# Decoding process

## Scaling, transformation and array construction process prior to deblocking filter process

### Derivation process for quantization parameters

Inputs of this process are:

– luma location ( xB, yB ) specifying the top-left luma sample of the current coding unit quantization group relative to the top‑left luma sample of the current picture,

Outputs of this process are:

– luma quantization parameter QP’Y,

– chroma quantization parameters QP’Cb and QP’Cr

Let a variable QPY\_A specifying the luma quantization parameter of the coding unit quantization group covering ( xB−1,  yB ), a variable tbAddrA specifying the treeblock address of treeblock containing the coding unit quantization group covering ( xB−1,  yB ), a variable QPY\_B specifying the luma quantization parameter of the coding unit quantization group covering ( xB,  yB−1 ), a variable , tbAddrB specifying the treeblock address of treeblock containing the coding unit quantization covering ( xB,  yB−1 ), and a variable QPY\_PREV specifying the luma quantization parameter of the previous coding unit quantization group in decoding order, respectively.

When at least one of the following conditions is true, QPY\_PREV is initially set equal to SliceQPY at the start of each slice for the first coding unit quantization group in the slice.

– A quantization group of coding units is the first quantization group of coding units in a slice.

– A quantization group of coding units is the first quantization group of coding units in a tile.

– A quantization group of coding units is the first quantization group of coding units in a treeblock row and tiles\_or\_entry\_coding\_sync\_idc is equal to 2.

The prediction of luma quantization parameter QPY\_PRED is derived as the following ordered steps:

1. If the coding unit quantization group covering ( xB−1,  yB ) in the current slice is not available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] or tbAddrA is not equal to CurrTbAddr, then QPY\_A is set equal to QPY\_PREV.
2. If the coding unit quantization group covering ( xB,  yB−1 ) in the current slice is not available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] or tbAddrB is not equal to CurrTbAddr, then QPY\_B is set equal to QPY\_PREV.
3. The prediction of luma quantization paramaeter QPY\_PRED is derived as:

 QPY\_PRED =  (QPY\_A + QPY\_B + 1) >> 1 (8‑252)

The value of QPY is derived as

QPY = ( ( ( QPY\_PRED + cu\_qp\_delta +52+ 2\*QpBdOffsetY )%( 52 + QpBdOffsetY ) ) − QpBdOffsetY (8‑253)

The value of luma quantization parameter QP’Y is derived as

QP’Y = QPY + QpBdOffsetY (8‑254)

The values of QPCb and QPCr are equal to the value of QPC as specified in Table 8‑11 based on the index qPI equal to qPICb and qPICr derived as:

qPICb = Clip3( −QpBdOffsetC, 51, QPY + cb\_qp\_offset) (8‑255)

qPICr = Clip3( −QpBdOffsetC, 51, QPY + cr\_qp\_offset) (8‑256)

The values of chroma quantization parameters for Cb and Cr components, QP’Cb and QP’Cr are derived as:

QP’Cb = QPCb + QpBdOffsetC (8‑257)

QP’Cr = QPCr + QpBdOffsetC (8‑258)

**Table 8‑11 – Specification of QPC as a function of qPI**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| qPI | <30 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 |
| QPC | = qPI | 29 | 30 | 31 | 32 | 32 | 33 | 34 | 34 | 35 | 35 | 36 | 36 | 37 | 37 | 37 | 38 | 38 | 38 | 39 | 39 | 39 | 39 |

### Scaling process for transform coefficients

Inputs of this process are:

– a variable nW specifying the width of the current transform unit,

– a variable nH specifying the height of the current transform unit,

– a (nW)x(nH) array c of transform coefficients with elements cij,

– a variable cIdx specifying the chroma component of the current block,

– a variable qP specifying the quantization parameter.

Output of this process is scaled transform coefficients as a (nW)x(nH) array of d with elements dij.

The variable log2TrSize is derived as follows:

log2TrSize = ( Log2( NW ) + Log2( NH ) ) >> 1 (8‑260)

The variable shift is derived as follows:

– If cIdx is equal to 0,

shift = BitDepthY + log2TrSize − 9 (8‑261)

levelLimit = 1 << Min( 15, 12 + BitDepthY + log2TrSize – qP / 6 ) (8‑262)

– Otherwise,

shift = BitDepthC + log2TrSize − 9 (8‑263)

levelLimit = 1 << Min( 15, 12 + BitDepthC + log2TrSize – qP / 6 ) (8‑264)

The scaling array levelScale[·] is specified as levelScale[k] = { 40, 45, 51, 57, 64, 72 } with k=0..5.

The variable shiftScale is set equal to shift + 4 – qP/6.

The elements of array M[ i ][ j ] with i = 0..nW − 1, j = 0..nH − 1 are set equal to ScalingFactor[ SizeID ][ RefMatrixID ][ trafoType ][ i\*nW+j ], where SizeID and RefMatrixID are specified in Table 7‑2 and Equation 7‑25, respectively, and trafoType is derived by

trafoType = ( ( nW = = nH ) ? 0 : ( ( nW > nH ) ? 1 : 2 ) ) (8‑265)

The scaled transform coefficient dij with i = 0..nW − 1, j = 0..nH − 1 is derived as follows.

– If scaling\_list\_present\_flag is equal to 0, [Ed. (WJ): do we need scaling\_list\_disable\_flag?]

yij = Clip3( −32768, 32767, cij ) (8‑266)

dij = Clip3( −32768, 32767,( ( yij \* levelScale[ qP%6 ] << ( qP/6 ) ) + ( 1 << ( shift − 1 ) ) ) >> shift ) (8‑267)

– Otherwise, if shiftScale is greater than 0,

yij = Clip3( −32768, 32767, cij ) (8‑268)

dij = Clip3( −32768, 32767, ( yij \* M[ i ][ j ]\*levelScale[ qP%6 ] +  ( 1 << ( shiftScale − 1 ) ) ) >> shiftScale ) (8‑269)

– Otherwise,

yij = Clip3( -LevelLimit, LevelLimit − 1, cij ) (8‑270)

dij = Clip3( −32768, 32767, ( yij \* M[ i ][ j ]\*levelScale[ qP%6 ] ) << ( -shiftScale ) ) (8‑271)

## In-loop filter process

### Deblocking filter process

#### Filtering process for coding unit

##### Decision process for luma block edge

Inputs of this process are:

– a luma location ( xC, yC ) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a luma location ( xB, yB ) specifying the top-left luma sample of the current block relative to the top left luma sample of the current coding unit,

– a variable verticalEdgeFlag,

– a variable bS specifying the boundary filtering strength,

Output of this process is:

– variables dE, dEp, dEq containing decisions,

This process is invoked only when bS is not equal to 0.

If verticalEdgeFlag is equal to 1, the sample values pi,k and qi,k with i = 0..3 and k = 0, 3 are derived as follows:

 qi,k = recPictureL[ xC + xB +i, yC + yB + k ] (8‑292)

 pi,k = recPictureL[ xC + xB – i − 1, yC + yB + k ] (8‑293)

Otherwise (verticalEdgeFlag is equal to 0), the sample values pi,k and qi,k with i = 0..3 and k = 0, 3 are derived as follows:

 qi,k = recPictureL[ xC + xB +k, yC + yB + i ] (8‑302)

 pi,k = recPictureL[ xC + xB +k, yC + yB – i − 1 ] (8‑303)

The variables QPQ and QPP are set equal to the QPY values of the coding units containing the sample q0,0 and p0,0, respectively.

If pcm\_enabled\_flag is equal to 1, the variables QPQ and QPP are modified as follows:

– When pcm\_flag of the coding unit containing the sample q0,0 is equal to 1, QPQ is set equal to 0;

* When pcm\_flag of the coding unit containing the sample p0,0 is equal to 1, QPP is set equal to 0.

A variable qPL is derived as follows:

 qPL = ( ( QPY + QPP + 1 ) >> 1 ) (8‑291)

A variables β is specified as Table 8‑12 with luma quantization parameter Clip3( 0, 51, qPL + ( beta\_offset\_div2 << 1 ) ) as input.

A variable tC is specified as Table 8‑12 with luma quantization parameter Clip3( 0, 55, qPL + 2\*( bS − 1) + ( tc\_offset\_div2 << 1) ) as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, the following ordered steps apply:

1. The variables dpq0, dpq3, dp, dq are derived as follows:

 dp0 = | p2,0 − 2\*p1,0 + p0,0 |

 dp3 =| p2,3 − 2\*p1,3 + p0,3 |  (8‑294)

 dq0 = | q2,0 − 2\*q1,0 + q0,0 |  (8‑295)

 dq0 = | q2,3 − 2\*q1,3 + q0,3 | (8‑296)

 dpq0 = dp0 + dq0 (8‑297)

 dpq3 = dp3 + dq3 (8‑298)

 dp = dp0 + dp3 (8‑299)

 dq = dq0 + dq3 (8‑300)

1. The variable d is derived as follows:

 d = dpq0 + dpq3 (8‑301)

1. The variables dE, dEp and dEq are set equal to 0.
2. if d is less than β, the following ordered steps apply:
3. For the sample location ( xC + xB, yC + yB ), the decision process for a luma sample specified in subclause 8.7.1.4.4 is invoked with sample values pi,0, qi,0, the variables 2\*dpq0, β and tC as inputs and a decision dSam0 as output.
4. For the sample location ( xC + xB, yC + yB + 3), the decision process for a luma sample specified in subclause 8.7.1.4.4 is invoked with sample values pi,3, qi,3, the variables 2\*dpq3, β and tC as inputs and a decision dSam3 as output.
5. The variable dE is set equal to 1.
6. If dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.
7. If dp is less than ( β + ( β >> 1 ) ) >> 3, the variable dEp is set equal to 1.
8. If dq is less than ( β + ( β >> 1 ) ) >> 3, the variable dEq is set equal to 1.

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. The variables dpq0, dpq3, dp, dq are derived as follows:

 dp0 = | p2,0 − 2\*p1,0 + p0,0 |  (8‑304)

 dp3 =| p2,3 − 2\*p1,3 + p0,3 |  (8‑305)

 dq0 = | q2,0 − 2\*q1,0 + q0,0 |  (8‑306)

 dq0 = | q2,3 − 2\*q1,3 + q0,3 | (8‑307)

 dpq0 = dp0 + dq0 (8‑308)

 dpq3 = dp3 + dq3 (8‑309)

 dp = dp0 + dp3 (8‑310)

 dq = dq0 + dq3 (8‑311)

1. The variables d is derived as follows:

 d = dpq0 + dpq3 (8‑312)

1. The variables dE, dEp and dEq are set equal to 0.
2. if d is less than β, the following ordered steps apply:
3. For the sample location ( xC + xB, yC + yB ), the decision process for a luma sample specified in subclause 8.7.1.4.4 is invoked with sample values pi,0, qi,0 the variables 2\*dpq0, β and tC as inputs and a decision dSam0 as output.
4. For the sample location ( xC + Xb + 3, yC + yB ), the decision process for a luma sample specified in subclause 8.7.1.4.4 is invoked with sample values pi,3, qi,3, the variables 2\*dpq3, β and tC as inputs and a decision dSam3 as output.
5. The variable dE is set equal to 1.
6. If dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.
7. If dp is less than ( β + ( β >> 1 ) ) >> 3, the variable dEp is set equal to 1.
8. If dq is less than ( β + ( β >> 1 ) ) >> 3, the variable dEq is set equal to 1.

##### Filtering process for luma block edge

Inputs of this process are:

– a luma location ( xC, yC ) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a luma location ( xB, yB ) specifying the top-left luma sample of the current block relative to the top left luma sample of the current coding unit,

– a variable verticalEdgeFlag,

– a variable bS specifying the boundary filtering strength,

– variables dE, dEp, dEq containing decisions,

– a variable bSL,

– a variable tCL,

Output of this process is:

– modified reconstruction of the picture.

This process is invoked only when bS is not equal to 0.

The variable β is specified in Table 8‑12 with luma quantization parameter Q equal to Clip3( 0, 51, qPL + ( beta\_offset\_div2 << 1 ) ).

A variable tC is specified as Table 8‑12 with luma quantization parameter Clip3( 0, 53, qPL + 2\*(bS − 1) + ( tc\_offset\_div2 << 1 ) ) as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, the following ordered steps apply:

1. The sample values pi,k and qi,k with i = 0..3 and k = 0..3 are derived as follows:

 qi,k = recPictureL[ xC + xB +i, yC + yB + k ] (8‑313)

 pi,k = recPictureL[ xC + xB – i − 1, yC + yB + k ] (8‑314)

1. If dE is not equal to 0, for each sample location ( xC + xB, yC + yB + k ), k = 0..3, the following ordered steps apply:
2. The filtering process for a luma sample specified in subclause 8.7.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, the decision dE, variables dEp and dEq, the variable tC as inputs and the number of filtered samples nDp and nDq from each side of the block boundary, and the filtered sample values pi’ and qj’ as outputs.
3. The filtered sample values pi’ and qj’ with i = 0..nDp − 1, j = 0..nDq − 1 replace the corresponding samples inside the sample array recPictureL as follows:

 recPictureL[ xC + xB +j, yC + yB + k ] = qj’ (8‑315)

 recPictureL[ xC + xB – i − 1, yC + yB + k ] = pi’ (8‑316)

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. If xPOS is equal to 1, the parameters ks and ke are set to −4 and 3 respectively. If xPOS is equal to 2, the parameters ks and ke are set to 0 and 4 respectively. Otherwise ks and ke are set to 0 and 3 respectively.
2. The sample values pi,k and qi,k with i = 0..3 and k = ks..ke are derived as follows:

 qi,k = recPictureL[ xC + xB +k, yC + yB + i ] (8‑317)

 pi,k = recPictureL[ xC + xB +k, yC + yB – i − 1 ] (8‑318)

1. If xPOS is less than 0 [Ed. (WJ): maybe wrong condition] and dEL is not equal to 0, for each sample location ( xC + xB + k, yC + yB ), k = −3..−1, the following ordered steps apply:
2. The filtering process for a luma sample specified in subclause 8.7.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, decision variables dEL, dEpL and dEqL [Ed. (WJ): dEpL and dEqL do not exist. Needs to be stored], the variable tCL as inputs and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values pi’ and qj’ as outputs.
3. The filtered sample values pi’ and qj’ with i = 0..nDp − 1, j = 0..nDq − 1 replace the corresponding samples inside the sample array recPictureL as follows:

 recPictureL[ xC + xB +k, yC + yB + j ] = qj’ (8‑319)

 recPictureL[ xC + xB +k, yC + yB – i − 1 ] = pi’ (8‑320)

1. If dE is not equal to 0, for each sample location ( xC + xB + k, yC + yB ), k = 0.. ke−4 , the following ordered steps apply:
2. The filtering process for a luma sample specified in subclause 8.7.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, decision dE, variables dEp and dEq, the variable tC as inputs and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values pi’ and qj’ as outputs.
3. The filtered sample values pi’ and qj’ with i = 0..nDp − 1, j = 0..nDq − 1 replace the corresponding samples inside the sample array recPictureL as follows:

 recPictureL[ xC + xB + k, yC + yB + j ] = qj’ (8‑321)

 recPictureL[ xC + xB + k, yC + yB – i − 1 ] = pi’ (8‑322)

##### Filtering process for chroma block edge

[Ed.: (WJ) cIdx cannot be 0 here]

Inputs of this process are:

– a luma location ( xC, yC ) specifying the top-left chroma sample of the current coding unit relative to the top left chroma sample of the current picture,

– a luma location ( xB, yB ) specifying the top-left chroma sample of the current block relative to the top left chroma sample of the current coding unit,

– a variable verticalEdgeFlag,

– a variable bS specifying the boundary filtering strength,

– a variable cIdx specifying the chroma component index.

– a variable xPOS,

– a variable bSL,

– a variable tCL

Output of this process is:

– modified reconstruction of the picture.

This process is invoked only when bS is greater than 1.

Let s’ be a variable specifying chroma sample array which is derived as follows.

– If cIdx is equal to 1, s’ represents the chroma sample array recPictureCb of the current picture.

– Otherwise (cIdx is equal to 2), s’ represents the chroma sample array recPictureCr of the current picture.

If verticalEdgeFlag is equal to 1, the values pi and qi with i = 0..1 and k = 0..3 are derived as follows:

 qi,k = s’[ xC + xB +i, yC + yB + k ] (8‑324)

 pi,k = s’[ xC + xB – i − 1, yC + yB + k ] (8‑325)

Otherwise (verticalEdgeFlag is equal to 0), the following applies.

1. The parameters ks and ke are derived as follows.

– If xPOS is equal to 1, the parameters ks and ke are set to −1 and 2 respectively.

– Otherwise if xPOS is equal to 2, the parameters ks and ke are set to 0 and 2 respectively.

– Otherwise, ks and ke are set to 0 and 3 respectively.

1. The sample values pi and qi with i = 0..1 and k = ks..ke are derived as follows:

 qi,k = s’[ xC + xB +k, yC + yB + i ] (8‑328)

 pi,k = s’[ xC + xB +k, yC + yB – i − 1 ] (8‑329)

The variables QPQ and QPP are set equal to the QPY values of the coding units containing the sample q0,0 and p0,0, respectively.

If pcm\_enabled\_flag is equal to 1, the variables QPQ and QPP are modified as follows:

– When pcm\_flag of the coding unit containing the sample q0,0 is equal to 1, QPQ is set equal to 0;

* When pcm\_flag of the coding unit containing the sample p0,0 is equal to 1, QPP is set equal to 0.

The variable QPC is specified in Table 8‑11 based on an index qPI derived as follows:

 qPI = ( ( QPY + QPP + 1 ) >> 1 ) (8‑323)

A variable tC is specified as Table 8‑12 with luma quantization parameter Clip3( 0, 53, QPC + 2\*(bS − 1) + (tc\_offset\_div2 << 1 ) ) as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, for each sample location ( xC + xB, yC + yB + k ), k = 0..3, the following ordered steps apply:

1. The filtering process for a sample specified in subclause 8.7.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bS and the variable tC as inputs and the filtered sample values p0’ and q0’ as outputs.
2. The filtered sample values p0’ and q0’ replace the corresponding samples inside the sample array s’ as follows:

 s’[ xC + xB , yC + yB + k ] = q0’ (8‑326)

 s’[ xC + xB − 1, yC + yB + k ] = p0’ (8‑327)

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. If xPOS is less than 0, and if bSL, is greater than 1, for each sample location ( xC + xB − 1, yC + yB ), k = 0.. ke, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.7.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bSL and the variable tCL as inputs and the filtered sample values p0’ and q0’ as outputs.
3. The filtered sample values p0’ and q0’ replace the corresponding samples inside the sample array s’ as follows:

 s’[ xC + xB +k, yC + yB ] = q0’ (8‑330)

 s’[ xC + xB +k, yC + yB – i − 1 ] = p’ (8‑331)

1. If bS is greater than 1, for each sample location ( xC + xB + k, yC + yB ), k = 0.. ke, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.7.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bS and the variable tC as inputs and the filtered sample values p0’ and q0’ as outputs.
3. The filtered sample values p0’ and q0’ replace the corresponding samples inside the sample array s’ as follows:

 s’[ xC + xB +k, yC + yB ] = q0’ (8‑332)

 s’[ xC + xB +k, yC + yB − 1 ] = p0’ (8‑333)