##### 8.4.5.2.13 Specification of INTRA\_LT\_CCLM, INTRA\_L\_CCLM and INTRA\_T\_CCLM intra prediction mode

Inputs to this process are:

* the intra prediction mode predModeIntra,
* a sample location ( xTbC, yTbC ) of the top-left sample of the current transform block relative to the top-left sample of the current picture,
* a variable nTbW specifying the transform block width,
* a variable nTbH specifying the transform block height,
* a variable cIdx specifying the colour component of the current block,
* chroma neighbouring samples p[ x ][ y ], with x = −1, y = 0..2 \* nTbH − 1 and x = 0.. 2 \* nTbW − 1, y = − 1.

Output of this process are predicted samples predSamples[ x ][ y ], with x = 0..nTbW − 1, y = 0..nTbH − 1.

The current luma location ( xTbY, yTbY ) is derived as follows:

( xTbY, yTbY )  =  ( xTbC << ( SubWidthC − 1 ), yTbC << ( SubHeightC − 1 ) ) (351)

The variables availL, availT and availTL are derived as follows:

* The derivation process for neighbouring block availability as specified in clause 6.4.4 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY, yTbY ), the neighbouring luma location ( xTbY − 1, yTbY ), checkPredModeY set equal to FALSE, and cIdx as inputs, and the output is assigned to availL.
* The derivation process for neighbouring block availability as specified in clause 6.4.4 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY, yTbY ), the neighbouring luma location ( xTbY, yTbY − 1 ), checkPredModeY set equal to FALSE, and cIdx as inputs, and the output is assigned to availT.
* The variable availTL is derived as follows:

availTL  =  availL  &&  availT (352)

* The number of available top-right neighbouring chroma samples numTopRight is derived as follows:
* The variable numTopRight is set equal to 0 and availTR is set equal to TRUE.
* When predModeIntra is equal to INTRA\_T\_CCLM, the following applies for x = nTbW..2 \* nTbW − 1 until availTR is equal to FALSE or x is equal to 2 \* nTbW − 1:
* The derivation process for neighbouring block availability as specified in clause 6.4.4 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY , yTbY ) the neighbouring luma location ( xTbY + x, yTbY − 1 ), checkPredModeY set equal to FALSE, and cIdx as inputs, and the output is assigned to availTR
* When availTR is equal to TRUE, numTopRight is incremented by one.
* The number of available left-below neighbouring chroma samples numLeftBelow is derived as follows:
* The variable numLeftBelow is set equal to 0 and availLB is set equal to TRUE.
* When predModeIntra is equal to INTRA\_L\_CCLM, the following applies for y = nTbH..2 \* nTbH − 1 until availLB is equal to FALSE or y is equal to 2 \* nTbH − 1:
* The derivation process for neighbouring block availability as specified in clause 6.4.4 is invoked with the current luma location ( xCurr, yCurr ) set equal to ( xTbY , yTbY ), the neighbouring luma location ( xTbY − 1, yTbY + y ), checkPredModeY set equal to FALSE, and cIdx as inputs, and the output is assigned to availLB
* When availLB is equal to TRUE, numLeftBelow is incremented by one.

The number of available neighbouring chroma samples on the top and top-right numSampT and the number of available neighbouring chroma samples on the left and left-below numSampL are derived as follows:

* If predModeIntra is equal to INTRA\_LT\_CCLM, the following applies:

numSampT = availT ? nTbW : 0 (353)

numSampL = availL ? nTbH : 0 (354)

* Otherwise, the following applies:

numSampT = ( availT  &&  predModeIntra  = =  INTRA\_T\_CCLM ) ?  
 ( nTbW +Min( numTopRight, nTbH ) ) : 0 (355)

numSampL = ( availL  &&  predModeIntra  = =  INTRA\_L\_CCLM ) ? ( nTbH +  
 Min( numLeftBelow, nTbW ) ) : 0 (356)

The variable bCTUboundary is derived as follows:

bCTUboundary = ( yTbY & ( CtbSizeY − 1 )  = =  0 ) ? TRUE : FALSE. (357)

The variable cntN and array pickPosN with N being replaced by L and T, are derived as follows:

* The variable numIs4N is derived as follows:

numIs4N = ( ( availT  &&  availL  &&  predModeIntra  = =  INTRA\_LT\_CCLM ) ? 0 : 1) (358)

* The variable startPosN is set equal to numSampN  >>  ( 2 + numIs4N ).
* The variable pickStepN is set equal to Max( 1, numSampN  >>  ( 1 + numIs4N ) ).
* If availN is equal to TRUE and predModeIntra is equal to INTRA\_LT\_CCLM or INTRA\_N\_CCLM, the following assignments are made:
  + - cntN is set equal to Min( numSampN, ( 1 + numIs4N ) << 1 ).
    - pickPosN[ pos ] is set equal to (startPosN + pos \* pickStepN), with pos = 0.. cntN − 1.
* Otherwise, cntN is set equal to 0.

The prediction samples predSamples[ x ][ y ] with x = 0..nTbW − 1, y = 0..nTbH − 1 are derived as follows:

* If both numSampL and numSampT are equal to 0, the following applies:

predSamples[ x ][ y ] = 1  <<  ( BitDepth − 1 ) (359)

* Otherwise, the following ordered steps apply:

– The one-dimensional filter coefficients array F1 and F2, and the 2-dimensional filter coefficients arrays F3 and F4 are specified as follows.

~~F1[ 0 ] = 2, F1[ 1 ] = 0 (361)~~

F2[ 0 ] = 1, F2[ 1 ] = 2, F2[ 2 ] = 1 (362)

F3[ i ][ j ] = F4[ i ][ j ] = 0, with i = 0..2, j = 0..2 (363)

– If both SubWidthC and SubHeightC are equal to 2, the following applies:

F1[ 0 ] = 1, F1[ 1 ] = 1 (364)

F3[ 0 ][ 1 ] = 1, F3[ 1 ][ 1 ] = 4, F3[ 2 ][ 1 ] = 1, F3[ 1 ][ 0 ] = 1, F3[ 1 ][ 2 ] = 1 (365)

F4[ 0 ][ 1 ] = 1, F4[ 1 ][ 1 ] = 2, F4[ 2 ][ 1 ] = 1 (366)

F4[ 0 ][ 2 ] = 1, F4[ 1 ][ 2 ] = 2, F4[ 2 ][ 2 ] = 1 (367)

– Otherwise, the following applies:

F3[ 1 ][ 1 ] = 8 (368)

F4[ 0 ][ 1 ] = 2, F4[ 1 ][ 1 ] = 4, F4[ 2 ][ 1 ] = 2, (369)

* 1. The collocated luma samples pY[ x ][ y ] with x = 0..nTbW \* SubWidthC − 1, y= 0..nTbH \* SubHeightC − 1 are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations ( xTbY + x, yTbY + y ).
  2. The neighbouring luma samples pY[ x ][ y ] are derived as follows:
     + When numSampL is greater than 0, the neighbouring left luma samples pY[ x ][ y ] with x = −1..− 3, y = 0..SubHeightC \* numSampL − 1, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations ( xTbY + x , yTbY +y ).
     + When availT is equal to FALSE, the neighbouring top luma samples pY[ x ][ y ] with x = −1..SubWidthC \* numSampT − 1, y = −1..−2, are set equal to the luma samples pY[ x ][ 0 ].
     + When availL is equal to FALSE, the neighbouring left luma samples pY[ x ][ y ] with x = −1..−3, y = −1..SubHeightC \* numSampL − 1, are set equal to the luma samples pY[ 0 ][ y ].
     + When numSampT is greater than 0, the neighbouring top luma samples pY[ x ][ y ] with x = 0..SubWidthC \* numSampT − 1, y = −1, −~~2~~3, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations ( xTbY+ x, yTbY + y ).
     + When availTL is equal to TRUE, the neighbouring top-left luma samples pY[ x ][ y ] with x = −1, y = −1, −2, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations ( xTbY+ x, yTbY + y ).
  3. The down-sampled collocated luma samples pDsY[ x ][ y ] with x = 0..nTbW − 1,  y = 0..nTbH − 1 are derived as follows:
     + If both SubWidthC and SubHeightC are equal to 1, the following applies:
       - pDsY[ x ][ y ] with x = ~~1~~0..nTbW − 1, y = ~~1~~0..nTbH − 1 is derived as follows:

pDstY[ x ][ y ] = pY[ x ][ y ] (360)

* + - Otherwise, the following applies:
      * ~~The one-dimensional filter coefficients array F1 and F2, and the 2-dimensional filter coefficients arrays F3 and F4 are specified as follows.~~

~~F1[ 0 ] = 2, F1[ 1 ] = 0 (361)~~

~~F2[ 0 ] = 1, F2[ 1 ] = 2, F2[ 2 ] = 1 (362)~~

~~F3[ i ][ j ] = F4[ i ][ j ] = 0, with i = 0..2, j = 0..2 (363)~~

* + - * + ~~If both SubWidthC and SubHeightC are equal to 2, the following applies:~~

~~F1[ 0 ] = 1, F1[ 1 ] = 1 (364)~~

~~F3[ 0 ][ 1 ] = 1, F3[ 1 ][ 1 ] = 4, F3[ 2 ][ 1 ] = 1, F3[ 1 ][ 0 ] = 1, F3[ 1 ][ 2 ] = 1 (365)~~

~~F4[ 0 ][ 1 ] = 1, F4[ 1 ][ 1 ] = 2, F4[ 2 ][ 1 ] = 1 (366)~~

~~F4[ 0 ][ 2 ] = 1, F4[ 1 ][ 2 ] = 2, F4[ 2 ][ 2 ] = 1 (367)~~

* + - * + ~~Otherwise, the following applies:~~

~~F3[ 1 ][ 1 ] = 8 (368)~~

~~F4[ 0 ][ 1 ] = 2, F4[ 1 ][ 1 ] = 4, F4[ 2 ][ 1 ] = 2, (369)~~

* + - * If sps\_chroma\_vertical\_collocated\_flag is equal to 1, the following applies:
        + pDsY[ x ][ y ] with x = 0..nTbW − 1, y = 0..nTbH − 1 is derived as follows:

pDsY[ x ][ y ] = ( F3[ 1 ][ 0 ] \* pY[ SubWidthC \* x ][ SubHeightC \* y − 1 ] +  
 F3[ 0 ][ 1 ] \* pY[ SubWidthC \* x − 1 ][ SubHeightC \* y ] +  
 F3[ 1 ][ 1 ] \* pY[ SubWidthC \* x ][ SubHeightC \* y ] + (370)  
 F3[ 2 ][ 1 ] \* pY[ SubWidthC \* x + 1 ][ SubHeightC \* y ] +  
 F3[ 1 ][ 2 ] \* pY[ SubWidthC \* x ][ SubHeightC \* y + 1 ] + 4 ) >> 3

* + - * Otherwise (sps\_chroma\_vertical\_collocated\_flag is equal to 0), the following applies:
        + pDsY[ x ][ y ] with x = 0..nTbW − 1, y = 0..nTbH − 1 is derived as follows:

pDsY[ x ][ y ] = ( F4[ 0 ][ 1 ] \* pY[ SubWidthC \* x − 1 ][ SubHeightC \* y ] +  
 F4[ 0 ][ 2 ] \* pY[ SubWidthC \* x − 1 ][ SubHeightC \* y + 1 ] +  
 F4[ 1 ][ 1 ] \* pY[ SubWidthC \* x ][ SubHeightC \* y ] + (371)  
 F4[ 1 ][ 2 ] \* pY[ SubWidthC \* x ][ SubHeightC \* y + 1] +  
 F4[ 2 ][ 1 ] \* pY[ SubWidthC \* x + 1 ][ SubHeightC \* y ] +  
 F4[ 2 ][ 2 ] \* pY[ SubWidthC \* x + 1][ SubHeightC \* y + 1 ] + 4 ) >> 3

* 1. When numSampL is greater than 0, the selected neighbouring left chroma samples pSelC[ idx ] are set equal to p[ −1 ][ pickPosL[ idx ] ] with idx = 0..cntL − 1, and the selected down-sampled neighbouring left luma samples pSelDsY[ idx ] with idx = 0..cntL − 1 are derived as follows:
     + The variable y is set equal to pickPosL[ idx ].
     + If both SubWidthC and SubHeightC are equal to 1, the following applies:

pSelDsY[ idx ] = pY[ − 1][ y ] (372)

* + - Otherwise the following applies:
      * If sps\_chroma\_vertical\_collocated\_flag is equal to 1, the following applies:

pSelDsY[ idx ] = ( F3[ 1 ][ 0 ] \* pY[ − SubWidthC ][ SubHeightC \* y − 1 ] +  
 F3[ 0 ][ 1 ] \* pY[ −1 − SubWidthC ][ SubHeightC \* y ] +  
 F3[ 1 ][ 1 ] \* pY[ −SubWidthC ][ SubHeightC \* y ]  + (373)  
 F3[ 2 ][ 1 ] \* pY[ 1 − SubWidthC ][ SubHeightC \* y ] +  
 F3[ 1 ][ 2 ] \* pY[ −SubWidthC ][ SubHeightC \* y + 1 ] + 4 ) >> 3

* + - * Otherwise (sps\_chroma\_vertical\_collocated\_flag is equal to 0), the following applies:

pSelDsY[ idx ] = ( F4[ 0 ][ 1 ] \* pY[ −1 − SubWidthC ][ SubHeightC \* y ] +  
 F4[ 0 ][ 2 ] \* pY[ −1 − SubWidthC ][ SubHeightC \* y + 1 ] +  
 F4[ 1 ][ 1 ] \* pY[ −SubWidthC ][ SubHeightC \* y ] + (374)  
 F4[ 1 ][ 2 ] \* pY[ −SubWidthC ][ SubHeightC \* y + 1] +  
 F4[ 2 ][ 1 ] \* pY[ 1 − SubWidthC ][ SubHeightC \* y ] +  
 F4[ 2 ][ 2 ] \* pY[ 1 − SubWidthC][ SubHeightC \* y + 1 ] + 4 ) >> 3

* 1. When numSampT is greater than 0, the selected neighbouring top chroma samples pSelC[ idx ] are set equal to p[ pickPosT[ idx − cntL ] ][ -1 ] with idx = cntL..cntL + cntT − 1, and the down-sampled neighbouring top luma samples pSelDsY[ idx ] with idx = 0..cntL + cntT − 1 are specified as follows:
     + The variable x is set equal to pickPosT[ idx − cntL ].
     + If both SubWidthC and SubHeightC are equal to 1, the following applies:

pSelDsY[ idx ] = pY[ x ][ − 1] (375)

* + - Otherwise, the following applies:
      * + If bCTUboundary is equal to FALSE, the following applies:

If sps\_chroma\_vertical\_collocated\_flag is equal to 1, the following applies:

* + - * + ~~If bCTUboundary is equal to FALSE, the following applies:~~

pSelDsY[ idx ] = ( F3[ 1 ][ 0 ] \* pY[ SubWidthC \* x ][ − 1 − SubHeightC ] +  
 F3[ 0 ][ 1 ] \* pY[ SubWidthC \* x − 1 ][ −SubHeightC ] +  
 F3[ 1 ][ 1 ] \* pY[ SubWidthC \* x ][ −SubHeightC] + (376)  
 F3[ 2 ][ 1 ] \* pY[ SubWidthC \* x + 1 ][ −SubHeightC] +  
 F3[ 1 ][ 2 ] \* pY[ SubWidthC \* x ][ 1 − SubHeightC ] + 4 ) >> 3

* + - * + ~~Otherwise (bCTUboundary is equal to TRUE), the following applies:~~

~~pSelDsY[ idx ] = ( F2[ 0 ] \* pY[ SubWidthC \* x − 1 ][ −1 ] +  
 F2[ 1 ] \* pY[ SubWidthC \* x ][ −1 ]  + (377)  
 F2[ 2 ] \* pY[ SubWidthC \* x + 1 ][ −1 ] + 2 ) >> 2~~

Otherwise (sps\_chroma\_vertical\_collocated\_flag is equal to 0), the following applies:

* + - * + ~~If bCTUboundary is equal to FALSE, the following applies:~~

pSelDsY[ idx ] = ( F4[ 0 ][ 1 ] \* pY[ SubWidthC x − 1 ][ −1 ] +  
 F4[ 0 ][ 2 ] \* pY[ SubWidthC \* x − 1 ][ −2 ] +  
 F4[ 1 ][ 1 ] \* pY[ SubWidthC \* x ][ −1 ] + (378)  
 F4[ 1 ][ 2 ] \* pY[ SubWidthC \* x ][ −2] +  
 F4[ 2 ][ 1 ] \* pY[ SubWidthC \* x + 1 ][ −1 ] +  
 F4[ 2 ][ 2 ] \* pY[ SubWidthC \* x + 1 ][ −2 ] + 4 ) >> 3

* + - * + Otherwise (bCTUboundary is equal to TRUE), the following applies:

pSelDsY[ idx ] = ( F2[ 0 ] \* pY[ SubWidthC \* x − 1 ][ −1 ] +  
 F2[ 1 ] \* pY[ SubWidthC \* x ][ −1 ] + (379)  
 F2[ 2 ] \* pY[ SubWidthC \* x + 1][ −1 ] + 2 ) >> 2