### Deblocking filter process

…

The deblocking filter process shall be applied to all prediction unit edges and transform unit edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by disable\_deblocking\_filter\_idc.

For luma component, for the transform units and prediction units with edges smaller than 8 samples in either vertical or horizontal direction, only the edges lying on the 8x8 sample grid are filtered. If the luma component of one edge is filtered by luma filtering, the corresponding chroma component of this edge is also filtered by chroma filtering.

If disable\_deblocking\_filter\_idc is equal to 1, the deblocking filter process is simply omitted otherwise the following ordered steps apply for each coding unit with the same order as decoding process.

#### 8.6.1.3 Derivation process of boundary filtering strength

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 variable log2CUSize specifying the size of the current coding unit,

– a two-dimensional arrays of size (nS)x(nS), horEdgeFlags and verEdgeFlags.

Output of this process are three arrays of size (2)x(nS)x(nS), bS, bSCb, bSCr specifying the boundary filtering strength for luma, and two chroma components respectively.

Let ( xEk, yEj ) with k = 0..nE-1 and j = 0..nE-1 specify a set of edge sample locations where nE is set equal to ( ( 1 << log2CUSize ) >> 2 ), xE0 = 0, yE0 = 0, xEk+1 = xEk + 4 and yEj+1 = yEj + 4.

For ( xEk, yEj ) with k = 0..nE-1 and j = 0..nE-1, the following applies.

* If horEdgeFlags[ xEk ][ yEj ] is equal to 1,
* Set sample p0 = recPicture[ xC + xEk ][ yC + yEj – 1 ] and q0 = recPicture[ xC + xEk ][ yC + yEj ].
* The variable filterDir is set equal to 1.
* Otherwise, if verEdgeFlags[ xEk ][ yEj ] is equal to 1,
* Set sample p0 = recPicture[ xC + xEk – 1 ][ yC + yEj ] and q0 = recPicture[ xC + xEk ][ yC + yEj ].
* The variable filterDir is set equal to 0.
* Depending on the value of filterDir, the variable bS[ filterDir ][ xEk ][ yEj ] is derived as follows.
* If the block edge is also a coding unit edge and the following condition is true, the variable bS[ 0 ][ xEk ][ yEj ] is set equal to 4.
* The sample p0 or q0 is in a coding unit coded with intra prediction mode
* Otherwise, if the following condition is true, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 3.
* The sample p0 or q0 is in a coding unit coded with intra prediction mode
* Otherwise, if the block edge is also a transform unit edge and the following condition is true, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 2.
* The sample p0 or q0 is in a transform unit which contains non-zero transform coefficient level.
* Otherwise, if any of the following conditions are true, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 1.
* The prediction unit containing sample p0 has different reference pictures or a different number of motion vectors with the prediction unit containing the sample q0.

NOTE – The determination of whether the reference pictures used for the two prediction are the same or different is based on which pictures are referenced, without regard to whether a prediction is formed using an index into list 0 or an index into list 1, and also without regard to whether or not the index position within a reference picture list is different or not.

* One motion vector is used to predict the prediction unit containing sample p0, one motion vector is used to predict the prediction unit containing sample q0, and the absolute difference between the horizontal or vertical component of the motion vector used is greater than or equal to 4 in units of quarter luma samples.
* [Ed.: (WJ) needs to be checked again whether this condition covers all 2-motion cases] Two motion vectors are used to predict the prediction unit containing sample p0, two motion vectors are used to predict the prediction unit containing sample q0, and at least one of the motion vector pairs corresponding the same reference pictures and the different boundary samples p0 and q0 satisfies the following condition:

1. The absolute difference between the horizontal or vertical component of a motion vector used in the prediction of the two prediction units is greater than or equal to 4 in units of quarter luma samples.

* Otherwise, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 0.
* Depending on the value of filterDir, the variables bSCb[ filterDir ][ xEk ][ yEj ] and bSCr[ filterDir ][ xEk ][ yEj ] are derived as follows.
* If the block edge is also a coding unit edge and the following condition is true, the variables bSCb[ 0 ][ xEk ][ yEj ] and bSCr[ filterDir ][ xEk ][ yEj ] are set equal to 4.
* The sample p0 or q0 is in a coding unit coded with intra prediction mode
* Otherwise, if the following condition is true, the variables bSCb[ 0 ][ xEk ][ yEj ] and bSCr[ filterDir ][ xEk ][ yEj ] are set equal to 3.
* The sample p0 or q0 is in a coding unit coded with intra prediction mode
* Otherwise,

- If the block edge is also a transform unit edge and the following condition is true, the variable bSCb[ filterDir ][ xEk ][ yEj ] is set equal to 2,

-The sample p0 or q0 is in a transform unit which contains non-zero chroma component Cb transform coefficient level.

-Otherwise, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 0.

- If the block edge is also a transform unit edge and the following condition is true, the variable bSCr[ filterDir ][ xEk ][ yEj ] is set equal to 2.

-The sample p0 or q0 is in a transform unit which contains non-zero chroma component Cr transform coefficient level.

-Otherwise, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 0.

# 

#### Filtering process for coding unit

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 log2CUSize specifying the coding unit size,

– an array bS specifying the boundary filtering strength.

Output of this process is:

– modified reconstruction of the picture.

The filtering process for luma edges in the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << ( log2CUSize – 3 ).
2. For xDk set equal to xB+( k << 3 ), k=0..nD – 1, the following applies:

* For yDm set equal to yB+( m << 3 ), m=0..nD – 1, the following ordered steps apply:
  + 1. Boundary filtering strength bSVer is derived as follows:

bSVer = Max( bS[ 0 ][ xDk ][ yDm + i ] ) for i = 0..7 (8‑428)

* + 1. Filtering process for luma block edge in subclause 8.6.1.4.1 is invoked with the luma location of the coding unit ( xC, yC ), the luma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 1, and the boundary filtering strength bSVer as inputs and the modified luma picture buffer as output.

1. For yDm set equal to yB+( m << 3 ), m=0..nD - 1, the following applies:

* For xDk set equal to xB+( k << 3 ), k=0..nD – 1, the following ordered steps apply:
  + 1. Boundary filtering strength bSHor is derived as follows:

bSHor = Max( bS[ 1 ][ xDk + i][ yDm ] ) for i = 0..7 (8‑429)

* + 1. Filtering process for luma block edge in subclause is invoked with the luma location of the coding unit ( xC, yC ), the luma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 0, and the boundary filtering strength bSHor as inputs and the modified luma picture buffer as output.

The filtering process for chroma edges in the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << ( log2CUSize – 3 ).
2. For xDk set equal to ( xB / 2 )+( k << 2 ), k=0..nD – 1, the following applies:

* For yDm set equal to ( yB / 2)+( m << 2 ), m=0..nD – 1, the following ordered steps apply:

1. Boundary filtering strength bSVer is derived as follows:

bSVer = bSCb[ 0 ][ xDk\*2 ][ yDm\*2 ] (8‑430)

1. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit ( xC/2, yC/2 ), the chroma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 1, a chroma component index cIdx set equal to 1 and the boundary filtering strength bSVer as inputs and the modified chroma picture buffer as output.
2. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit ( xC/2, yC/2 ), the chroma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 1, a chroma component index cIdx set equal to 2 and the boundary filtering strength bSVer as inputs and the modified chroma picture buffer as output.
3. For yDm set equal to ( yB / 2 )+( m << 2 ), m=0..nD – 1, the following applies:

* For xDk set equal to ( xB / 2 )+( k << 2 ), k=0..nD – 1, the following ordered steps apply:

1. Boundary filtering strength bSHor is derived as follows:

bSHor = bSCr[ 1 ][ xDk\*2 ][ yDm\*2 ] (8‑431)

1. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit ( xC/2, yC/2 ), the chroma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 0, a chroma component index cIdx set equal to 1 and the boundary filtering strength bSHor as inputs and the modified chroma picture buffer as output.
2. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit ( xC/2, yC/2 ), the chroma location of the block ( xDk, yDm ), a variable verticalEdgeFlag set equal to 0, a chroma component index cIdx set equal to 2 and the boundary filtering strength bSHor as inputs and the modified chroma picture buffer as output.

##### 8.6.1.4.1 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:

–

– modified reconstruction of the picture.

Let s’ represent the luma sample array recPictureL of the current picture.

A variable β is specified as Table 8‑13 with luma quantization parameter qPL as input.

A variable tC is specified as follows:

– If bS is greater than 2, the variable tC is specified specified as Table 8‑13 with luma quantization parameter ( qPL + 2 ) as input,

– Otherwise (bS is equal or less than 2), the variable tC is specified specified as Table 8‑13 with luma quantization parameter qPL 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..7 are derived as follows:

qi,k = s’[ xC + xB +i, yC + yB + k ] (8‑432)

pi,k = s’[ xC + xB – i – 1, yC + yB + k ] (8‑433)

1. The variables dp, dq and d are derived as follows:

dp = | p2,2 – 2\*p1,2 + p0,2 |  + | p2,5 – 2\*p1,5 + p0,5 |  (8‑434)

dq = | q2,2 – 2\*q1,2 + q0,2 | + | q2,5 – 2\*q1,5 + q0,5 | (8‑434)

d = dp+dq

1. If bS is not equal to 0 and d is less than β, for each sample location ( xC + xB, yC + yB + k ), k = 0..7, the following ordered steps apply:
2. The filtering process for a luma sample specified in subclause 8.6.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, the boundary filtering strength bS and the variables d, dp, dq, β and tC as inputs and the number of filtered samples nD and the filtered sample values pi,k’ and qi,k’ as outputs.
3. The filtered sample values pi,k’ and qi,k’ with i = 0..nD – 1 replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +i, yC + yB + k ] = qi,k’ (8‑435)

s’[ xC + xB – i – 1, yC + yB + k ] = pi,k’ (8‑436)

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

1. The sample values pi,k and qi,k with i = 0..3 and k = 0..7 are derived as follows:

qi,k = s’[ xC + xB +k, yC + yB + i ] (8‑437)

pi,k = s’[ xC + xB +k, yC + yB – i – 1 ] (8‑438)

1. The variables dp, dq and d are derived as follows:

dp = | p2,2 – 2\*p1,2 + p0,2 |  + | p2,5 – 2\*p1,5 + p0,5 |  (8‑434)

dq = | q2,2 – 2\*q1,2 + q0,2 | + | q2,5 – 2\*q1,5 + q0,5 | (8‑434)

d = dp+dq

3.If bS is not equal to 0 and d is less than β, for each sample location ( xC + xB + k, yC + yB ), k = 0..7, the following ordered steps apply:

1. The filtering process for a luma sample specified in subclause 8.6.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, the boundary filtering strength bS and the variables d, dp, dq, β and tC as inputs and the number of filtered samples nD and the filtered sample values pi’ and qi’ as outputs.
2. The filtered sample values pi,k’ and qi,k’ with i = 0..nD – 1 replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +k, yC + yB + i ] = qi,k’ (8‑440)

s’[ xC + xB +k, yC + yB – i – 1 ] = pi,k’

##### 8.6.1.4.3 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,

– variables bS, bSCr and bSCb specifying the boundary filtering strength,

– a variable cIdx specifying the chroma component index.

Output of this process is:

– modified reconstruction of the picture.

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.

A variable β is specified as Table 8‑13 with chroma quantization parameter qPC as input

A variable tC is specified as follows:

– If bS is greater than 2, the variable tC is specified specified as Table 8‑13 with luma quantization parameter ( qPL + 2 ) as input,

– Otherwise (bS is equal or less than 2), the variable tC is specified as Table 8‑13 with luma quantization parameter qPL 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..1 and k=0,…,3, are derived as follows:

qi,k = s’[ xC + xB +i, yC + yB + k ] (8‑442)

pi,k = s’[ xC + xB – i – 1, yC + yB + k ] (8‑443)

1. The variable d is derived as follows:

d = |p0,1 – p1,1| + |q0,1 – q1,1| + |p0,2 – p1,2| + |q0,2 – q1,2|

1. If cIdx is equal to 1 and bSCb is greater than 1, and d is less than β, for each sample location ( xC + xB, yC + yB + k ), k = 0..3, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bSCb and the variable tC as inputs and the filtered sample values p0,k’ and q0,k’ as outputs.
3. The filtered sample values p0,k’ and q0,k’ replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB , yC + yB + k ] = q0,k’ (8‑444)

s’[ xC + xB – 1, yC + yB + k ] = p0,k’ (8‑445)

1. If cIdx is equal to 2 and bSCr is greater than 1, and d is less than β, for each sample location ( xC + xB, yC + yB + k ), k = 0..3, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bSCr and the variable tC as inputs and the filtered sample values p0,k’ and q0,k’ as outputs.
3. The filtered sample values p0,k’ and q0,k’ replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB , yC + yB + k ] = q0,k’ (8‑444)

s’[ xC + xB – 1, yC + yB + k ] = p0,k’

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

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

qi,k = s’[ xC + xB +k, yC + yB + i ] (8‑446)

pi,k = s’[ xC + xB +k, yC + yB – i – 1 ] (8‑447)

1. The variable d is derived as follows:

d = |p0,1 – p1,1| + |q0,1 – q1,1| + |p0,2 – p1,2| + |q0,2 – q1,2|

1. If cIdx is equal to 1 and bSCb is greater than 1, and d is less than β, for each sample location ( xC + xB +k, yC + yB ), k = 0..3, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bSCb and the variable tC as inputs and the filtered sample values p0,k’ and q0,k’ as outputs.
3. The filtered sample values p0,k’ and q0,k’ replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +k, yC + yB ] = q0,k’ (8‑448)

s’[ xC + xB +k, yC + yB – 1 ] = p0,k’

1. If cIdx is equal to 2 and bSCr is greater than 1, and d is less than β, for each sample location ( xC + xB +k, yC + yB ), k = 0..3, the following ordered steps apply:
2. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values pi,k, qi,k, with i = 0..1, the boundary filtering strength bSCr and the variable tC as inputs and the filtered sample values p0,k’ and q0,k’ as outputs.
3. The filtered sample values p0,k’ and q0,k’ replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +k, yC + yB ] = q0,k’ (8‑448)

s’[ xC + xB +k, yC + yB – 1 ] = p0,k’

##### 8.6.1.4.5 Filtering process for a luma sample

[Ed: (WJ) no filtering when bS is equal to 0]

Inputs of this process are:

– sample values, pi and qi with i = 0..3,

– a variable bS specifying the boundary filtering strength,

– variables d, dp, dq, β and tC.

Output of this process is:

– number of filtered samples nD,

– filtered sample values, pi’ and qi’ with i = 0..nD – 1,

When the variable bS is not equal to 0, the following applies:

– If d is less than ( β >> 2 ), | p3 – p0 | + | q0 – q3 | is less than ( β >> 3 ) and | p0 – q0 | is less than ( 5\*tC + 1 ) >> 1, the following strong filtering applies while nD is set equal to 3:

p0’ = Clip1Y( ( p2 + 2\*p1 + 2\*p0 + 2\*q0 + q1 + 4 ) >> 3 ) (8‑450)

p1’ = Clip1Y( ( p2 + p1 + p0 + q0 + 2 ) >> 2 ) (8‑451)

p2’ = Clip1Y( ( 2\*p3 + 3\*p2 + p1 + p0 + q0 + 4 ) >> 3 ) (8‑452)

q0’ = Clip1Y( ( p1 + 2\*p0 + 2\*q0 + 2\*q1 + q2 + 4 ) >> 3 ) (8‑453)

q1’ = Clip1Y( ( p0 + q0 + q1 + q2 + 2 ) >> 2 ) (8‑454)

q2’ = Clip1Y( ( p0 + q0 + q1 + 3\*q2 + 2\*q3 + 4 ) >> 3 ) (8‑455)

– Otherwise, the following weak filtering applies while nD is set equal to 2:

Δ =  (9\*(q0-p0)-3\*(q1-p1)+8)>>4 (8‑456)

If abs( Δ) is smaller than tC \*10 the following steps apply

Δ = Clip3( -tC, tC, Δ) (8‑456)

p0’ = Clip1Y( p0 + Δ ) (8‑457)

q0’ = Clip1Y( q0 - Δ ) (8‑458)

If dp is smaller than (β+ (β>>1))>>3 the following steps apply[[1]](#footnote-2)

Δp = Clip3( - (tC>>1), tC>>1, (((p2+p0+1)>>1)- p1+Δ)>>1 )

p1’ = Clip1Y( p1 + Δp ) (8‑459)

If dq is smaller than (β+ (β>>1))>>3 the following steps apply[[2]](#footnote-3)

Δq = Clip3( - (tC>>1), tC>>1, (((q2+q0+1)>>1)- q1-Δ)>>1 )

q1’ = Clip1Y( q1 + Δq ) (8‑460)

##### 8.6.1.4.6 Filtering process for a chroma sample

[Ed: (WJ) no filtering when bS is equal or less than 2]

Inputs of this process are:

– sample values, pi and qi with i = 0..1,

– a variable bS specifying the boundary filtering strength.

– a variable tC.

Output of this process is:

– The filtered sample values, p0’ and q0’.

When the variable bS is greater than 1, the filtered sample values p0’ and q0’ are derived by

Δ = Clip3( -tC, tC, ( ( 12\*( q0 – p0 ) ) + 3\*(p1 – q1) + 16 ) >> 5 ) ) (8‑461)

p0’ = Clip1C( p0 + Δ ) (8‑462)

q0’ = Clip1C( q0 - Δ )

1. This condition can be computed instead on a higher level, in ”Filtering process for luma block edge subsection” as it is applied to all the lines in the current 8-pixel edge. [↑](#footnote-ref-2)
2. This condition can be computed instead on a higher level, in ”Filtering process for luma block edge” subsection as it is applied to all the lines in the current 8-pixel edge. [↑](#footnote-ref-3)