# Number of operations

# 8.7.2.4 bS calculation

31 compares

12 add/sub

The variables xDi, yDj, xN, and yN are derived as follows:

* If edgeType == EDGE\_VER,   
   xDi = ( i  <<  3 )  
   yDj = ( j  <<  2 )  
   xN = ( 1  <<  ( log2CbSize − 3 ) ) − 1  
   yN = ( 1  <<  ( log2CbSize − 2 ) ) − 1.
* Otherwise (edgeType == EDGE\_HOR)  
   xDi = ( i  <<  2 )  
   yDj = ( j  <<  3 )  
   xN = ( 1  <<  ( log2CbSize − 2 ) ) − 1  
   yN = ( 1  <<  ( log2CbSize − 3 ) ) − 1.

For xDi with i = 0..xN and yDj with j = 0..yN, the following applies:

* If edgeFlags[ xDi ][ yDj ] == 0  
   the variable bS[ xDi ][ yDj ] = 0
* Otherwise (edgeFlags[ xDi ][ yDj ] is equal to 1), the following applies:
* The sample values p0 and q0 are derived as follows:
  + - If edgeType == EDGE\_VER  
       p0 = recPictureL[ xCb + xDi − 1 ][ yCb + yDj ]  
       q0 = recPictureL[ xCb + xDi ][ yCb + yDj ]
    - Otherwise (edgeType is equal to EDGE\_HOR)  
       p0 = recPictureL[ xCb + xDi ][ yCb + yDj − 1 ]  
       q0 = recPictureL[ xCb + xDi ][ yCb + yDj ]
* The variable bS[ xDi ][ yDj ] is derived as follows:
  + - If block(p0) == MODE\_INTRA || block(q0) == MODE\_INTRA  
       bS[ xDi ][ yDj ] = 2
    - Otherwise, if (TU(p0) != TU(q0) && (NumNonZeroCoeffLuma( block(p0) ) > 0 || NumNonZeroCoeffLuma( block(q0) ) > 0))  
       bS[ xDi ][ yDj ] = 1
    - Otherwise, if one or more of the following conditions are true, bS[ xDi ][ yDj ] = 1:
      * (NumRefPic(block(p0) != NumRefPic(block(q0)) || (NumMv(block(p0)) != NumMv(block(q0)))
      * (NumMv(block(p0)) == 1 && NumMv(block(q0) == 1)) &&   
        ((Abs(MvHor(block(p0) - MvHor(block(q0)) >=4 ) ||  
        Abs(MvVer(block(p0) - MvVer(block(q0)) >= 4))
      * NumMv(block(p0)) == 2 && (RefPic(Mv0(block(p0)) != RefPic(Mv1(block(p0)) )&&  
        NumMv(block(q0)) == 2 && (RefPic(Mv0(block(q0)) == RefPic(Mv1(block(q0)) ))&&  
        Abs(Mv0Hor(block(q0)) - Mv1Hor(block(q0))) >= 4 ) ||  
        Abs(Mv0Ver(block(q0)) - Mv1Ver(block(q0))) >= 4))
      * NumMv(block(p0)) == 2 && (RefPic(Mv0(block(p0)) == RefPic(Mv1(block(p0)) )&&  
        NumMv(block(q0)) == 2 && (RefPic(Mv0(block(q0)) == RefPic(Mv1(block(q0)) ))&&
        + Abs(Mv0Hor(block(p0)) - Mv1Hor(block(p0))) >=4 ||  
          Abs(Mv0Ver(block(p0))- Mv1Ver(block(p0))) >=4 ||  
          Abs(Mv1Hor(block(q0)) - Mv1Hor(block(q0))) >=4 ||  
          Abs(Mv1Ver(block(q0)) - Mv1Ver(block(q0))) >=4
        + Abs(Mv0Hor(block(p0)) - Mv0Hor(block(q0))) >=4 ||  
          Abs(Mv0Ver(block(p0)) - Mv0Ver(block(q0)) )>=4 ||  
          Abs(Mv1Hor(block(p0)) - Mv1Hor(block(q0))) >=4 ||  
          Abs(Mv1Ver(block(p0)) - Mv1Ver(block(q0))) >=4
    - Otherwise, bS[ xDi ][ yDj] = 0

# 8.7.2.5.3

23\*N\*2log2N-5 add/sub, 2 LUT access (for a CU), 10\*N\*2log2N-5 compare

qPL = ( ( QpQ + QpP + 1 )  >>  1 ) (8‑288)

Q = Clip3( 0, 51, qPL + ( slice\_beta\_offset\_div2  <<  1 ) ) (8‑289)

β = β′ \* ( 1  <<  ( BitDepthY − 8 ) ) (8‑290)

Q = Clip3( 0, 53, qPL + 2 \* ( bS − 1 ) + ( slice\_tc\_offset\_div2  <<  1 ) ) (8‑291)

tC = tC′ \* ( 1  <<  ( BitDepthY − 8 ) ) (8‑292)

Depending on the value of edgeType, the following applies:

– If edgeType == EDGE\_VER :

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

dp0 = Abs( p2,0 − 2 \* p1,0 + p0,0 ) (8‑293)

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

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

dq3 = Abs( 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)

d = dpq0 + dpq3 (8‑301)

1. The variables dE = 0, dEp = 0, and dEq = 0
2. When d < β, the following ordered steps apply:
3. dpq = 2 \* dpq0
4. – If (dpq < ( β  >>  2 ) && (Abs( p3 − p0 ) + Abs( q0 − q3 ) < ( β  >>  3 )) && Abs( p0 − q0 ) < ( 5 \* tC + 1 )  >>  1)  
    dSam0 = 1 (from 8.7.2.5.7)

– Otherwise, dSam0 = 0.

1. dpq = 2 \* dpq3.
2. – If (dpq < ( β  >>  2 ) && (Abs( p3 − p0 ) + Abs( q0 − q3 ) < ( β  >>  3 )) && Abs( p0 − q0 ) < ( 5 \* tC + 1 )  >>  1)  
    dSam3 = 1 (from 8.7.2.5.7)

– Otherwise, dSam3 = 0

1. dE = 1.
2. When dSam0 == 1 and dSam3 == 1, dE = 2.
3. When dp < ( β + ( β  >>  1 ) )  >>  3, dEp = 1.
4. When dq < ( β + ( β  >>  1 ) )  >>  3, dEq = 1.

– Otherwise (edgeType is equal to EDGE\_HOR), the following ordered steps apply:

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

dp0 = Abs( p2,0 − 2 \* p1,0 + p0,0 ) (8‑302)

dp3 = Abs( p2,3 − 2 \* p1,3 + p0,3 ) (8‑303)

dq0 = Abs( q2,0 − 2 \* q1,0 + q0,0 ) (8‑304)

dq3 = Abs( q2,3 − 2 \* q1,3 + q0,3 ) (8‑305)

dpq0 = dp0 + dq0 (8‑306)

dpq3 = dp3 + dq3 (8‑307)

dp = dp0 + dp3 (8‑308)

dq = dq0 + dq3 (8‑309)

d = dpq0 + dpq3 (8‑310)

1. The variables dE, dEp, and dEq are set equal to 0.
2. When d is less than β, the following ordered steps apply:
3. The variable dpq is set equal to 2 \* dpq0.
4. For the sample location ( xCb + xBl, yCb + yBl ), the decision process for a luma sample as specified in subclause is invoked with sample values p0,0, p3,0, q0,0, and q3,0, the variables dpq, β, and tC as inputs, and the output is assigned to the decision dSam0.
5. The variable dpq is set equal to 2 \* dpq3.
6. For the sample location ( xCb + xBl + 3, yCb + yBl ), the decision process for a luma sample as specified in subclause is invoked with sample values p0,3, p3,3, q0,3, and q3,3, the variables dpq, β, and tC as inputs, and the output is assigned to the decision dSam3.
7. The variable dE is set equal to 1.
8. When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.
9. When dp is less than ( β + ( β  >>  1 ) )  >>  3, the variable dEp is set equal to 1.
10. When dq is less than ( β + ( β  >>  1 ) )  >>  3, the variable dEq is set equal to 1.

# 8.7.2.5.7

add/sub :

- Normal filter : 17\*(N\*2log2N-5)

- Strong filter : 42\*(N\*2log2N-5)

compare :

- Normal filter : 9\*(N\*2log2N-5)

- Strong filter : 1\*(N\*2log2N-5)

– If dE == 2

nDp = 3, nDq = 3

p0′ = Clip3( p0 − 2 \* tC, p0 + 2 \* tC, ( p2 + 2 \* p1 + 2 \* p0 + 2 \* q0 + q1 + 4 )  >>  3 ) (8‑330)

p1′ = Clip3( p1 − 2 \* tC, p1 + 2 \* tC, ( p2 + p1 + p0 + q0 + 2 )  >>  2 ) (8‑331)

p2′ = Clip3( p2 − 2 \* tC, p2 + 2\*tC, ( 2 \* p3 + 3 \* p2 + p1 + p0 + q0 + 4 )  >>  3 ) (8‑332)

q0′ = Clip3( q0 − 2 \* tC, q0 + 2 \* tC, ( p1 + 2 \* p0 + 2 \* q0 + 2 \* q1 + q2 + 4 )  >>  3 ) (8‑333)

q1′ = Clip3( q1 − 2 \* tC, q1 + 2 \* tC, ( p0 + q0 + q1 + q2 + 2 )  >>  2 ) (8‑334)

q2′= Clip3( q2 − 2 \* tC, q2 + 2 \* tC, ( p0 + q0 + q1 + 3 \* q2 + 2 \* q3 + 4 )  >>  3 ) (8‑335)

– Otherwise, nDp = ,0 nDq = 0, and the following weak filtering applies:

* + The following applies:

Δ = ( 9 \* ( q0 −  p0 ) − 3 \* ( q1 − p1 ) + 8 )  >>  4 (8‑336)

* + When Abs(Δ) < tC \* 10, the following ordered steps apply:
    - The filtered sample values p0′ and q0′ are specified as follows:

Δ = Clip3( −tC, tC, Δ ) (8‑337)

p0′ = Clip1Y( p0 + Δ ) (8‑338)

q0′ = Clip1Y( q0 − Δ ) (8‑339)

* + - When dEp == 1, the filtered sample value p1′ is specified as follows:

Δp = Clip3( −( tC  >>  1 ), tC  >>  1, ( ( ( p2 + p0 + 1 )  >>  1 ) − p1 + Δ )  >>  1 ) (8‑340)

p1′ = Clip1Y( p1 + Δp ) (8‑341)

* + - When dEq == 1, the filtered sample value q1′ is specified as follows:

Δq = Clip3( −( tC  >>  1 ), tC  >>  1, ( ( ( q2 + q0 + 1 )  >>  1 ) − q1 − Δ )  >>  1 ) (8‑342)

q1′ = Clip1Y( q1 + Δq ) (8‑343)

* + - nDp is set equal to dEp + 1 and nDq is set equal to dEq + 1.

When nDp > 0 and one or more of the following conditions are true, nDp is set equal to 0:

– pcm\_loop\_filter\_disabled\_flag == 1 and pcm\_flag[ xP0 ][ yP0 ] is == 1.

– cu\_transquant\_bypass\_flag of the coding unit that includes the coding block containing the sample p0 is equal to 1.

When nDq > 0 and one or more of the following conditions are true, nDq is set equal to 0:

– pcm\_loop\_filter\_disabled\_flag == 1 and pcm\_flag[ xQ0 ][ yQ0 ] == 1.

– cu\_transquant\_bypass\_flag of the coding unit that includes the coding block containing the sample q0 is equal to 1.