**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 and bSU and bSV specifying the boundary filtering strength

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 following condition is true, the variable bS[ 0 ][ xEk ][ yEj ] is set equal to 4.
* The prediction unit containing sample p0 has different luma intra prediction mode with the prediction unit containing the sample q0.
* Otherwise, if the following condition is true, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 3.
* The prediction unit containing sample p0 has the same luma intra prediction mode with the prediction unit containing the sample q0.
* Otherwise, if 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.
2. Otherwise, the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 0.

* Depending on the value of filterDir, the variable bSU[ filterDir ][ xEk ][ yEj ] and bSV[ filterDir ][ xEk ][ yEj ] are derived as follows.
* If the following condition is true, the variable bSU[ filterDir  ][ xEk ][ yEj ] and bSV[ filterDir ][ xEk ][ yEj ] are set equal to 4.
* The prediction unit containing sample p0 has different chroma intra prediction mode with the prediction unit containing the sample q0.
* Otherwise, if the following condition is true, the variable bSU[ filterDir ][ xEk ][ yEj ] and bSV[ filterDir ][ xEk ][ yEj ] are set equal to 3.
* The prediction unit containing sample p0 has the same chroma intra prediction mode with the prediction unit containing the sample q0.
* Otherwise,
* If the following condition is true, the variable bSU[ filterDir ][ xEk ][ yEj ] is set equal to 2,
  + 1. The sample p0 or q0 is in a transform unit which contains non-zero chroma component U transform coefficient level.
* Otherwise, the variable bSU[ filterDir ][ xEk ][ yEj ] is set equal to 0.
* If the following condition is true, the variable bSU[ filterDir ][ xEk ][ yEj ] is set equal to 2,
  + 1. The sample p0 or q0 is in a transform unit which contains non-zero chroma component V transform coefficient level.
* Otherwise, the variable bSV[ filterDir ][ xEk ][ yEj ] is set equal to 0.

##### 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:

– a variable dE containing a decision,

– one-dimensional array of size (8), dS containing decisions.

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

A variables 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 3, the variable tC is specified as Table 8‑14 with luma quantization parameter ( qPL + 3 ) as input,

– Otherwise, if bS is greater than 2, the variable tC is specified as Table 8‑14 with luma quantization parameter ( qPL + 1 ) as input

– Otherwise (bS is equal or less than 2), the variable tC is specified as Table 8‑14 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 = 2,5 are derived as follows:

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

pi,k = s’[ xC + xB – i – 1, yC + yB + k ] (8‑448)

1. The variable dp0 and dq0 and d is derived as follows:

d p0 = | p2,2 – 2\*p1,2 + p0,2 | +  | p2,5 – 2\*p1,5 + p0,5 | (8‑449)

d q0 = | q2,2 – 2\*q1,2 + q0,2 | +  | q2,5 – 2\*q1,5 + q0,5 | (8‑449)

d = d p0 + d q0 (8‑449)

1. The variable dE is set equal to 0.
2. If bS is not equal to 0 and d is less than β, the following ordered steps apply:
3. for each sample location ( xC + xB, yC + yB + k ), k = 0..7, the following ordered steps apply:
   1. The decision process for a luma sample specified in subclause 8.6.1.4.4 is invoked with sample values pi,k, qi,k with i = 0..3, the boundary filtering strength bS and the variables d, β and tC as inputs and a decision dSam as output.
   2. The variable dS[k] is set equal to dSam
4. The variable dE is set equal to 1.

– 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 = 2,5 are derived as follows:

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

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

1. The variable dp0 and dq0 and d are derived as follows:

d p0 = | p2,2 – 2\*p1,2 + p0,2 | +  | p2,5 – 2\*p1,5 + p0,5 | (8‑437)

d q0 = | q2,2 – 2\*q1,2 + q0,2 | +  | q2,5 – 2\*q1,5 + q0,5 | (8‑437)

d = d p0 + d q0 (8‑437)

1. The variable dE is set equal to 0.
2. If bS is not equal to 0 and d is less than β, the following ordered steps apply:
3. The variable dp1 and dq1 are derived as follows:

d p1 = | p3,2 – 2\*p2,2 + p1,2 | +  | p3,5 – 2\*p2,5 + p1,5 | (8‑437)

d q1 = | q3,2 – 2\*q2,2 + q1,2 | +  | q3,5 – 2\*q2,5 + q1,5 | (8‑437)

1. For each sample location ( xC + xB + k, yC + yB ), k = 0..7, the following ordered steps apply:
   1. The decision process for a luma sample specified in subclause 8.6.1.4.4 is invoked with sample values pi,k, qi,k with i = 0..3, the boundary filtering strength bS and the variables d, dp1, dq1, β and tC as inputs and a decision dSam as output.
   2. The variable dS[k] is set equal to dSam.
2. The variable dE is set equal to 1.

##### 8.6.4.1.2 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,

– a variable dE containing a decision,

– one-dimensional array of size (8), dS containing decisions,

– a variable bSL,

– a variable tCL,

Output of this process is:

– modified reconstruction of the picture.

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

Depending on pcm\_flag, a variable β is specified as follows:

– If pcm\_flag is equal to 1, the variables β is specified as Table 8‑11 with luma quantization parameter 0 as input.

– Otherwise, the variables β is specified as Table 8‑11 with luma quantization parameter qPL as input.

A variable tC is specified as follows:

– If bS is greater than 3, the variable tC is specified as Table 8‑14 with luma quantization parameter ( qPL + 3 ) as input,

– Otherwise, if bS is greater than 2, the variable tC is specified as Table 8‑14 with luma quantization parameter ( qPL + 1 ) as input

– Otherwise (bS is equal or less than 2), the variable tC is specified as Table 8‑14 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. If dE is not equal to 0, 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 decision dS[k], the boundary filtering strength bS and the variable tC as inputs and the number of filtered samples nD and the filtered sample values pi’ and qi’ as outputs.
3. The filtered sample values pi’ and qi’ with i = 0..nD – 1 replace the corresponding samples inside the sample array s’ as follows:

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

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

– 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 -3 and 4 respectively. If xD 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 7 respectively.
2. The sample values pi,k and qi,k with i = 0..3 and k = ks..ke 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. If xPOS is less than 0 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 8.6.1.4.5 is invoked with sample values pi,k, qi,k with i = 0..3, decision dSL[k+3], the boundary filtering strength bSL and the variable tCL as inputs and the number of filtered samples nD and the filtered sample values pi’ and qi’ as outputs.
3. The filtered sample values pi’ and qi’ with i = 0..nD – 1 replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +k, yC + yB + i ] = qi’ (8‑459)

s’[ xC + xB +k, yC + yB – i – 1 ] = pi’ (8‑460)

1. If dE is not equal to 0, for each sample location ( xC + xB + k, yC + yB ), k = 0.. ke , 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, decision dS[k], the boundary filtering strength bS and the variable tC as inputs and the number of filtered samples nD and the filtered sample values pi’ and qi’ as outputs.
3. The filtered sample values pi’ and qi’ with i = 0..nD – 1 replace the corresponding samples inside the sample array s’ as follows:

s’[ xC + xB +k, yC + yB + i ] = qi’ (8‑461)

s’[ xC + xB +k, yC + yB – i – 1 ] = pi’ (8‑462)

##### 8.6.4.1.3 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,

– a variable bSU and bSV specifying the boundary filtering strength,

– a variable cIdx specifying the chroma component index.

– a variable xPOS,

– a variable bSU and bSV,

– a variable tCU and tCV

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 tCU is specified as follows:

– If bSU is greater than 3, the variable tCU is specified as Table 8‑14 with luma quantization parameter ( qPL + 3 ) as input,

– Otherwise, if bSU is greater than 2, the variable tCU is specified as Table 8‑14 with luma quantization parameter ( qPL + 1 ) as input

– Otherwise (bSU is equal or less than 2), the variable tCU is specified as Table 8‑14 with luma quantization parameter qPL as input.

A variable tCV is specified as follows:

– If bSV is greater than 3, the variable tCV is specified as Table 8‑14 with luma quantization parameter ( qPL + 3 ) as input,

– Otherwise, if bSV is greater than 2, the variable tCV is specified as Table 8‑14 with luma quantization parameter ( qPL + 1 ) as input

– Otherwise (bSV is equal or less than 2), the variable tCV is specified as Table 8‑14 with luma quantization parameter qPL 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 sample values pi and qi with i = 0..1 are derived as follows:

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

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

1. If cIdx is equal to 1, and if bSU is greater than 0, 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, qi, with i = 0..1, the boundary filtering strength bSU and the variable tCU 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 , yC + yB + k ] = q0’ (8‑444)

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

1. If cIdx is equal to 2, and if bSV is greater than 0, 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, qi, with i = 0..1, the boundary filtering strength bSV and the variable tCV 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 , yC + yB + k ] = q0’ (8‑444)

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

– 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 -1 and 2 respectively. If xD 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.
2. The sample values pi and qi with i = 0..1 and k = ks..ke are derived as follows:

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

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

1. If xPOS is less than 0, and if cIdx is equal to 1, and if bSU, is greater than 0, for each sample location ( xC + xB - 1, yC + yB ), 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, qi, with i = 0..1, the boundary filtering strength bSU and the variable tCU 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’ (8‑452)

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

1. If xPOS is less than 0, and if cIdx is equal to 2, and if bSV, is greater than 0, for each sample location ( xC + xB - 1, yC + yB ), 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, qi, with i = 0..1, the boundary filtering strength bSV and the variable tCV 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’ (8‑452)

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

1. If cIdx is equal to 1, and if bSU is greater than 0, 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 8.6.1.4.6 is invoked with sample values pi, qi, with i = 0..1, the boundary filtering strength bSU and the variable tCU 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’ (8‑448)

s’[ xC + xB +k, yC + yB – 1 ] = p0’ (8‑449)

1. If cIdx is equal to 2, and if bSV is greater than 0, 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 8.6.1.4.6 is invoked with sample values pi, qi, with i = 0..1, the boundary filtering strength bSV and the variable tCV 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’ (8‑448)

s’[ xC + xB +k, yC + yB – 1 ] = p0’ (8‑449)

##### 8.6.1.4.4 Decision 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..2,

– a variable bS specifying the boundary filtering strength,

– variables dp1, dq1, d, β and tC.

Output of this process is:

– a variable dSam containing a decision

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

– If both dp1 and dq1 are less than ( β / 6 ), 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, dSam is set equal to 1.

– Otherwise, dSam is set equal to 0.

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

Inputs of this process are:

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

– a variable dSam containing a decision,

– a variable dp0 and dq0 containing a decision,

– a variable tC.

Output of this process is:

– number of filtered samples nD,

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

Depending on dSam, the following applies:

– When the variable dSam is equal to 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 = Clip3( -tC, tC, ( ( ( q0 – p0 ) << 2 ) +  ( q1 – p1 ) + 4 ) >> 3 ) (8‑456)

– If both dp0 and dq0 are less than ( β / 6 ),

p0’ = Clip1Y( p0 + Δ ) (8‑457)

q0’ = Clip1Y( q0 - Δ ) (8‑458)

– Otherwise,

p0’ = Clip1Y( p0 + Δ/2 ) (8‑457)

q0’ = Clip1Y( q0 - Δ/2 ) (8‑458)

– If dp0 is less than ( β / 6 ),

p1’ = Clip1Y( p1 + Δ/2 ) (8‑459)

– If dq0 is less than ( β / 6 ),

q1’ = Clip1Y( q1 – Δ/2 ) (8‑460)

Each of the filtered sample values, pi’ with i = 0..nD-1, is substituted by the corresponding input sample value pi if all of the following conditions are true.

– pi is a sample of an I\_PCM block.

– pcm\_loop\_filter\_disable\_flag value is equal to 1.

Similary, each of the filtered sample values, qi’ with i = 0..nD-1, is substituted by the corresponding input sample value qi if all of the following conditions are true.

– qi is a sample of an I\_PCM block.

– pcm\_loop\_filter\_disable\_flag value is equal to 1.

[Ed. (WJ): for PCM case, deblocking filter applies first and the filtered pixels are restored. Rather than this, it’s better to skip the filtering itself for PCM samples since first filtering is actually not needed.]

Table ‑ – Derivation of threshold variables β and tC from input Q

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Q** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| **β** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 7 | 8 |
| **tC** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **Q** | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| **β** | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 |
| **tC** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 |
| **Q** | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |  |
| **β** | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 64 | 64 | 64 | 64 |  |
| **tC** | 5 | 5 | 6 | 6 | 7 | 8 | 9 | 9 | 10 | 10 | 11 | 11 | 12 | 12 | 13 | 13 | 14 | 14 |  |

##### 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 0, the filtered sample values p0’ and q0’ are derived by

Δ = Clip3( -tC, tC, ( ( ( ( q0 – p0 ) << 2 ) + p1 – q1 + 4 ) >> 3 ) ) (8‑461)

p0’ = Clip1C( p0 + Δ ) (8‑462)

q0’ = Clip1C( q0 - Δ ) (8‑463)

The filtered sample value, p0’ is substituted by the corresponding input sample value p0 if all of the following conditions are true.

– p0 is a sample of an I\_PCM block.

– pcm\_loop\_filter\_disable\_flag value is equal to 1.

Similary, the filtered sample value, q0’ is substituted by the corresponding input sample value q0 if all of the following conditions are true.

– q0 is a sample of an I\_PCM block.

– pcm\_loop\_filter\_disable\_flag value is equal to 1.

[Ed. (WJ): for PCM case, deblocking filter applies first and the filtered pixels are restored. Rather than this, it’s better to skip the filtering itself for PCM samples since first filtering is actually not needed.]