H.264 / MPEG-4 Part 10 White Paper

Transform and quantization

1. Introduction

The Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG are finalising a new standard for the coding (compression) of natural video images. The new standard [1] will be known as H.264 and also MPEG-4 Part 10, “Advanced Video Coding”. This document describes the transform and quantization processes defined, or implied, by the standard.

Each residual macroblock is transformed, quantized and coded. Previous standards such as MPEG-1, MPEG-2, MPEG-4 and H.263 made use of the 8x8 Discrete Cosine Transform (DCT) as the basic transform. The “baseline” profile of H.264 uses three transforms depending on the type of residual data that is to be coded: a transform for the 4x4 array of luma DC coefficients in intra macroblocks (predicted in 16x16 mode), a transform for the 2x2 array of chroma DC coefficients (in any macroblock) and a transform for all other 4x4 blocks in the residual data. If the optional “adaptive block size transform” mode is used, further transforms are chosen depending on the motion compensation block size (4x8, 8x4, 8x8, 16x8, etc).

Data within a macroblock are transmitted in the order shown in Figure 1-1. If the macroblock is coded in 16x16 Intra mode, then the block labelled “-1” is transmitted first, containing the DC coefficient of each 4x4 luma block. Next, the luma residual blocks 0-15 are transmitted in the order shown (with the DC coefficient set to zero in a 16x16 Intra macroblock). Blocks 16 and 17 contain a 2x2 array of DC coefficients from the Cb and Cr chroma components respectively. Finally, chroma residual blocks 18-25 (with zero DC coefficients) are sent.

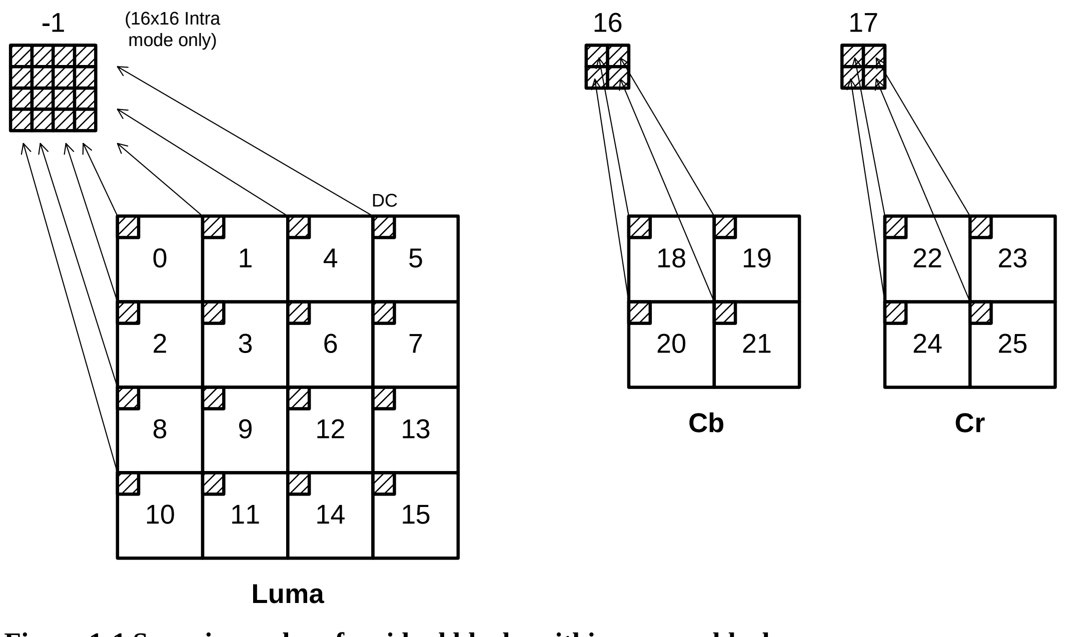

Figure 1-1 Scanning order of residual blocks within a macroblock

2. 4x4 residual transform and quantization (blocks 0-15, 18-25)

This transform operates on 4x4 blocks of residual data (labelled 0-15 and 18-25 in Figure 1-1) after motion-compensated prediction or Intra prediction. The transform is based on the DCT but with some fundamental differences:

1. It is an integer transform (all operations can be carried out with integer arithmetic, without loss of accuracy).

2. The inverse transform is fully specified in the H.264 standard and if this specification is followed correctly, mismatch between encoders and decoders should not occur.
3. The core part of the transform is multiply-free, i.e. it only requires additions and shifts.
4. A scaling multiplication (part of the complete transform) is integrated into the quantizer (reducing the total number of multiplications).

The entire process of transform and quantization can be carried out using 16-bit integer arithmetic and only a single multiply per coefficient, without any loss of accuracy.

2.1 Development from the 4x4 DCT

The 4x4 DCT of an input array **X** is given by:

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

Equation 2-1

where:

$$a = \frac{1}{2}$$
$$b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right)$$
$$c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$$

This matrix multiplication can be factorised [2] to the following equivalent form (Equation 2-2):

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

Equation 2-2

CXC$^{\mathbf{T}}$ is a "core" 2-D transform. **E** is a matrix of scaling factors and the symbol ⊗ indicates that each element of (**CXC$^{\mathbf{T}}$**) is multiplied by the scaling factor in the same position in matrix **E** (scalar multiplication rather than matrix multiplication). The constants *a* and *b* are as before; *d* is *c/b* (approximately 0.414).

To simplify the implementation of the transform, *d* is approximated by 0.5. To ensure that the transform remains orthogonal, *b* also needs to be modified so that:

$$a = \frac{1}{2}$$
$$b = \sqrt{\frac{2}{5}}$$
$$d = \frac{1}{2}$$

The 2nd and 4th rows of matrix **C** and the 2nd and 4th columns of matrix $\mathbf{C^T}$ are scaled by a factor of 2 and the post-scaling matrix **E** is scaled down to compensate. (This avoids multiplications by ½ in the "core" transform $\mathbf{CXC^T}$ which would result in loss of accuracy using integer arithmetic). The final forward transform becomes:

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

Equation 2-3

This transform is an approximation to the 4x4 DCT. Because of the change to factors *d* and *b*, the output of the new transform will not be identical to the 4x4 DCT.

The inverse transform (defined in [1]) is given by:

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

Equation 2-4

This time, **Y** is **pre-scaled** by multiplying each coefficient by the appropriate weighting factor from matrix $\mathbf{E_i}$. Note the factors +/-½ in the matrices **C** and $\mathbf{C^T}$; these can be implemented by a right-shift without a significant loss of accuracy because the coefficients **Y** are pre-scaled.

The forward and inverse transforms are orthogonal, i.e. $\mathbf{T^{-1}(T(X))} = \mathbf{X}$.

2.2 Quantization

H.264 uses a scalar quantizer . The definition and implementation are complicated by the requirements to (a) avoid division and/or floating point arithmetic and (b) incorporate the post- and pre-scaling matrices $\mathbf{E_f}$ and $\mathbf{E_i}$ described above.

The basic forward quantizer operation is as follows:

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

where Y_{ij} is a coefficient of the transform described above, Qstep is a quantizer step size and Z_{ij} is a quantized coefficient.

A total of 52 values of Qstep are supported by the standard and these are indexed by a Quantization Parameter, QP. The values of Qstep corresponding to each QP are shown in Table 2-1. Note that Qstep doubles in size for every increment of 6 in QP; Qstep increases by 12.5% for each increment of 1 in QP. The wide range of quantizer step sizes makes it possible for an encoder to accurately and flexibly control the trade-off between bit rate and quality. The values of QP may be different for luma and chroma; both parameters are in the range 0-51 but QP_{Chroma} is derived from QP_Y so that it QP_C is less that QP_Y for values of QP_Y above 30. A user-defined offset between QP_Y and QP_C may be signalled in a Picture Parameter Set.

Table 2-1 Quantization step sizes in H.264 CODEC

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

The post-scaling factor a^2, ab/2 or $b^2/4$ (Equation 2-3) is incorporated into the forward quantizer. First, the input block **X** is transformed to give a block of unscaled coefficients $\mathbf{W = CXC^T}$. Then, each coefficient W_{ij} is quantized and scaled in a single operation:

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

Equation 2-5

PF is a^2, ab/2 or $b^2/4$ depending on the position (i,j) (see Equation 2-3):

Position	PF
(0,0), (2,0), (0,2) or (2,2)	a^2
(1,1), (1,3), (3,1) or (3,3)	$b^2/4$
Other	ab/2

The factor (PF/Qstep) is implemented in the H.264 reference model software [3] as a multiplication by MF (a multiplication factor) and a right-shift, thus avoiding any division operations:

$$Z_{ij} = \text{round}\left(W_{ij} \cdot \frac{MF}{2^{qbits}}\right)$$

Equation 2-6

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

In integer arithmetic, Equation 2-6 can be implemented as follows:

$|Z_{ij}| = (|W_{ij}|.MF + f) >> qbits$
$sign(Z_{ij}) = sign(W_{ij})$

Equation 2-7

where >> indicates a binary shift right. In the reference model software, f is $2^{qbits}/3$ for Intra blocks or $2^{qbits}/6$ for Inter blocks.

Example:

QP = 4, hence Qstep = 1.0
(i,j) = (0,0), hence PF = a^2 = 0.25
qbits = 15, hence 2^{qbits} = 32768

$\frac{MF}{2^{qbits}} = \frac{PF}{Qstep}$, hence MF = (32768x0.25)/1 = **8192**

The first 6 values of MF can be calculated as follows, depending on QP and the coefficient position (i,j):

Table 2-2 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

The 2nd and 3rd columns of this table (positions with factors $b^2/4$ and $ab/2$) have been modified slightly[1] from the results of Equation 2-6.

For QP>5, the factors MF remain unchanged but the divisor 2^{qbits} increases by a factor of 2 for each increment of 6 in QP. For example, qbits=16 for $6 \leq QP \leq 11$; qbits=17 for $12 \leq QP \leq 17$; and so on.

2.3 Rescaling

The basic rescale (or "inverse quantizer") operation is:

$Y'_{ij} = Z_{ij}.Qstep$

Equation 2-8

The pre-scaling factor for the inverse transform (matrix $\mathbf{E_i}$, containing values a^2, ab and b^2 depending on the coefficient position) is incorporated in this operation, together with a further constant scaling factor of 64 to avoid rounding errors:

$W'_{ij} = Z_{ij}.Qstep.PF.64$

Equation 2-9

W'_{ij} is a scaled coefficient which is then transformed by the "core" inverse transform ($\mathbf{C_i^T W C_i}$: see Equation 2-4). The values at the output of the inverse transform are divided by 64 to remove the scaling factor (this can be implemented using only an addition and a right-shift).

[1] It is acceptable to modify a forward quantizer in order to improve perceptual quality at the decoder, since only the rescaling (inverse quantizer) process is standardised.

The H.264 standard does not specify Qstep or PF directly. Instead, the parameter V=(Qstep.PF.64) are defined for 0≤QP≤5 and each coefficient position and the rescaling operation is:

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

Equation 2-10

Example:

QP=3, hence Qstep = 0.875 and $2^{floor(QP/6)} = 1$
(i,j)=(1,2), hence PF = ab = 0.3162
V=(Qstep.PF.64)= 0.875x0.3162x65 ≅ 18
$W'_{ij} = Z_{ij}$x18x1

The values of V for 0≤QP≤5 are defined in the standard as follows:

Table 2-3 Rescaling factor V

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

The factor $2^{floor(QP/6)}$ in Equation 2-10 makes the rescaled output increase by a factor of 2 for every increment of 6 in QP.

3. 4x4 luma DC coefficient transform and quantization (16x16 Intra-mode only)

If the macroblock is encoded in 16x16 Intra prediction mode (where the entire 16x16 luminance component is predicted from neighbouring pixels), each 4x4 residual block is first transformed using the "core" transform described above ($\mathbf{C_f X C_f^T}$). The DC coefficient of each 4x4 block is then transformed again using a 4x4 Hadamard transform:

$$\mathbf{Y}_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \left[\mathbf{W}_D \right] \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2$$

Equation 3-1

$\mathbf{W_D}$ is the block of 4x4 DC coefficients and $\mathbf{Y_D}$ is the block after transformation. The output coefficients $Y_{D(i,j)}$ are divided by 2 (with rounding).

The output coefficients $Y_{D(i,j)}$ are then quantized to produce a block of quantized DC coefficients:

$|Z_{D(i,j)}| = (|Y_{D(i,j)}|.MF_{(0,0)} + 2f) >> (qbits+1)$
$sign(Z_{D(i,j)}) = sign(Y_{D(i,j)})$

Equation 3-2

where MF, f and qbits are defined as before and MF depends on the position (i,j) within the 4x4 DC coefficient block as before.

At the decoder, an inverse Hadamard transform is applied **followed by** rescaling (note that the order is not reversed as might be expected):

$$\mathbf{W}_{QD} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} & \mathbf{Z}_D & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right)$$

If QP is greater than or equal to 12, rescaling is performed by:

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

If QP is less than 12, rescaling is performed by:

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

V is defined as before. The rescaled DC coefficients $\mathbf{W'_D}$ are then inserted into their respective 4x4 blocks and each 4x4 block of coefficients is inverse transformed using the core DCT-based inverse transform ($\mathbf{C_i^T W' C_i}$).

In an intra-coded macroblock, much of the energy is concentrated in the DC coefficients and this extra transform helps to de-correlate the 4x4 luma DC coefficients (i.e. to take advantage of the correlation between the coefficients).

4. 2x2 chroma DC coefficient transform and quantization

Each chroma component in a macroblock is made up of four 4x4 blocks of samples. Each 4x4 block is transformed as described in section 2. The DC coefficients of each 4x4 block of coefficients are grouped in a 2x2 block ($\mathbf{W_D}$) and are further transformed prior to quantization:

$$\mathbf{Y}_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{W}_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Equation 4-1

Quantization of the 2x2 output block $\mathbf{Y_D}$ is performed by:

$|Z_{D(i,j)}| = (|Y_{D(i,j)}|.MF_{(0,0)} + 2f) >> (qbits+1)$
$sign(Z_{D(i,j)}) = sign(Y_{D(i,j)})$

Equation 4-2

where MF, f and qbits are defined as before.

During decoding, the inverse transform is applied before rescaling:

$$\mathbf{W}_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{Z}_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Equation 4-3

If QP is greater than or equal to 6, rescaling is performed by:

$W'_{D(i,j)} = W_{QD(i,j)}.V_{(0.0)}.2^{floor(QP/6)-1}$

If QP is less than 6, rescaling is performed by:

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

The rescaled coefficients are replaced in their respective 4x4 blocks of chroma coefficients which are then transformed as above ($\mathbf{C_i^T W' C_i}$). As with the Intra luma DC coefficients, the extra transform helps to de-correlate the 2x2 chroma DC coefficients and hence improves compression performance.

5. The complete transform, quantization, rescaling and inverse transform process

The complete process from input residual block **X** to output residual block **X'** is described below and illustrated in Figure 5-1.

Encoding:

1. Input: 4x4 residual samples: $\mathbf{X}$

2. Forward "core" transform: $\mathbf{W = C_f X C_f^T}$
(followed by forward transform for Chroma DC or Intra-16 Luma DC coefficients)

3. Post-scaling and quantization: $\mathbf{Z = W}.\dfrac{PF}{Qstep.2^{qbits}}$

(modified for Chroma DC or Intra-16 Luma DC)

Decoding:

(inverse transform for Chroma DC or Intra-16 Luma DC coefficients)
4. Re-scaling (incorporating inverse transform pre-scaling):
$\mathbf{W'} = \mathbf{Z}.Qstep.PF.64$
(modified for Chroma DC or Intra-16 Luma DC)

5. Inverse "core" transform: $\mathbf{X' = C_i^T W' C_i}$

6. Post-scaling: $\mathbf{X''} = round(X'/64)$

7. Output: 4x4 residual samples: $\mathbf{X''}$

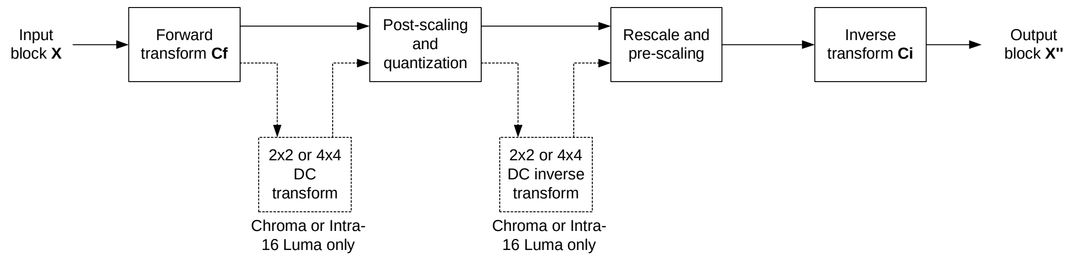
Input block X
Forward transform Cf
Post-scaling and quantization
Rescale and pre-scaling
Inverse transform Ci
Output block X''
2x2 or 4x4 DC transform
Chroma or Intra-16 Luma only
2x2 or 4x4 DC inverse transform
Chroma or Intra-16 Luma only

Figure 5-1 Transform, quantization, rescale and inverse transform flow diagram

Example (luma 4x4 residual block, Inter mode):

QP = 10

Input block **X:**

	j=0	1	2	3
i=0	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

Output of “core” transform **W:**

	j=0	1	2	3
i=0	140	-1	-6	7
1	-19	-39	7	-92
2	22	17	8	31
3	-27	-32	-59	-21

MF = 8192, 3355 or 5243 (depending on the coefficient position) and qbits=16. Output of forward quantizer **Z**:

	j=0	1	2	3
i=0	17	0	-1	0
1	-1	-2	0	-5
2	3	1	1	2
3	-2	-1	-5	-1

V = 16, 25 or 20 (depending on position) and $2^{\mathrm{floor}(QP/6)} = 2^1 = 2$. Output of rescale **W’:**

	j=0	1	2	3
i=0	544	0	-32	0
1	-40	-100	0	-250
2	96	40	32	80
3	-80	-50	-200	-50

Output of “core” inverse transform **X’’** (after division by 64 and rounding)**:**

	j=0	1	2	3
i=0	4	13	8	10
1	8	8	4	12
2	1	10	10	3
3	18	5	14	7

6. References

1 ITU-T Rec. H.264 / ISO/IEC 11496-10, “Advanced Video Coding”, Final Committee Draft, Document JVT-F100, December 2002
2 A. Hallapuro and M. Karczewicz, “Low complexity transform and quantization – Part 1: Basic Implementation”, JVT document JVT-B038, February 2001
3 JVT Reference Software version 4.0, ftp://ftp.imtc-files.org/jvt-experts/reference_software/