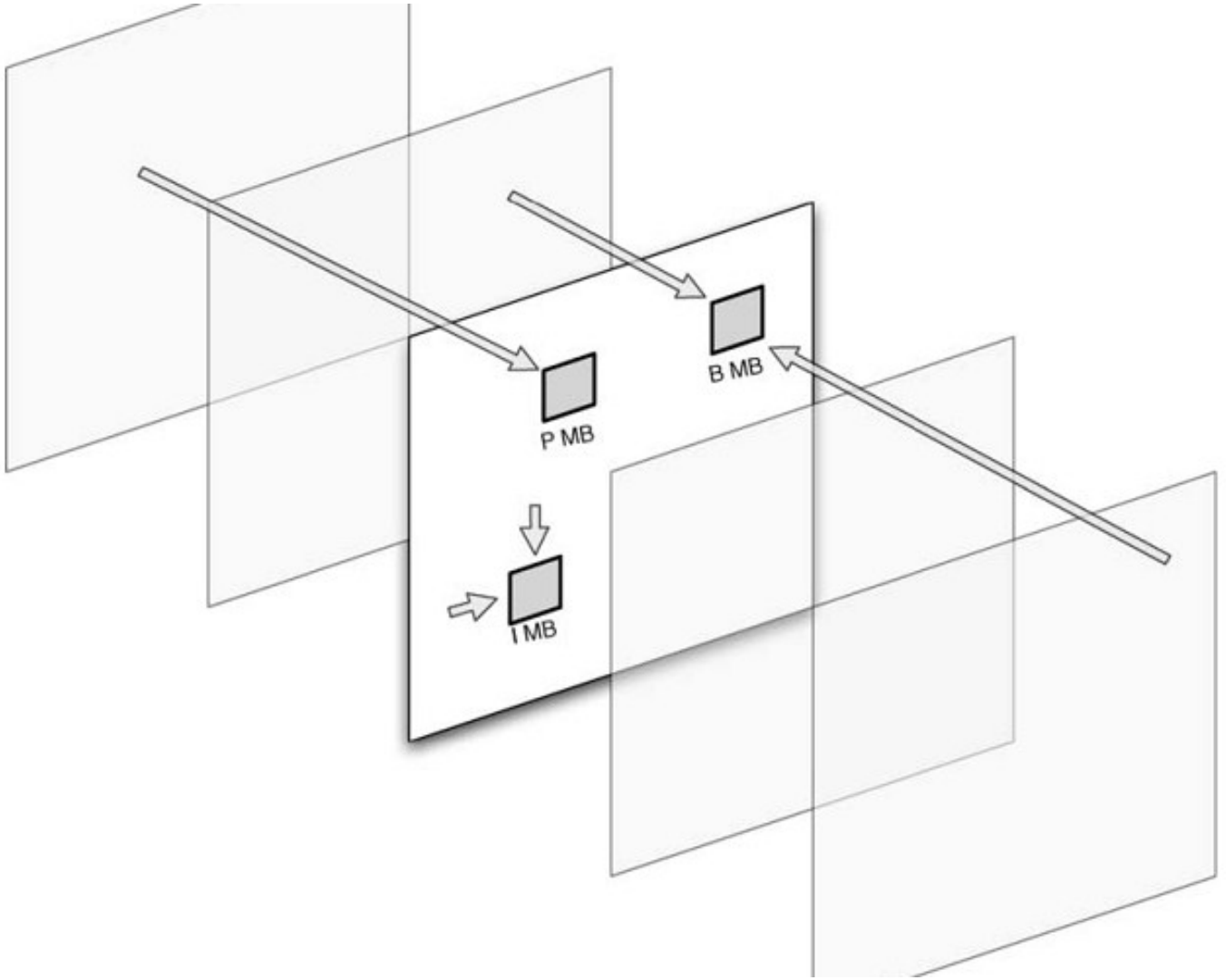
Parameter	Binary code	Symbol	Discussion
first_mb_in_slice	1	0	First MB at position 0
slice_type	110	5	P slice, can contain I or P MBs
pic_parameter_set_id	1	0	Use PPS 0
frame_num	1	1	Frame 1
pic_order_cnt_lsb	10	2	Picture 2
num_ref_idx_active_override_flag	1	1	Default number of reference pictures overridden by following parameter
num_ref_idx_l0_active_minus1	1	0	One reference picture in List 0
ref_pic_list_reordering_flag_l0	0	0	No re-ordering of reference pictures
adaptive_ref_pic_buffering_flag	0	0	Reference pictures handled as 'first in / first out' (default)
slice_qp_delta	1000	4	QP = 26 + 4 = 30

Macro-Block(MB) Layer



H.264/AVC Prediction

Macro-Block Prediction



Intra Prediction

Table 6.1 Intra prediction types

Intra prediction block size	Notes
16 × 16 (luma)	A single 16 × 16 prediction block P is generated. Four possible prediction modes.
8 × 8 (luma)	An 8 × 8 prediction block P is generated for each 8 × 8 luma block. Nine possible prediction modes. 'High' Profiles only.
4 × 4 (luma)	A 4 × 4 prediction block P is generated for each 4 × 4 luma block. Nine possible prediction modes.
Chroma	One prediction block P is generated for each chroma component. Four possible prediction modes. The same prediction mode is used for both chroma components.

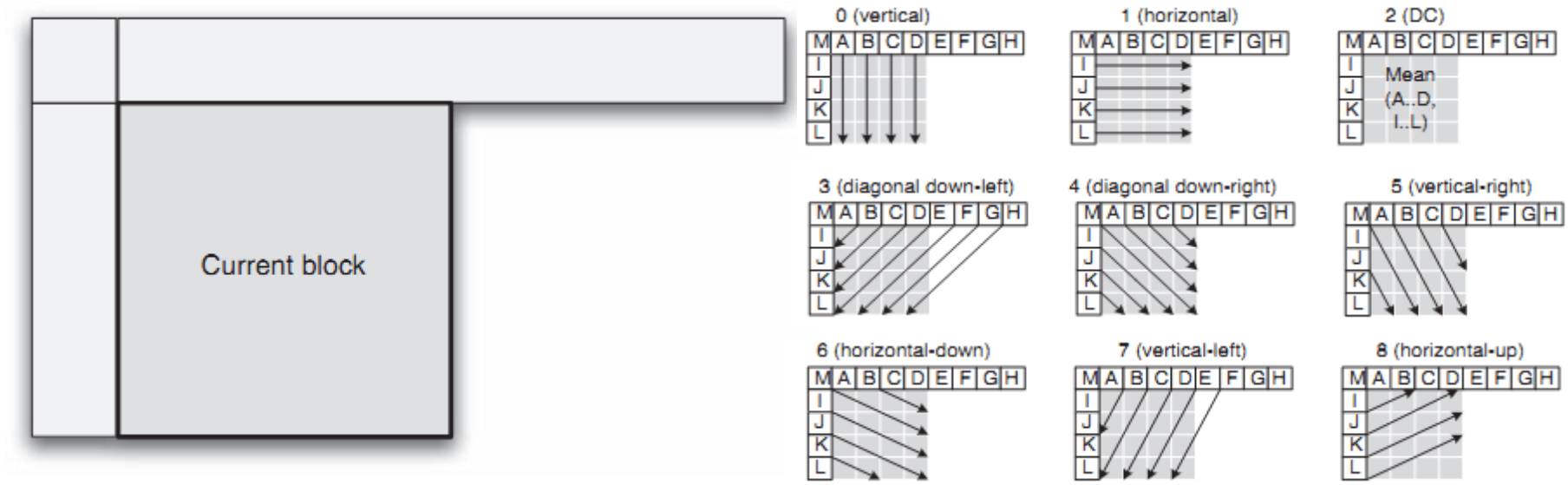


Figure 6.3 Intra prediction source samples, 4 × 4 or 8 × 8 luma blocks

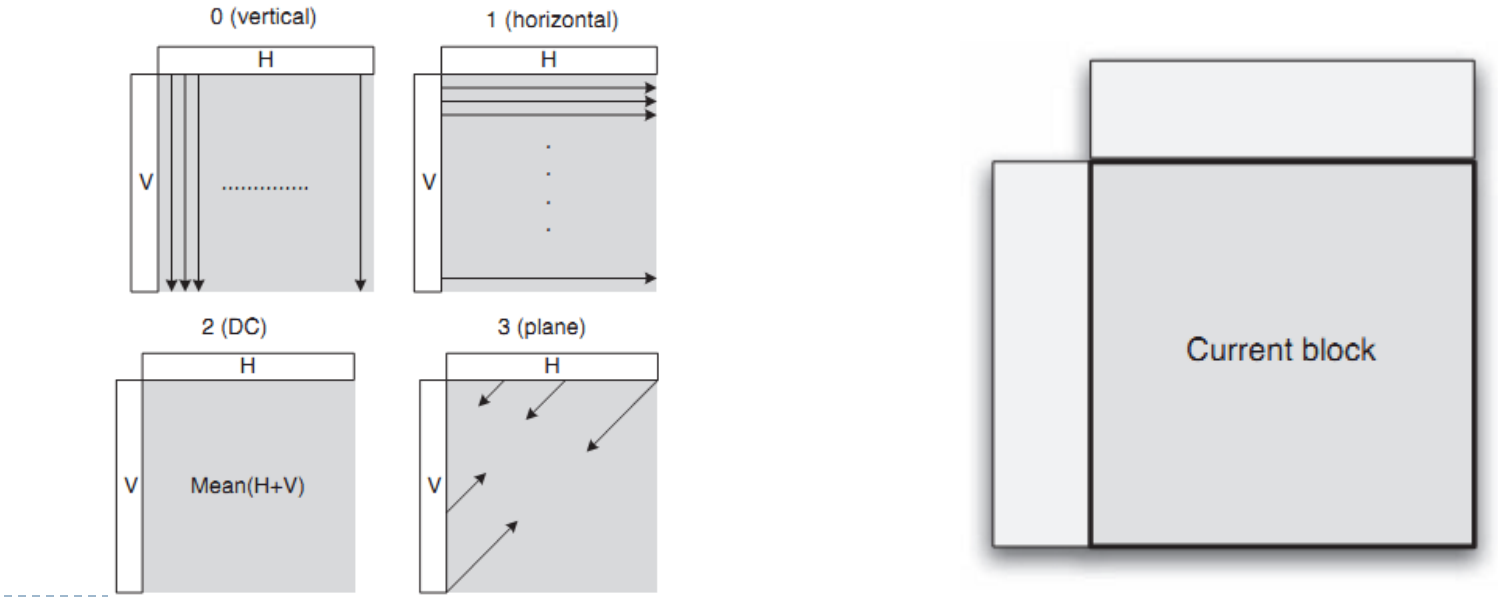


Figure 6.4 Intra prediction source samples, chroma or 16 × 16 luma blocks

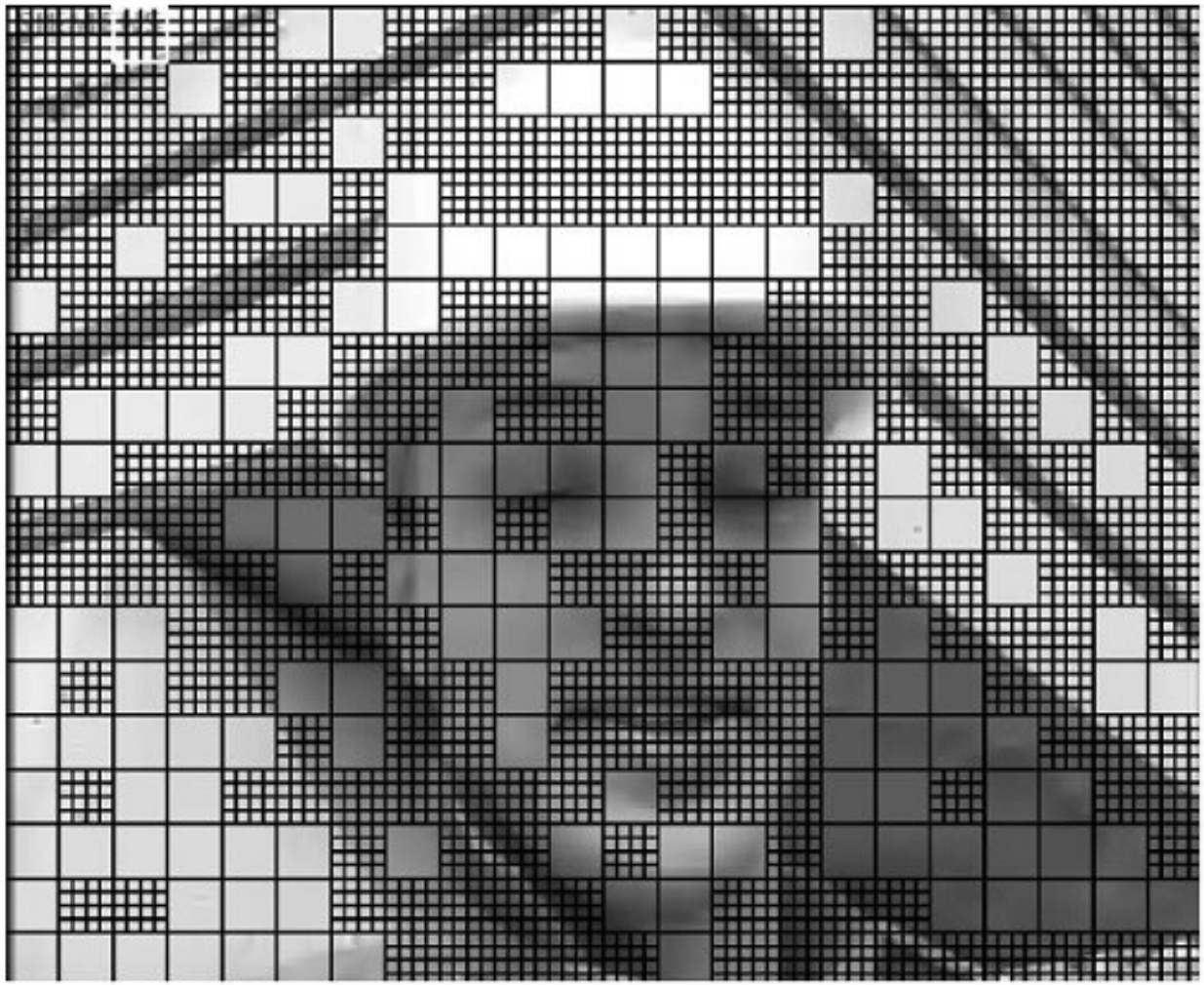


Figure 6.5 Example of intra block size choices, CIF, Baseline Profile. Reproduced by permission of Elecard.

Inter Prediction

A macroblock in frame *n* is shown in Figure 6.17. The macroblock is divided into two partitions, each consisting of 8 × 16 luma samples and corresponding chroma samples. The left partition (A)

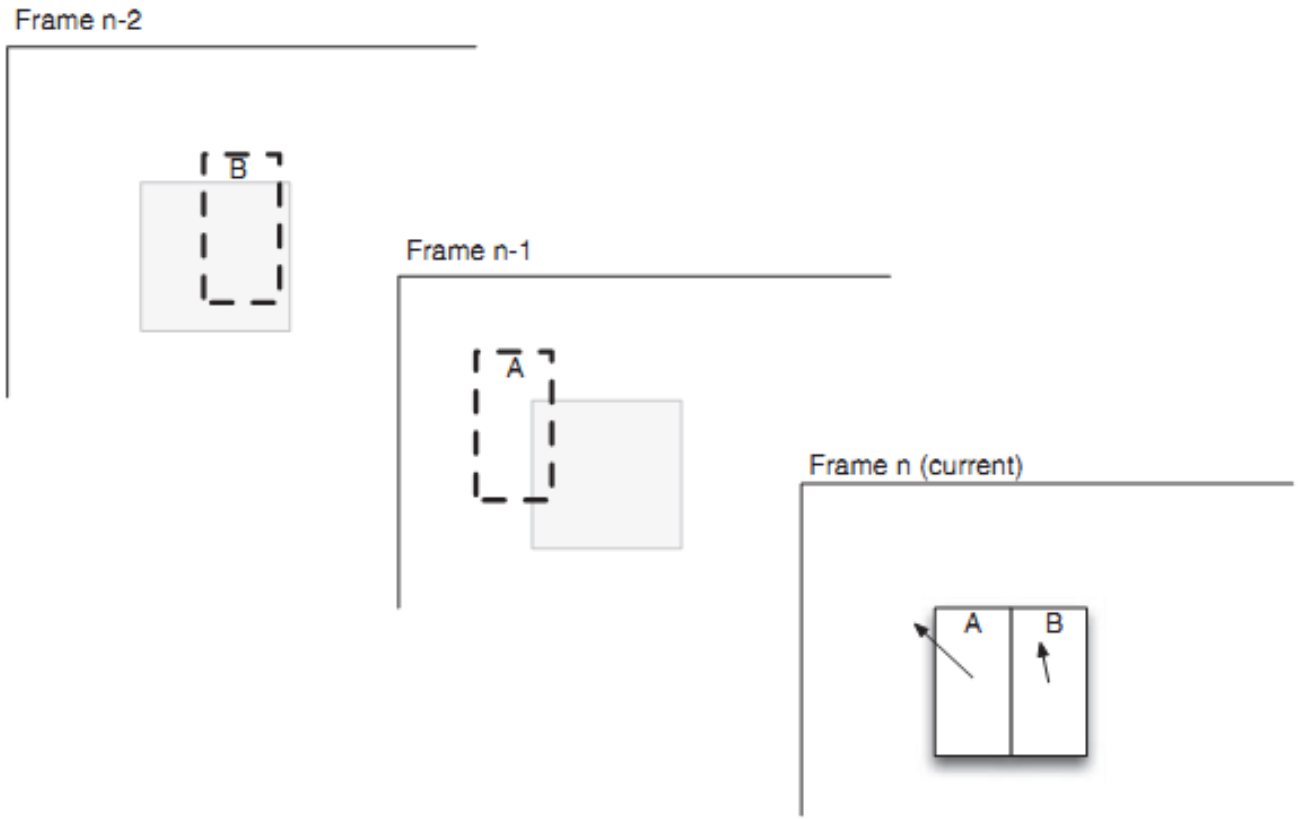


Figure 6.17 P macroblock prediction example

is predicted from a region in the previous frame, frame *n−1*, and the right partition (B) is predicted from a region in frame *n−2*. Partition A has a motion vector (−6.5, −5.75), i.e. the reference region is offset by −6.5 samples in the x direction (left) and −5.75 samples in the y direction (up). Partition B has motion vector (−1.25, −4), i.e. −1.25 samples in the x direction (left) and −4 samples in the y direction (up).

1. Interpolate the picture(s) in the Decoded Picture Buffer to generate $^1/_4$-sample positions in the luma component and 1/8-sample positions in the chroma components. (section 6.4.2).
2. Choose an inter prediction mode from the following options:
   (a) Choice of reference picture(s), previously-coded pictures available as sources for prediction. (section 6.4.1).
   (b) Choice of macroblock partitions and sub-macroblock partitions, i.e. prediction block sizes. (section 6.4.3).
   (c) Choice of prediction types:
      (i) prediction from one reference picture in list 0 for P or B macroblocks or list 1 for B macroblocks only (section 6.4.5.1).
      (ii) bi-prediction from two reference pictures, one in list 0 and one in list 1, B macroblocks only, optionally using weighted prediction (section 6.4.5.2).
3. Choose motion vector(s) for each macroblock partition or sub-macroblock partition, one or two vectors depending on whether one or two reference pictures are used.
4. Predict the motion vector(s) from previously-transmitted vector(s) and generate motion vector difference(s). Optionally, use Direct Mode prediction, B macroblocks only. (section 6.4.4).
5. Code the macroblock type, choice of prediction reference(s), motion vector difference(s) and residual. (Chapters 5 and 7).
6. Apply a deblocking filter prior to storing the reconstructed picture as a prediction reference for further coded pictures. (section 6.5).

Table 6.3 Reference picture sources

Slice type	MB type	Reference picture sources
P	P	List0 (P slice)
B	P	List0 (B slice)
B	B	List0 (B slice) and List1 (B slice)

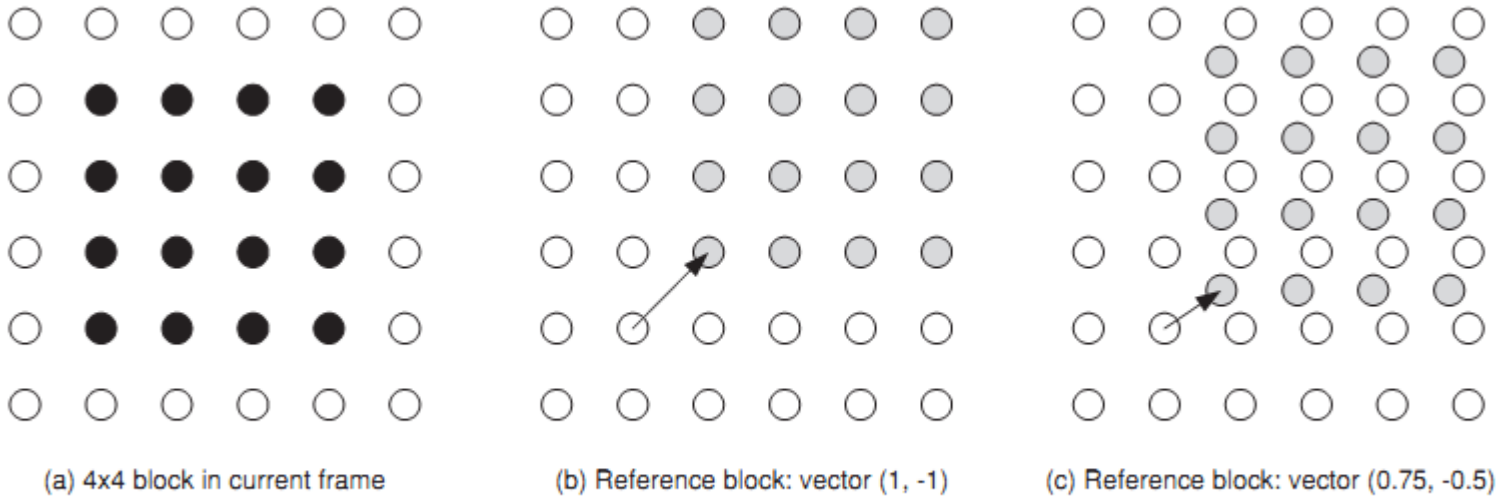


Figure 6.18 Example of integer and sub-pixel prediction

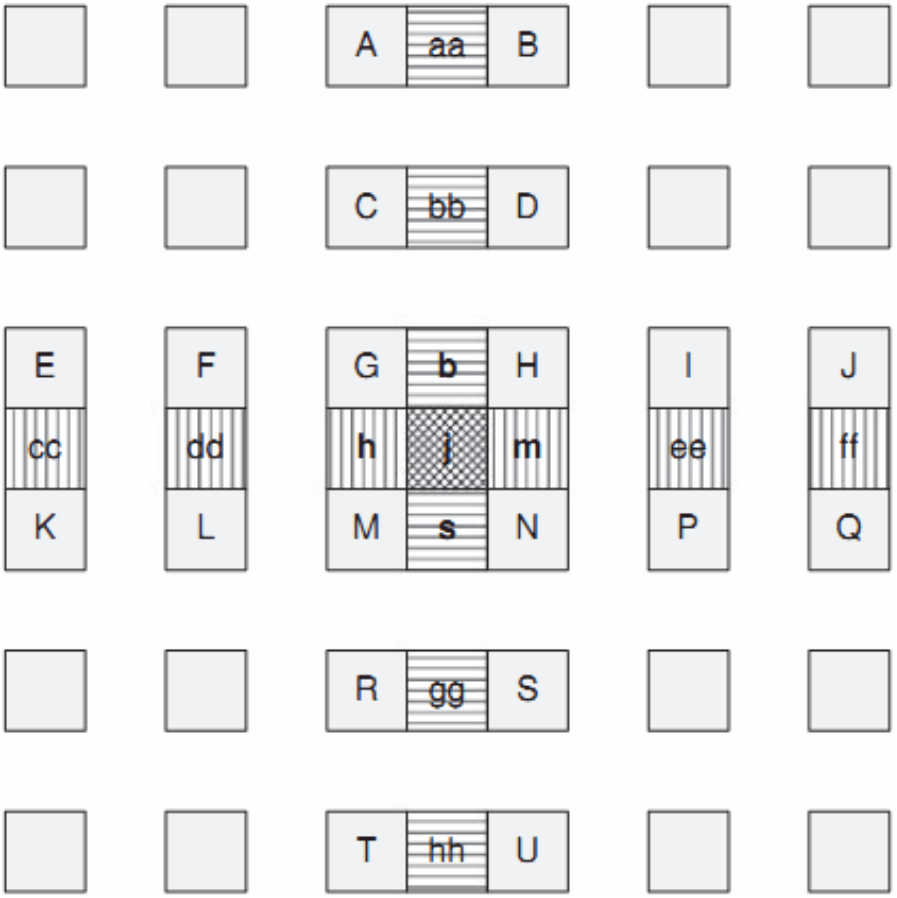


Figure 6.25 Interpolation of luma half-pel positions

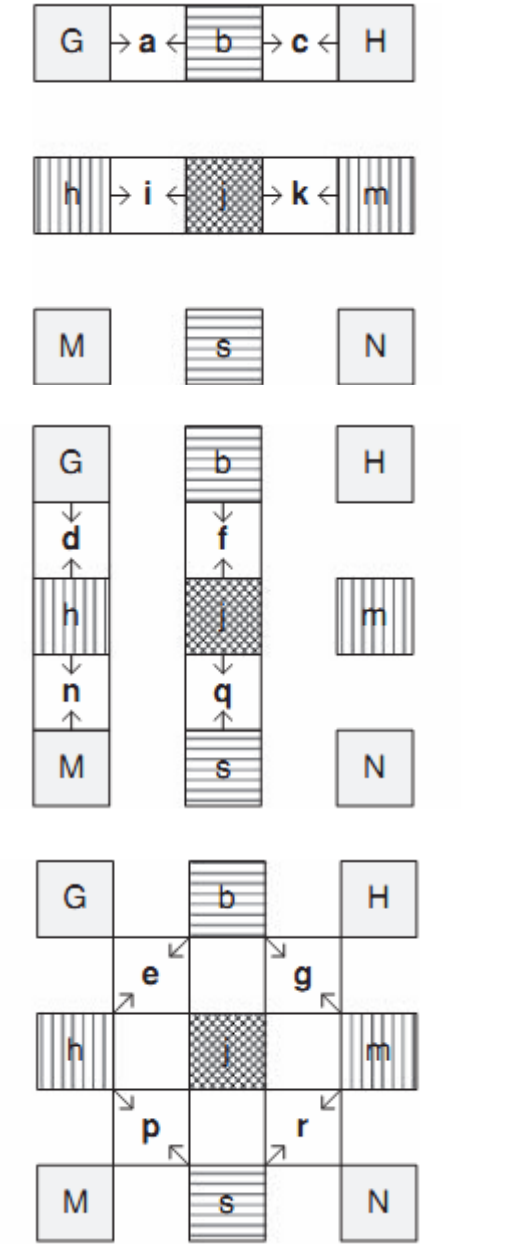


Figure 6.26 Interpolation of luma quarter-pel positions

Sub-MB Partition

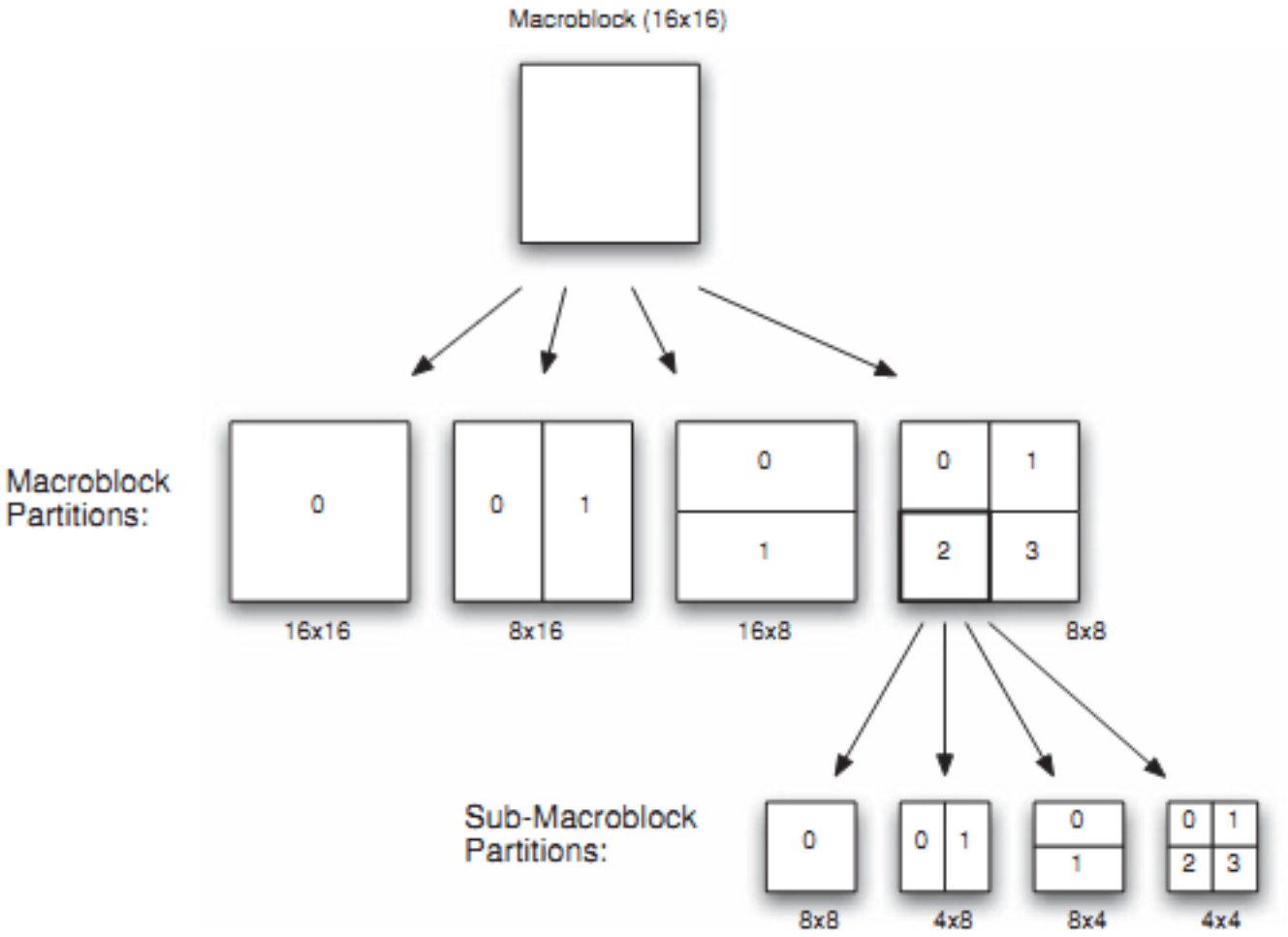
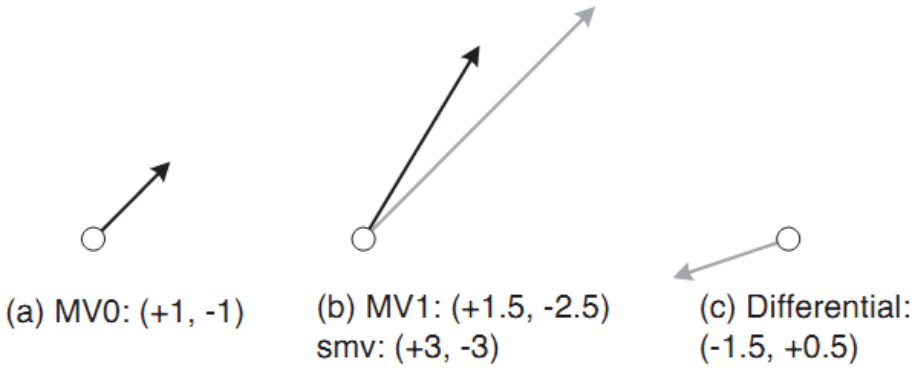
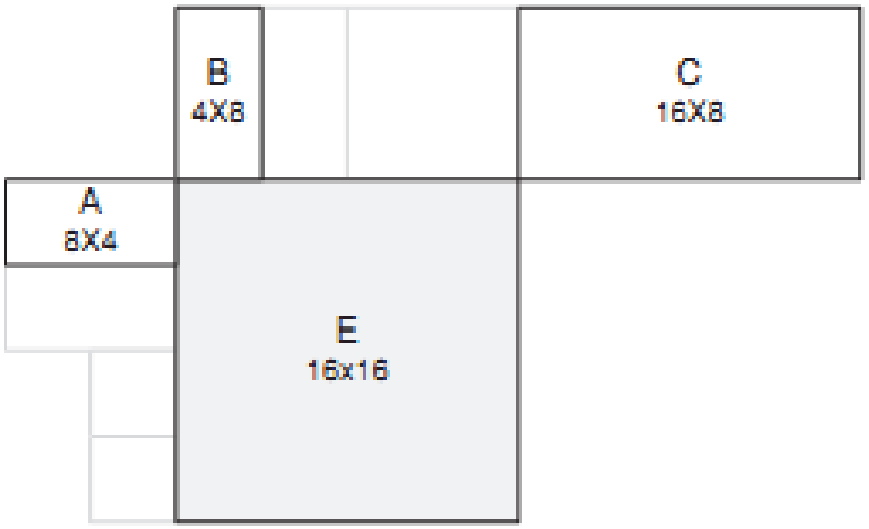


Figure 6.29 Macroblock partitions and sub-macroblock partitions

Motion Vector





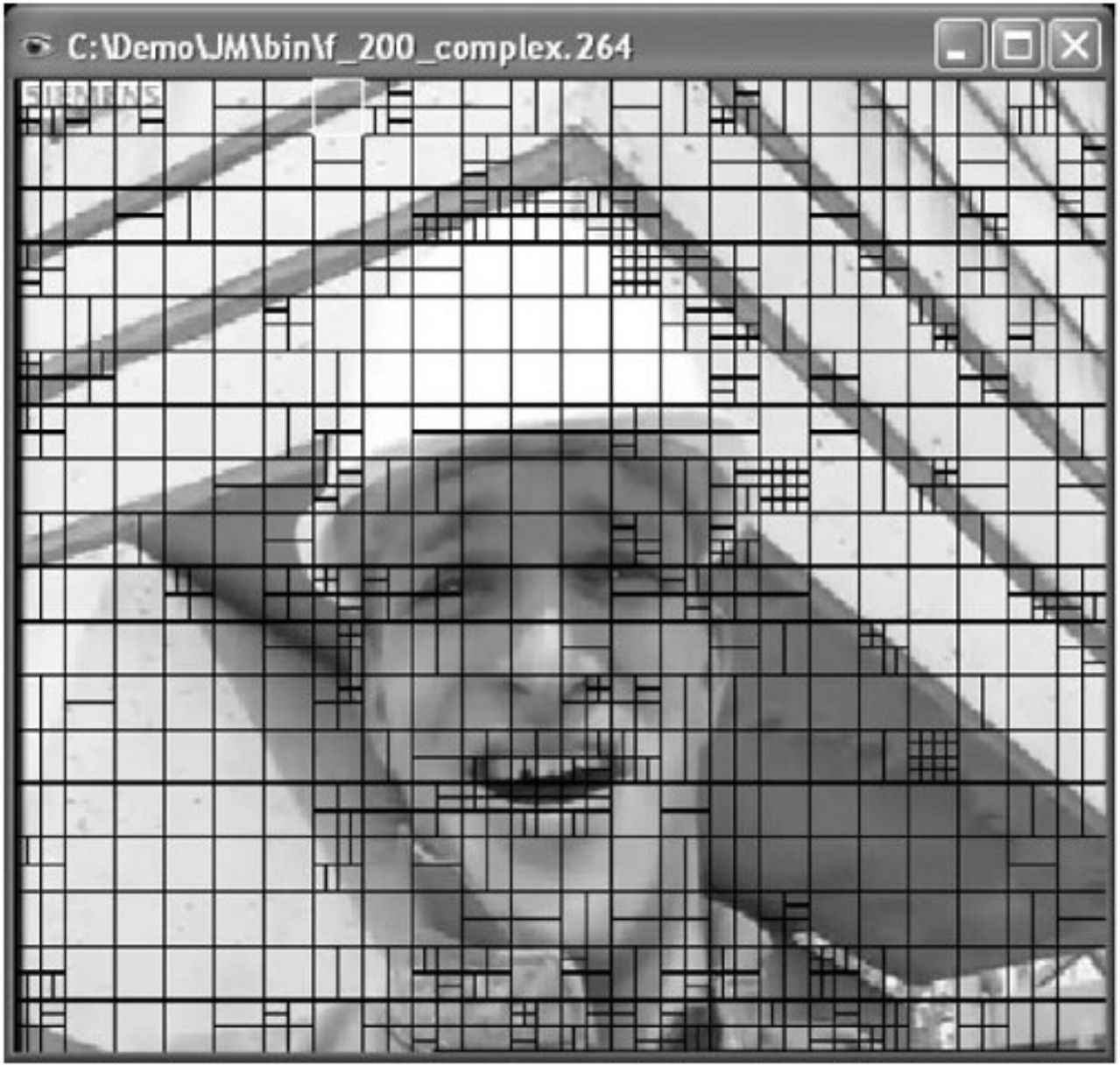


Figure 6.37 P slice showing partition choices. Reproduced by permission of Elecard

Prediction Structures

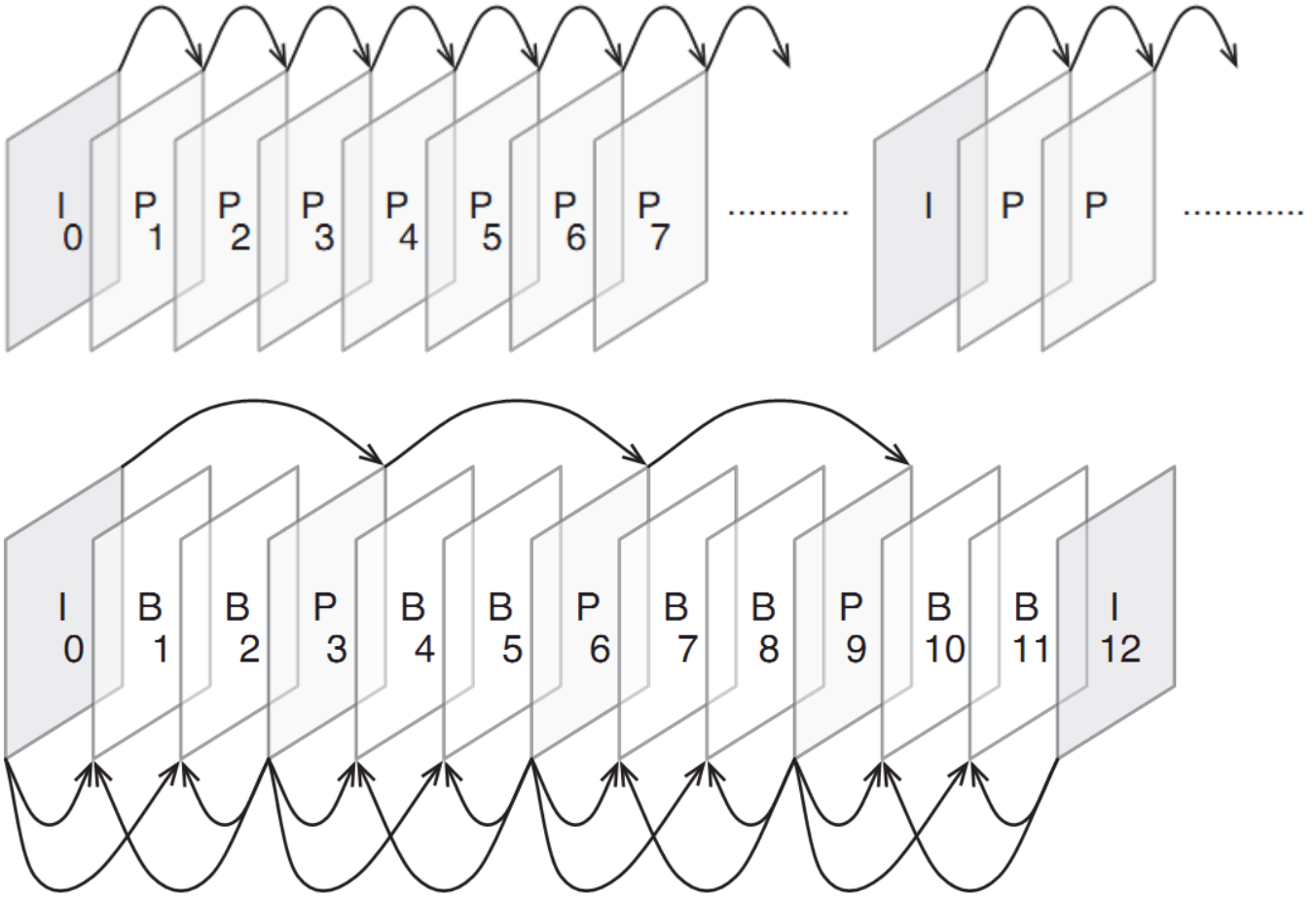


Figure 6.42 'Classic' Group of Pictures prediction structure

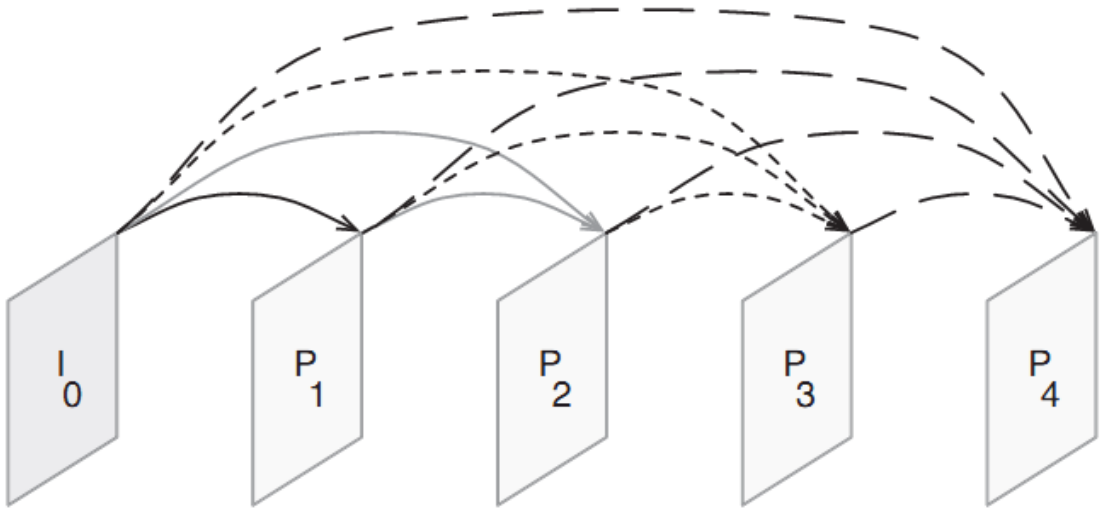


Figure 6.43 IPPP . . . with multiple reference pictures

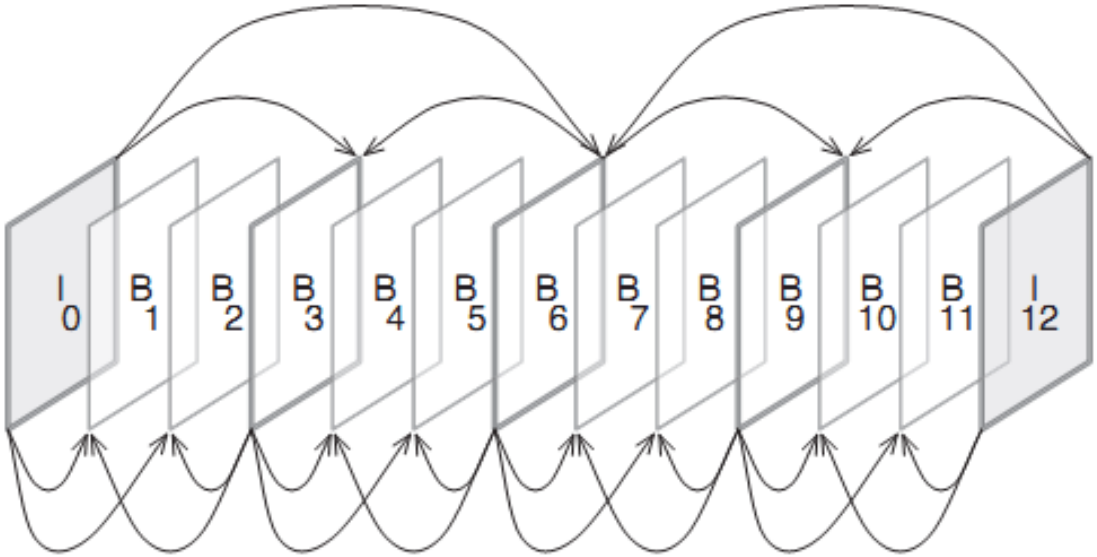


Figure 6.44 Hierarchical GOP structure

Loop Filter

Figure 6.48 Original frame, 'violin' frame 2

Figure 6.49 Reconstructed, QP = 36, no filter

Figure 6.50 Reconstructed, QP = 36, with filter

The choice of filtering outcome depends on the **boundary strength** and on the **gradient** of image samples across the boundary. The boundary strength parameter **Bs** is chosen according to the following rules:

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; p or q contain coded coefficients	Bs = 2
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

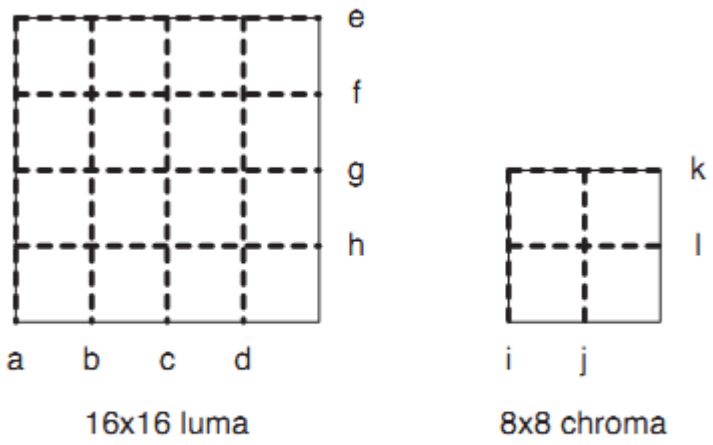


Figure 6.45 Edge filtering order in a macroblock

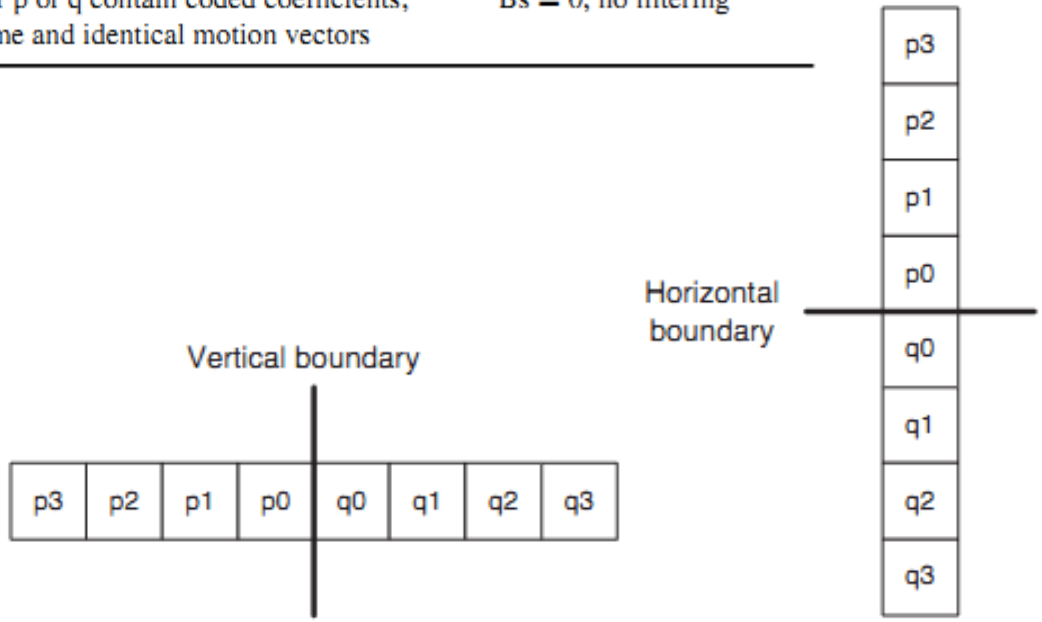


Figure 6.46 Pixels adjacent to vertical and horizontal boundaries

H.264/AVC Transform & Coding

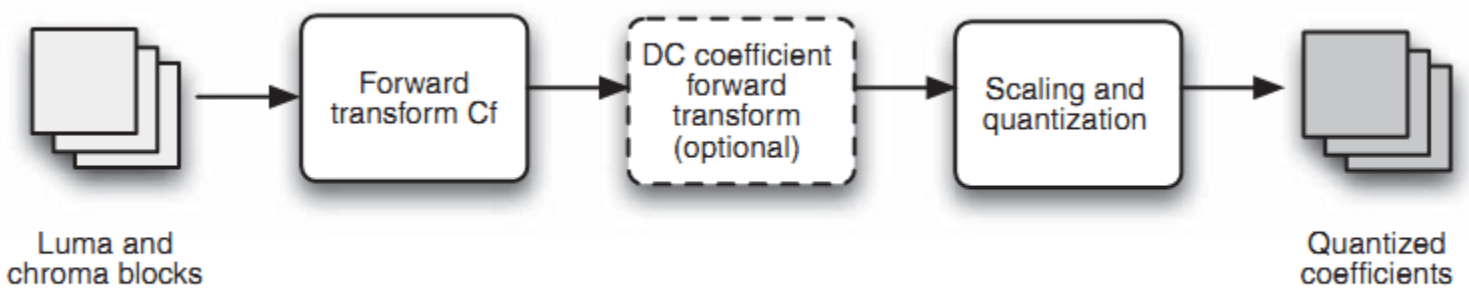


Figure 7.2 Forward transform and quantization

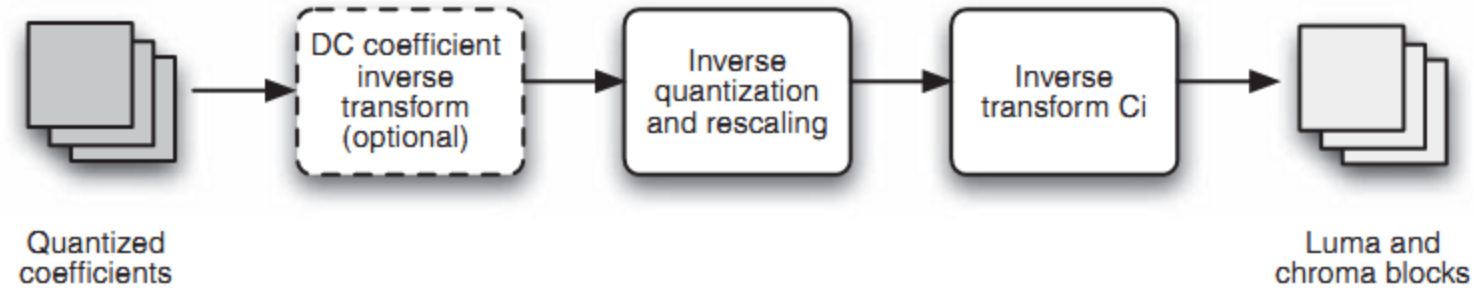


Figure 7.1 Re-scaling and inverse transform

Luma Intra 16x16

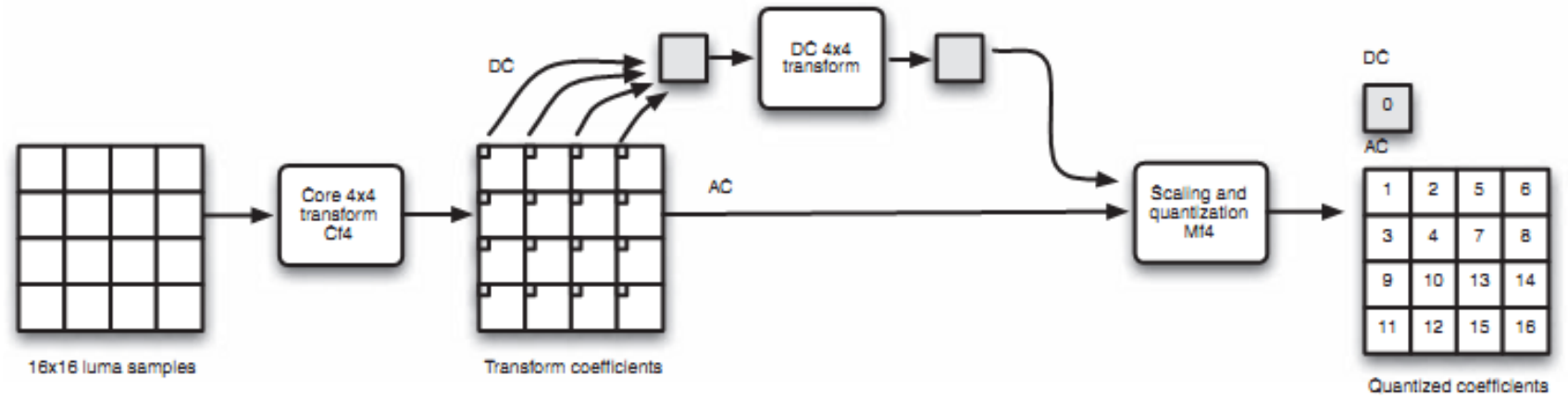


Figure 7.5 Luma forward transform : Intra 16 × 16 mode

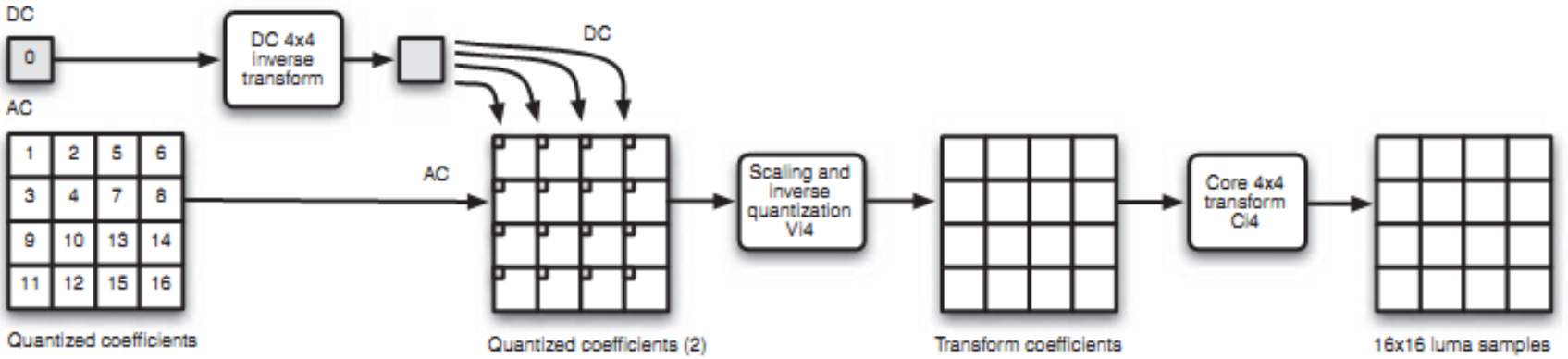


Figure 7.6 Luma inverse transform : Intra 16 × 16 mode

Luma Intra 4x4/8x8

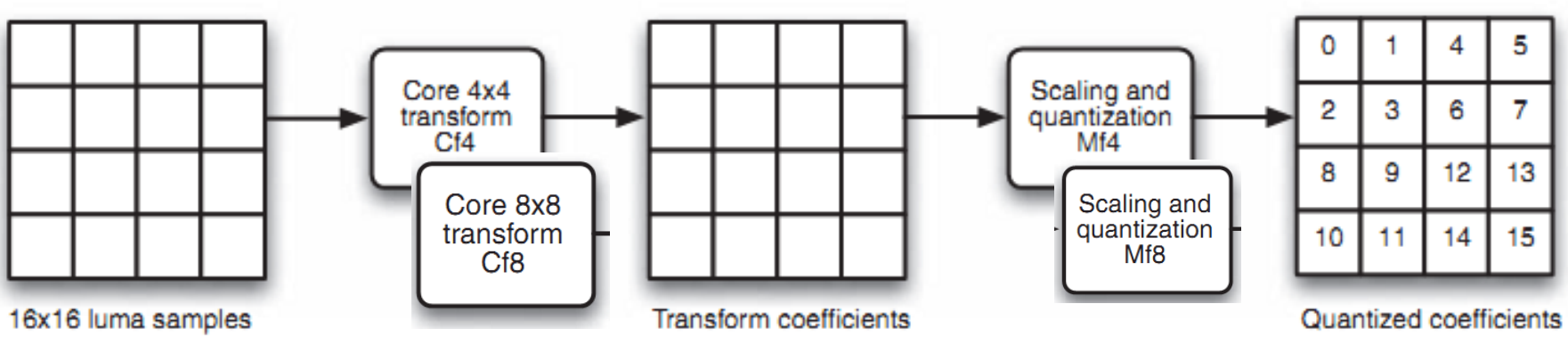


Figure 7.3 Luma forward transform : default

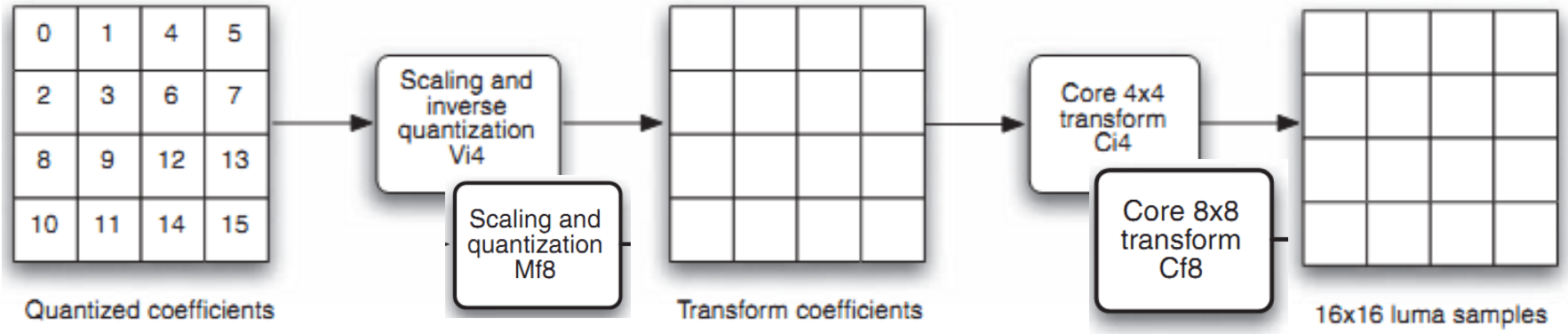


Figure 7.4 Luma inverse transform : default

Chroma Intra

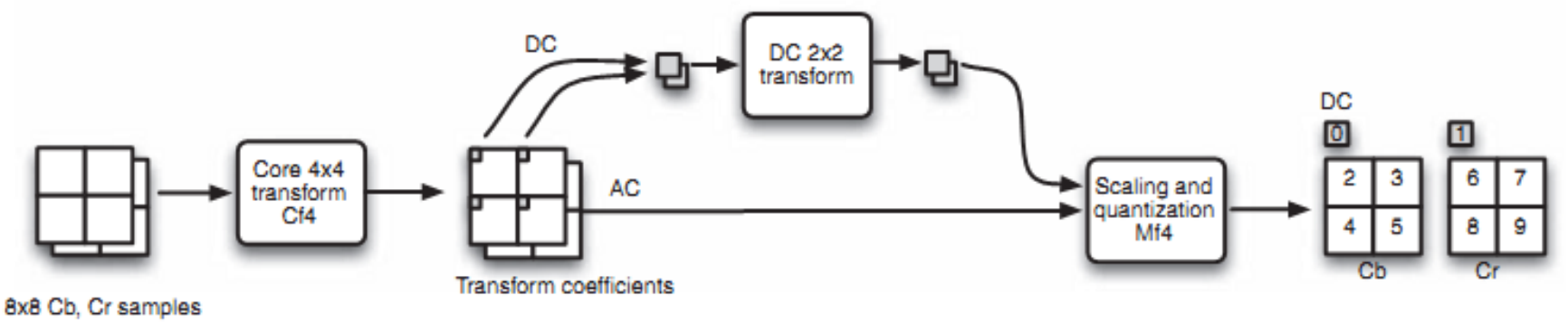


Figure 7.9 Chroma forward transform : 4:2:0 macroblock

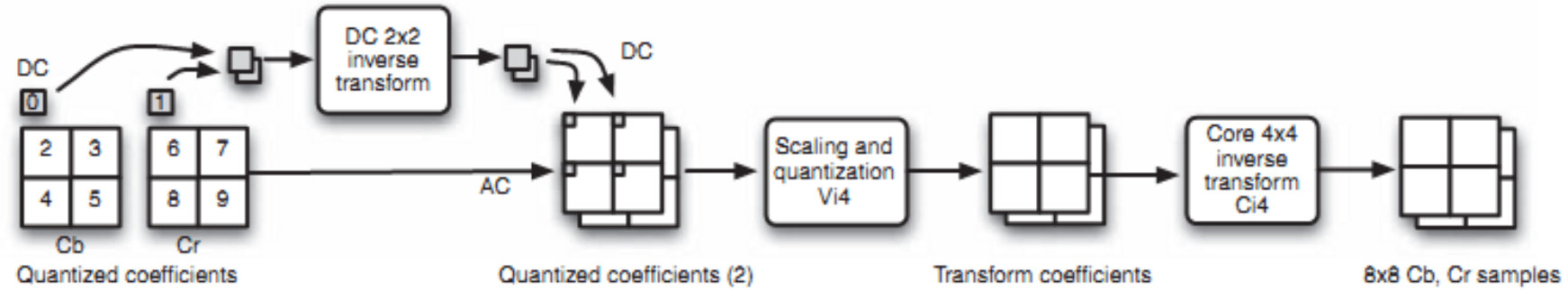


Figure 7.10 Chroma inverse transform : 4:2:0 macroblock

Integer DCT transformation

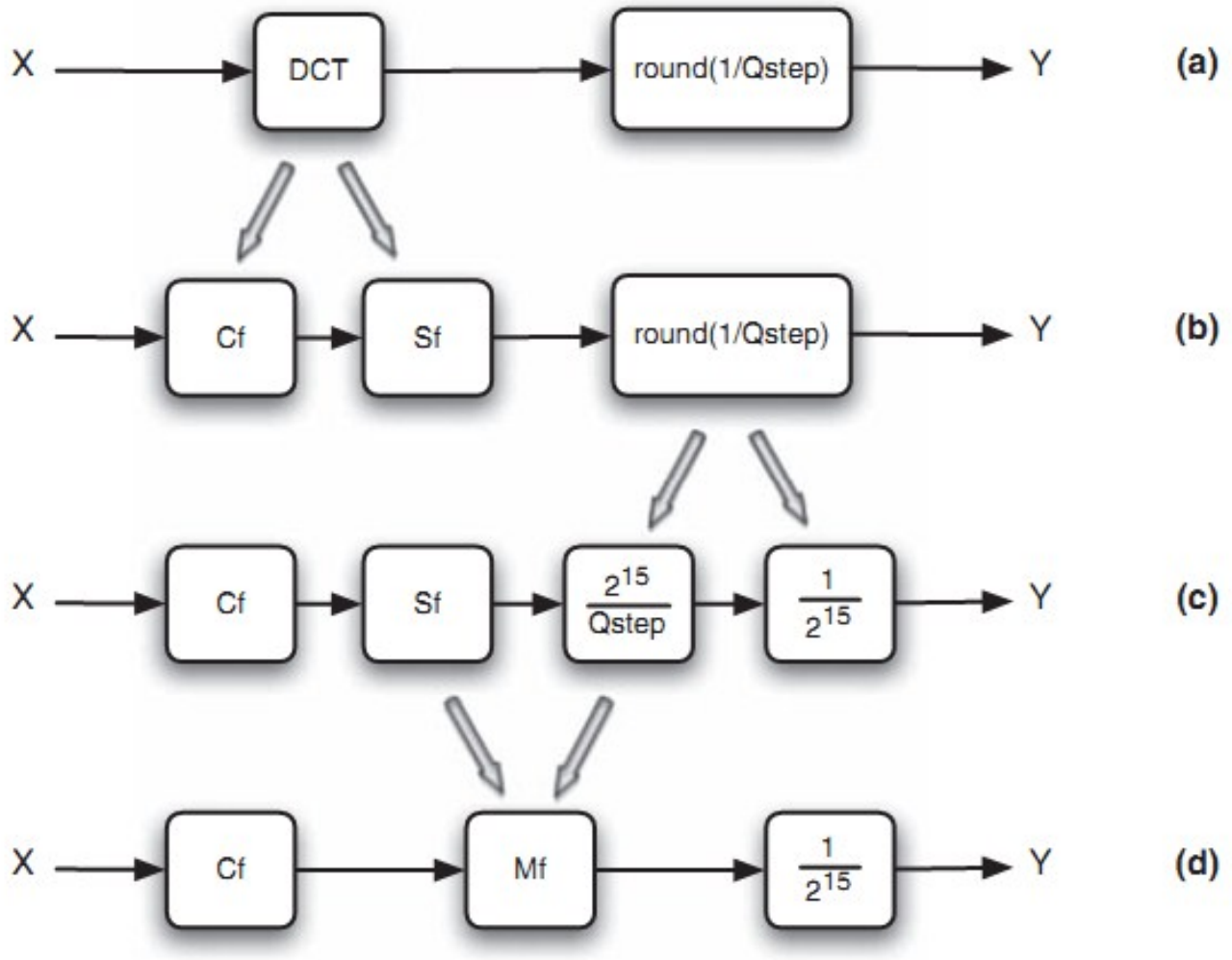


Figure 7.13 Development of the forward transform and quantization process

$$\mathbf{Y} = \mathbf{A} \cdot \mathbf{X} \cdot \mathbf{A}^{\mathbf{T}}$$

$$\mathbf{A} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}, \quad \begin{aligned} a &= {}^{1}/_{2} \\ b &= \sqrt{{}^{1}/_{2}} \cos {}^{\pi}/_{8} = 0.6532\ldots \\ c &= \sqrt{{}^{1}/_{2}} \cos {}^{3\pi}/_{8} = 0.2706\ldots \end{aligned}$$

$$\mathbf{A}_1 = \mathbf{C}_{f4} \bullet \mathbf{R}_{f4} \quad \text{where } \mathbf{R}_{f4} = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{bmatrix}$$

$$\mathbf{C}_{f4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

$$\mathbf{A}_1 = \mathbf{C}_{f4} \bullet \mathbf{R}_{f4} \quad \text{where } \mathbf{R}_{f4} = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{bmatrix}$$

$$= [\mathbf{C}_{f4} \cdot \mathbf{X} \cdot \mathbf{C}_{f4}^{\mathbf{T}}] \bullet [\mathbf{R}_{f4} \bullet \mathbf{R}_{f4}^{\mathbf{T}}]$$

$$= [\mathbf{C}_{f4} \cdot \mathbf{X} \cdot \mathbf{C}_{f4}^{\mathbf{T}}] \bullet \mathbf{S}_{f4}$$

Where,

$$\mathbf{S}_{f4} = \mathbf{R}_{f4} \bullet \mathbf{R}_{f4}^{T} = \begin{bmatrix} 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \\ 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \end{bmatrix}$$

Integer IDCT transformation

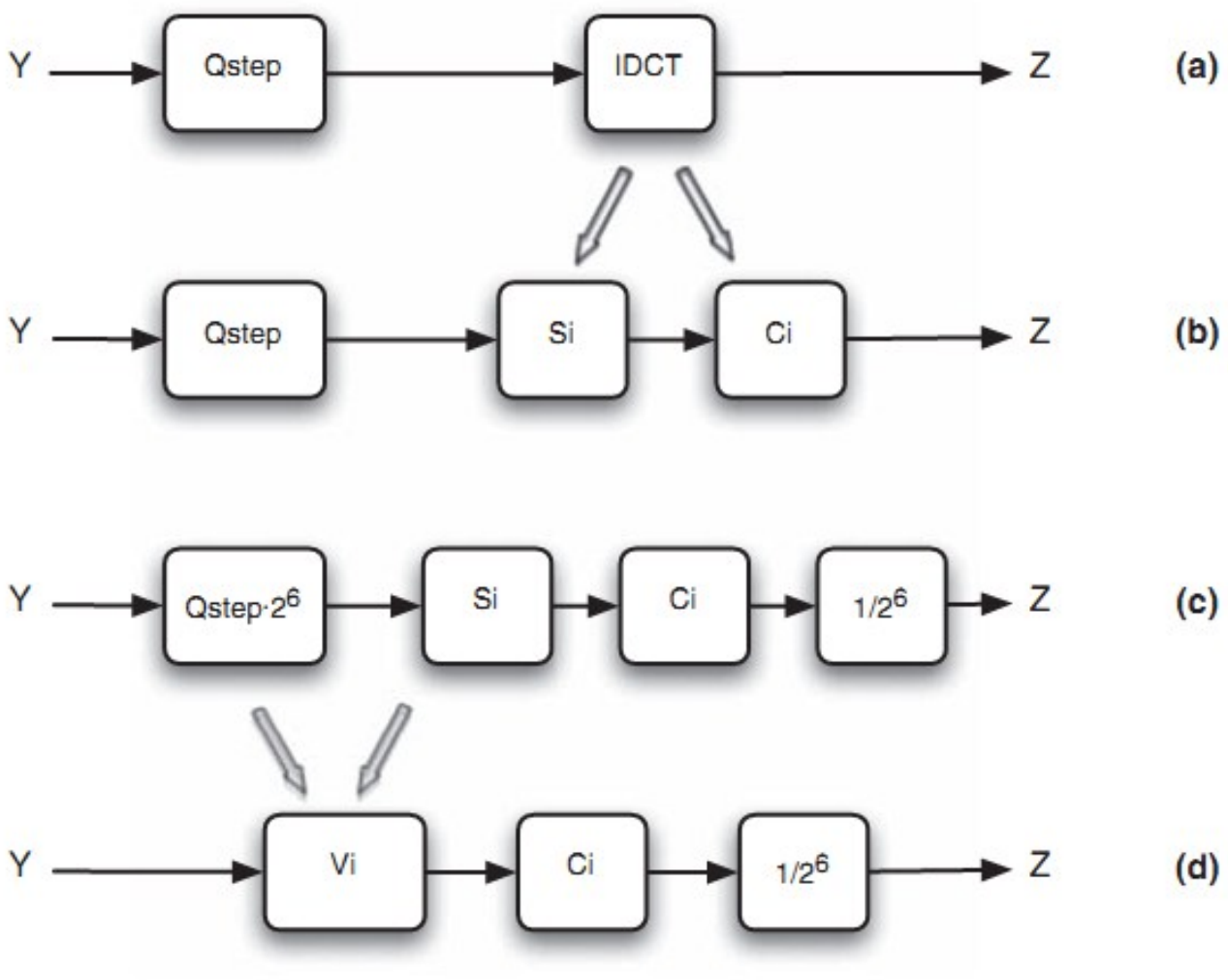


Figure 7.14 Development of the rescaling and inverse transform process

Quantization

Table 1 Quantization step sizes in H.264/AVC[2]

QP	0	1	2	3	4	5
QStep	0.625	0.6875	0.8125	0.875	1	1.125
QP	6	7	8	9	10	11
QStep	1.25	1.375	1.625	1.75	2	2.25
QP	...	...	...	...	...	...
QStep	...	...	...	...	...	...
QP	48	49	50	51		
QStep	160	181.333	202.666	224		

52 values of Qstep indexed by QP are defined in H.264

Qstep doubles for every increment of 6 in QP

Qstep increases by 12.5% for each increment of 1 in QP

4×4 Quantization and Scaling in H.264/AVC

- H.264 assumes a scalar quantization

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

Y_{ij} : Coefficient after integer DCT
Qstep : Quantization step
Z_{ij} : Quantized coefficient

$$Y_{ij} = W_{ij} \times PF$$

W_{ij}: unscaled coeffients
PF: Post-scaling Factor, which is depending on the position (i,j)

- The forward quantizaion formula:

$$\left|Z_{ij}\right| = \left(\left|W_{ij}\right| * \mathrm{LevelScale}_{m,i,j} + f\right) >> qbits$$

$$\mathrm{Sign}\left(Z_{ij}\right) = \mathrm{Sign}\left(W_{ij}\right) \qquad qbits = 15 + floor(QP/6)$$

$$\mathrm{LevelScale}_{m,i,j} = \mathrm{MF}_{m,S_{ij}} * 16 / \mathrm{Weight\_Scale}_{ij}$$

$$Z_{ij} = round(W_{ij} * MF_{m,S_{ij}} / 2^{qbits}),$$

$$S_{ij} = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 2 & 1 & 2 & 1 \\ 0 & 2 & 0 & 2 \\ 2 & 1 & 2 & 1 \end{bmatrix}, m = QP\%6, i, j = 0...3$$

$$MF = \begin{pmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{pmatrix}$$

Multiplication factor MF

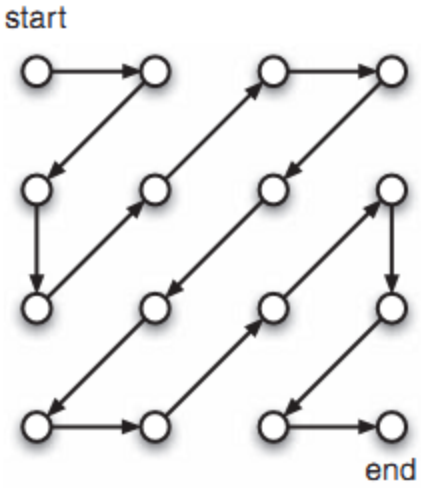
QP	Positions (0,0),(2,0),(2,2),(0,2)	Positions (1,1),(1,3),(3,1),(3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

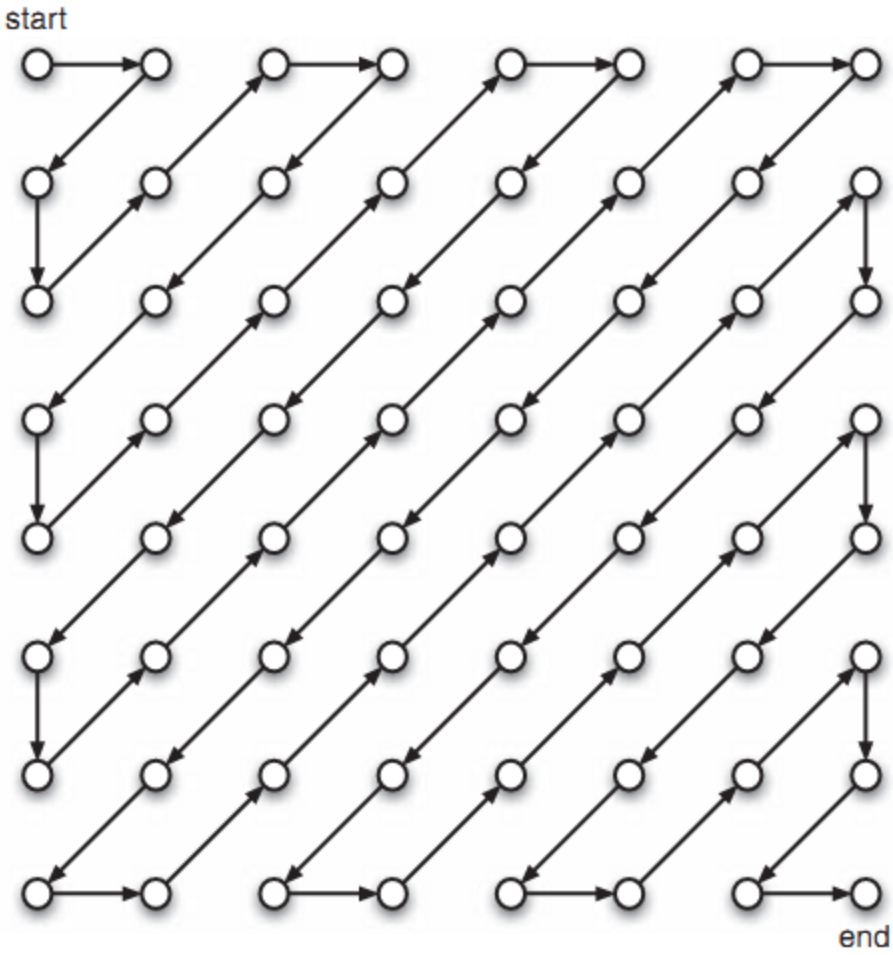
0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

For Example: QP =4, (i,j) = (0,0)

(PF = 0.25, Qstep = 1, PF/Qstep = 0.25)

Scan Order





Coding

Fixed length code: A symbol is converted into a binary code with a specified length (n bits).

Exponential-Golomb variable length code: The symbol is represented as an Exp-Golomb codeword with a varying number of bits (v bits). In general, shorter Exp-Golomb codewords are assigned to symbols that occur more frequently.

CAVLC: Context-Adaptive Variable Length Coding, a specially-designed method of coding transform coefficients in which different sets of variable-length codes are chosen depending on the statistics of recently-coded coefficients, using context adaptation.

CABAC: Context-Adaptive Binary Arithmetic Coding, a method of arithmetic coding in which the probability models are updated based on previous coding statistics.

> Symbols occurring in the syntax above the slice data level (Chapter 5) are coded using Fixed Length Codes or Exp-Golomb codes. Symbols at the slice data level and below are coded in one of two ways. If CABAC mode is selected, all of these symbols are coded using CABAC; otherwise, coefficient values are coded using CAVLC and other symbols are coded using fixed length or Exp-Golomb codes.

Exp-Golomb Coding

An Exp-Golomb codeword has the following structure:

```
[Zero prefix][1][INFO]
```

Table 7.9 Exp-Golomb Codewords

code_num	Codeword
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001
...	...

CAVLC
(Context Adaptive Variable Length Coding)

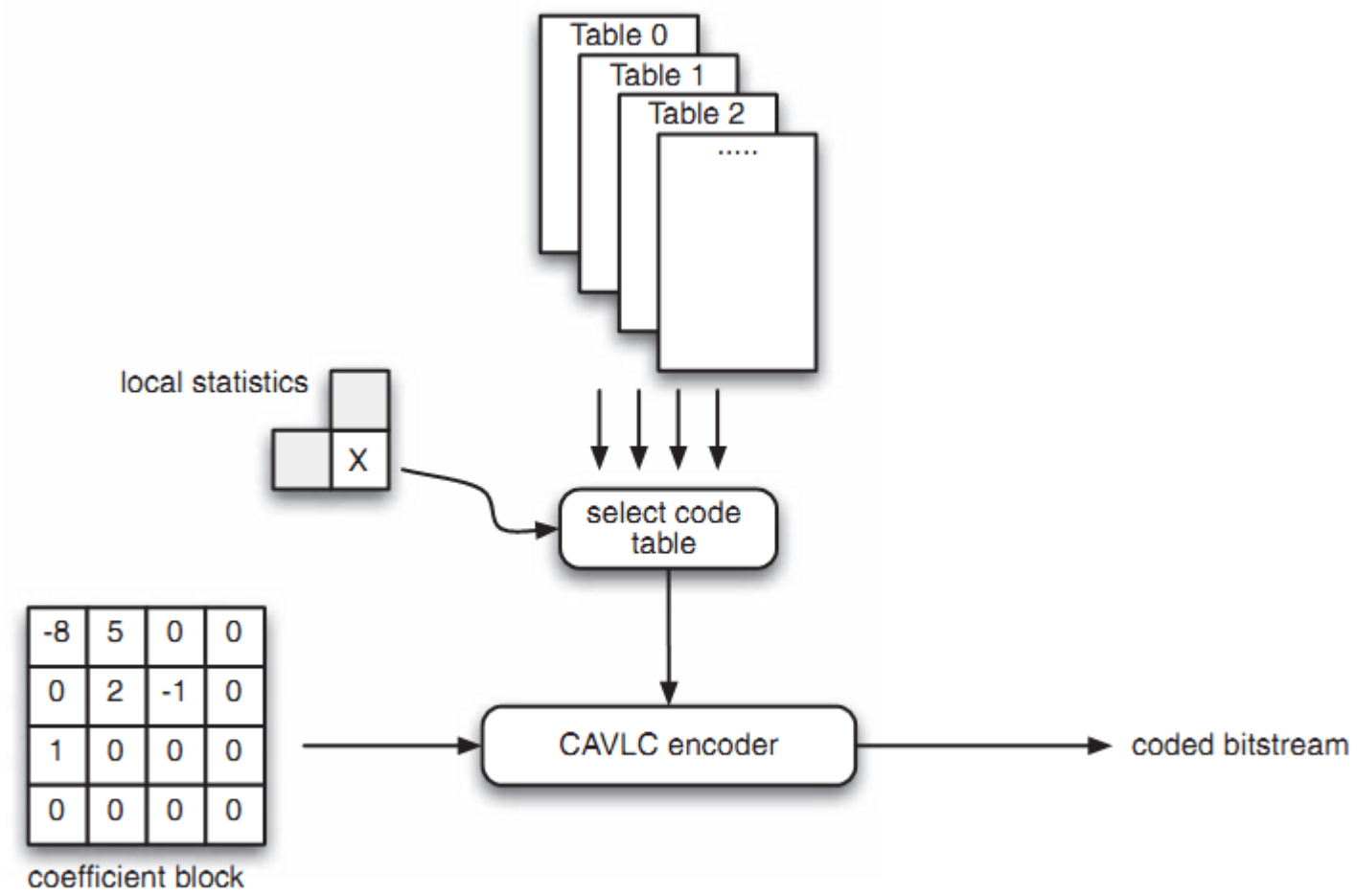


Figure 7.19 CAVLC encoder overview

CABAC
(Context Adaptive Binary Arithmetic Coding)

(a) selecting probability models for each syntax element according to the element's context,
(b) adapting probability estimates based on local statistics and
(c) using arithmetic coding rather than variable-length coding.

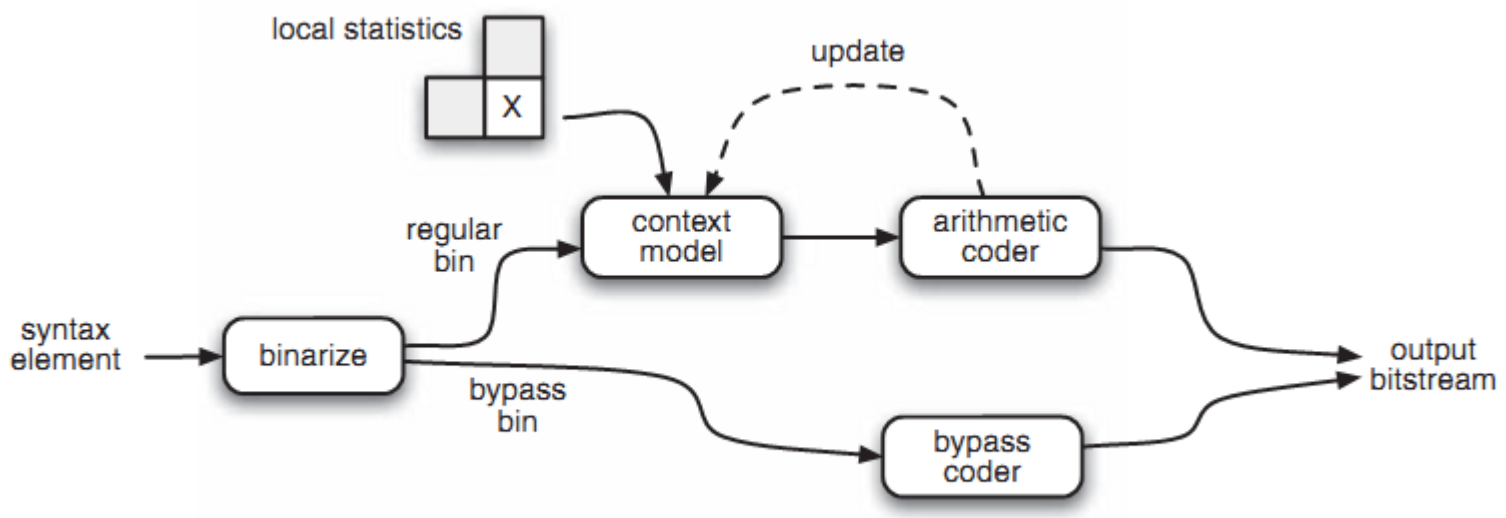


Figure 7.20 CABAC coding process overview

Video Codec VLSI Design

Project

Video Encoder Analysis Report

a) JM (Joint-Model)
b) x264