**Draft Text Specification**

The proposed text changes are based on the document JCTVC-Q1005-v1.doc for the IntraBC as inter PU in SCCE1 Test 2.1. The changes are marked in yellow. The changes for IntraBC merge mode in SCCE1 Test 2.3 are marked in green.

**Syntax**

**Coding unit syntax**

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | **Descriptor** |
| if( transquant\_bypass\_enabled\_flag ) |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| if( slice\_type != I ) |  |
| **cu\_skip\_flag**[ x0 ][ y0 ] | ae(v) |
| nCbS = ( 1  <<  log2CbSize ) |  |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else { |  |
| if( slice\_type != I || intra\_block\_copy\_enabled\_flag) |  |
| **pred\_mode\_flag** | ae(v) |
| …… |  |

**Prediction unit syntax**

|  |  |
| --- | --- |
| prediction\_unit( x0, y0, nPbW, nPbH ) { | Descriptor |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { /\* MODE\_INTER \*/ |  |
| **merge\_flag**[ x0 ][ y0 ] | ae(v) |
| if( merge\_flag[ x0 ][ y0 ] ) { |  |
| if(slice\_type != I && intra\_block\_copy\_enabled\_flag) | ae(v) |
| **intra\_bc\_merge\_flag**[ x0 ][ y0 ] |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { |  |

**Semantics**

**7.4.9.6 Prediction unit semantics**

**intra\_bc\_merge\_flag**[ x0 ][ y0 ] equal to 1 specifies that the block vector in current prediction unit is coded in merge mode (merge with the left block vector or the above block vector). intra\_bc\_merge\_flag[ x0 ][ y0 ] equal to 0 specifies that the block vector of current prediction unit is not coded in merge mode. When not present, if slice\_type is equal to I, intra\_block\_copy\_enabled\_flag is equal to 1 and merge\_flag[ x0 ][ y0 ] is equal to 1, the value of intra\_bc\_merge\_flag is inferred to be equal to 1; otherwise, intra\_bc\_merge\_flag is inferred to be equal to 0. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

**ref\_idx\_l0**[ x0 ][ y0 ] specifies the list 0 reference picture index for the current prediction unit. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When ref\_idx\_l0[ x0 ][ y0 ] is not present it is inferred to be equal to 0.

**ref\_idx\_l1**[ x0 ][ y0 ] has the same semantics as ref\_idx\_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

The variable intra\_bc\_flag[ x0 ][ y0 ] is derived as the following:

* If intra\_block\_copy\_enabled\_flag is equal to 1, pred\_mode\_flag is equal to MODE\_INTER, and inter\_pred\_idc[ x0 ][ y0 ] is equal to PRED\_L0, and one of the following conditions is true, intra\_bc\_flag[ x0 ][ y0 ] is set to be 1.
  + When slice\_type is equal to I.
  + When slice\_type is not equal to I and ref\_idx\_l0 [ x0 ][ y0 ] is equal to 1.
  + When slice\_type is not equal to I and intra\_bc\_merge\_flag[ x0 ][ y0 ] equal to 1
* Otherwise, intra\_bc\_flag[ x0 ][ y0 ] is set to be 0.

**Decoding Process**

**6.4.2 Derivation process for prediction block availability**

Inputs to this process are:

* the luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* the luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,
* a variable partIdx specifying the partition index of the current prediction unit within the current coding unit,
* the luma location ( xNbY, yNbY ) covered by a neighbouring prediction block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring prediction block covering the location ( xNbY, yNbY ), denoted as availableN is derived as follows:

……

When availableN is equal to TRUE, ~~CuPredMode[ xNbY ][ yNbY ] is equal to MODE\_INTRA, availableN is set equal to FALSE.~~

–  If CuPredMode[ xPb][ yPb ] is equal to MODE\_INTER, and CuPredMode[ xNbY ][ yNbY ]is equal to MODE\_INTRA, availableN is set equal to FALSE.

– Otherwise, if intra\_bc\_flag[ xPb][ yPb ] is equal to 1, and intra\_bc\_flag[ xNbY ][ yNbY ] is equal to 0, availableN is set equal to FALSE.

**8.5.3.2 Derivation process for motion vector components and reference indices**

……

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1, as well as predFlagL0 and predFlagL1, the following applies:

* If merge\_flag[ xPb ][ yPb ] is equal to 1, and if intra\_bc\_merge\_flag[ xPb ][ yPb ] is equal to 0, the derivation process for luma motion vectors for merge mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1. If merge\_flag[ xPb ][ yPb ] is equal to 1and intra\_bc\_merge\_flag[ xPb ][ yPb ] is equal to 1, the derivation process for luma motion vectors for merge mode as specified in subclause 0 is invoked with the luma location ( xCb, yCb ), the luma location ( xPb, yPb ), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0.
* Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, and refIdxLX, in PRED\_LX, and in the syntax elements ref\_idx\_lX and MvdLX, the following applies:

1. The variables refIdxLX and predFlagLX are derived as follows:

* If inter\_pred\_idc[ xPb ][ yPb ] is equal to PRED\_LX or PRED\_BI,

refIdxLX = (ref\_idx\_lX[ xPb ][ yPb ] != 0 && inter\_pred\_idc[ xPb ][ yPb ] == PRED\_L0) ? ref\_idx\_lX[ xPb ][ yPb ] – 1 : ref\_idx\_lX[ xPb ][ yPb ] (8‑72)

1. …….
2. .……
3. When predFlagLX is equal to 1, the luma motion vector mvLX is derived as follows:

uLX[ 0 ] = ( mvpLX[ 0 ] + mvdLX[ 0 ] + 216 ) % 216 (8‑78)

mvLX[ 0 ] = ( uLX[ 0 ] >= 215 ) ? ( uLX[ 0 ] − 216 ) : uLX[ 0 ] (8‑79)

uLX[ 1 ] = ( mvpLX[ 1 ] + mvdLX[ 1 ] + 216 ) % 216 (8‑80)

mvLX[ 1 ] = ( uLX[ 1 ] >= 215 ) ? ( uLX[ 1 ] − 216 ) : uLX[ 1 ] (8‑81)

NOTE – The resulting values of mvLX[ 0 ] and mvLX[ 1 ] as specified above will always be in the range of −215 to 215 − 1, inclusive.

If intra\_bc\_flag[ xPb ][ yPb ] is equal to 1, the followings apply:

– When the derivation process for z-scan order block availability as specified in subclause   is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + mvL0[ 0 ], yPb + mvL0[ 1 ] ) as inputs, the output shall be equal to TRUE.

– When the derivation process for z-scan order block availability as specified in subclause  is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + mvL0[ 0 ] + nPbW − 1, yPb + mvL0[ 1 ] + nPbH – 1 ) as inputs, the output shall be equal to TRUE.

– One or both of the following conditions shall be true:

– mvL0[ 0 ] + nPbW <= 0

– mvL0[ 1 ] + nPbH <= 0

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in subclause is invoked with mvLX as input, and the output being mvCLX.

##### **8.5.3.2.5 Derivation process for luma motion vector prediction**

…….

The motion vector predictor mvpLX is derived in the following ordered steps:

1. …….
2. If both availableFlagLXA and availableFlagLXB are equal to 1 and mvLXA is not equal to mvLXB, or if intra\_bc\_flag[xPb][yPb] is equal to 1, availableFlagLXCol is set equal to 0. Otherwise, the derivation process for temporal luma motion vector prediction in subclause is invoked with luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and refIdxLX, with X being 0 or 1, as inputs, and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.
3. The motion vector predictor candidate list, mvpListLX, is constructed as follows:

i = 0  
if( availableFlagLXA )  
 mvpListLX[ i++ ] = mvLXA  
if( availableFlagLXB )  
 mvpListLX[ i++ ] = mvLXB (8‑143)  
if( availableFlagLXCol )  
 mvpListLX[ i++ ] = mvLXCol

1. The motion vector predictor list is modified as follows:
   * + When mvLXA and mvLXB have the same value, mvLXB is removed from the list and the variable numMvpCandLX is set equal to the number of elements within the mvpListLX.
     + When numMvpCandLX is less than 2, the following applies repeatedly until numMvpCandLX is equal to 2:

If intra\_bc\_flag[xPb][yPb] is equal to 1

mvpListLX[ numMvpCandLX ][ 0 ] = (numMvpCandLX==0) ? -2\* nCbS : -nCbS (8‑144)

otherwise,

mvpListLX[ numMvpCandLX ][ 0 ] = 0 (8‑144)

mvpListLX[ numMvpCandLX ][ 1 ] = 0 (8‑145)

numMvpCandLX = numMvpCandLX + 1 (8‑146)

* + - When numMvpCandLX is greater than 2, all motion vector predictor candidates mvpListLX[ idx ] with idx greater than 1 are removed from the list.

1. If intra\_bc\_merge\_flag[ xPb ][ yPb ] is equal to 1, mvp\_lX\_flag[ xPb ][ yPb ] is set to be merge\_idx[ xPb ][ yPb ]. The motion vector of mvpListLX[ mvp\_lX\_flag[ xPb ][ yPb ] ] is assigned to mvpLX.

##### **8.5.3.2.6 Derivation process for motion vector predictor candidates**

…….

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. ……
2. ……
3. ……
4. ……
5. ……
6. The following applies for ( xNbAk, yNbAk ) from ( xNbA0, yNbA0 ) to ( xNbA1, yNbA1 ):

* When availableAk is equal to TRUE and availableFlagLXA is equal to 0, the following applies:
* If PredFlagLX[ xNbAk ][ yNbAk ] is equal to 1 and DiffPicOrderCnt( RefPicListX[ RefIdxLX[ xNbAk ][ yNbAk ] ], RefPicListX[ refIdxLX ] ) is equal to 0, or if intra\_bc\_flag[xPb][ yPb ] is equal to 1, availableFlagLXA is set equal to 1 and the following applies:

mvLXA = MvLX[ xNbAk ][ yNbAk ] (8‑147)

* Otherwise, when PredFlagLY[ xNbAk ][ yNbAk ] (with Y = !X) is equal to 1 and DiffPicOrderCnt( RefPicListY[ RefIdxLY[ xNbAk ][ yNbAk ] ], RefPicListX[ refIdxLX ] ) is equal to 0, availableFlagLXA is set equal to 1 and the following applies:

mvLXA = MvLY[ xNbAk ][ yNbAk ] (8‑148)

1. When availableFlagLXA is equal to 0, and intra\_bc\_flag[xPb][yPb] is equal to 0, the following applies for ( xNbAk, yNbAk ) from ( xNbA0, yNbA0 ) to ( xNbA1, yNbA1 ) or until availableFlagLXA is equal to 1:

……

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. The sample locations ( xNbB0, yNbB0 ), ( xNbB1, yNbB1 ), and ( xNbB2, yNbB2 ) are set equal to ( xPb + nPbW, yPb − 1 ), ( xPb + nPbW − 1, yPb − 1 ), and ( xPb − 1, yPb − 1 ), respectively.
2. The availability flag availableFlagLXB is set equal to 0 and the both components of mvLXB are set equal to 0. If intra\_bc\_flag[xPb][yPb] is equal to 1, isScaledFlagLX is set to 1.
3. The following applies for ( xNbBk, yNbBk ) from ( xNbB0, yNbB0 ) to ( xNbB2, yNbB2 ):

* The availability derivation process for a prediction block as specified in subclause  is invoked with the luma location ( xCb, yCb ), the current luma coding block size nCbS, the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location ( xNbY, yNbY ) set equal to ( xNbBk, yNbBk ), and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableBk.
* When availableBk is equal to TRUE and availableFlagLXB is equal to 0, the following applies:
* If PredFlagLX[ xNbBk ][ yNbBk ] is equal to 1, and DiffPicOrderCnt( RefPicListX[ RefIdxLX[ xNbBk ][ yNbBk ] ], RefPicListX[ refIdxLX ] ) is equal to 0, or intra\_bc\_flag[xPb][yPb] is equal to 1, availableFlagLXB is set equal to 1 and the following assignments are made:

mvLXB = MvLX[ xNbBk ][ yNbBk ] (8‑160)

refIdxB = RefIdxLX[ xNbBk ][ yNbBk ] (8‑161)

* Otherwise, when PredFlagLY[ xNbBk ][ yNbBk ] (with Y = !X) is equal to 1 and DiffPicOrderCnt( RefPicListY[ RefIdxLY[ xNbBk ][ yNbBk ] ], RefPicListX[ refIdxLX ] ) is equal to 0, availableFlagLXB is set equal to 1 and the following assignments are made:

mvLXB = MvLY[ xNbBk ][ yNbBk ] (8‑162)

refIdxB = RefIdxLY[ xNbBk ][ yNbBk ] (8‑163)

……

##### **8.5.3.2.1 Derivation process for luma motion vectors for merge mode**

This process is only invoked when merge\_flag[ xPb ][ yPb ] is equal to 1 and when intra\_bc\_merge\_flag[ xPb ][ yPb ] is equal to 0, where ( xPb, yPb ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

……

##### **8.5.3.2.10 Derivation process for luma motion vector for intra block copying merge mode**

This process is only invoked when intra\_bc\_merge\_flag[ xPb ][ yPb ] is equal to 1, where ( xPb, yPb ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

* the luma motion vector bv of this prediction unit

The luma motion vector bv of the prediction unit is derived by invoking the derivation process for luma motion vector prediction specified in subclause with the luma coding block location ( xCb, yCb ), the size of the current luma coding block nCbS,the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, the reference index refIdxL0 and the prediction unit index partIdx as inputs.

##### **8.5.3.3 Decoding process for inter prediction samples**

##### **8.5.3.3.1 General**

……

Let predSamplesL0L and predSamplesL1L be (nPbW)x(nPbH) arrays of predicted luma sample values and when ChromaArrayType is not equal to 0, predSampleL0Cb, predSampleL1Cb, predSampleL0Cr, and predSampleL1Cr be (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays of predicted chroma sample values.

If intra\_bc\_flag[xCb + xBl][ yCb + yBl] is equal to 1,

* The array predSampleL of the prediction samples of luma componentis derived by invoking the intra block copying sample prediction process specified in subclause with the luma coding block location ( xCb, yCb ), the luma prediction block location ( xBl, yBl ), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvL0, chroma component index cIdx=0, and the Luma component of the current picture sample array as inputs.
* when ChromaArrayType is not equal to 0, the array predSampleCb of the prediction samples of component Cb, is derived by invoking the intra block copying sample prediction process specified in subclause with the luma coding block location ( xCb, yCb ), the luma prediction block location ( xBl, yBl ), the prediction block width nPbW/ SubWidthC, the prediction block height nPbH/ SubHeightC, the motion vectors mvL0, chroma component index cIdx=1, and the Cb component of the current picture sample array as inputs.
* when ChromaArrayType is not equal to 0, the array predSampleCr of the prediction samples of component Cr are derived byinvoking the intra block copying sample prediction process specified in subclause with the luma coding block location ( xCb, yCb ), the luma prediction block location ( xBl, yBl ), the prediction block width nPbW/ SubWidthC, the luma prediction block height nPbH/ SubHeightC, the motion vectors mvL0, chroma component index cIdx=2, and the Cr component of the current picture sample array as inputs.

Otherwise, for X being each of 0 and 1, when predFlagLX is equal to 1, the following applies:

– The reference picture consisting of an ordered two-dimensional array refPicLXL of luma samples and when ChromaArrayType is not equal to 0, two ordered two-dimensional arrays refPicLXCb and refPicLXCr of chroma samples is derived by invoking the process specified in subclause  with refIdxLX as input.

– The array predSamplesLXL, and when ChromaArrayType is not equal to 0, the arrays predSamplesLXCb, and predSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause with the luma locations ( xCb, yCb ) and ( xBl, yBl ), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX and, when ChromaArrayType is not equal to 0, mvCLX, and the reference arrays refPicLXL, refPicLXCb, and refPicLXCr as inputs.

If intra\_bc\_flag[xCb + xBl][ yCb + yBl] is equal to 1, weighted sample prediction process does not apply to the prediction sample arrays; otherwise:

The array predSampleL of the prediction samples of luma component is derived by invoking the weighted sample prediction process specified in subclause with the luma prediction block width nPbW, the luma prediction block height nPbH, and the sample arrays predSamplesL0L and predSamplesL1L, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 0 as inputs.

……

##### **8.5.3.3.5 Intra block copying sample prediction process**

Inputs to this process are:

– a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location ( xBl, yBl ) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the prediction block,

– a variable bvIntra specifying the block copying vector,

– a variable cIdx specifying the colour component of the current block.

– a reference sample array refPic

Output of this process is an (nPbW)x(nPbH) array predSamples of the predicted samples

Let ( xPbY, yPbY ) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where xPbY = xCb + xBl and yPbY = yCb + yBl.

The sample location ( xPb, yPb ) specifying the top-left sample of the current prediction block relative to the top-left sample of the current picture is derived as follows:

( xPb, yPb ) = ( cIdx  = =  0 ) ? ( xPbY, yPbY ) : ( xPbY / SubWidthC, yPbY / SubHeightC ) (8‑226)

The variable bv representing the block vector for prediction in full-sample units is derived as follows:

– If cIdx is not equal to 0, and nPbW or nPbH is smaller than 4, the following applies:

– If ChromaArrayType is equal to 1, bv is set equal to bvIntra[ xCb + 4 ][ yCb + 4 ].

– Otherwise, if ChromaArrayType is equal to 2, bv is set equal to bvIntra[ xCb + 4 ][ yCb ].

– bv[ 0 ] = bv[0] >> ( SubWidthC − 1 )

– bv[ 1 ] = bv[1] >> ( SubHeightC − 1 )

– Otherwise, the following applies:

bv[ 0 ] = bvIntra[ xPb ][ yPb ][ 0 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubWidthC ) − 1 ) (8‑227)

bv[ 1 ] = bvIntra[ xPb ][ yPb ][ 1 ] >> ( ( ( cIdx = = 0 ) ? 1 : SubHeightC ) − 1 ) (8‑228)

– The (nPbW)x(nPbH) array of predicted samples, with x = 0..nTbS − 1, and y = 0…nPbH – 1, is derived as follows:

– The reference sample location (xRefCmp, yRefCmp ) is specified by:

( xRefCmp, yRefCmp ) = ( xPb + x + bv[ 0 ], yPb + y + bv[ 1 ] ) (8‑229)

* Each sample at the location ( xRefCmp, yRefCmp ) in refPic is assigned to predSamples[ x ][ y ].

##### **9.3.2.2 Initialization process for context variables**

**Table 9‑4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxTable** | **initType** | | |
| **0** | **1** | **2** |
|  |  |  |  |  |  |
| prediction\_unit( ) | merge\_flag[ ][ ] |  |  | 0 | 1 |
| intra\_bc\_merge\_flag[][] | Table 9-XX | 0 | 1 | 2 |
| merge\_idx[ ][ ] |  |  | 0 | 1 |
| inter\_pred\_idc[ ][ ] |  |  | 0..4 | 5..9 |
| ref\_idx\_l0[ ][ ], ref\_idx\_l1[ ][ ] |  |  | 0..1 | 2..3 |
| mvp\_l0\_flag[ ][ ], mvp\_l1\_flag[ ][ ] |  |  | 0 | 1 |

**Table 9‑XX– Values of initValue for ctxIdx of intra\_bc\_merge\_flag**

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of intra\_bc\_merge\_flag** | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

##### **9.3.3 Binarization process**

##### **9.3.3.1 General**

| **Table 9‑34 – Syntax elements and associated binarizations** | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
|  |  |  |  |
| prediction\_unit( ) | merge\_flag[ ][ ] | FL | cMax = 1 |
| intra\_bc\_merge\_flag | FL | cMax = 1 |
| merge\_idx[ ][ ] | TR | cMax = MaxNumMergeCand − 1, cRiceParam = 0 |
| inter\_pred\_idc[ x0 ][ y0 ] |  | nPbW, nPbH |
| ref\_idx\_l0[ ][ ] | TR | cMax = num\_ref\_idx\_l0\_active\_minus1, cRiceParam = 0 |
| mvp\_l0\_flag[ ][ ] | FL | cMax = 1 |
| ref\_idx\_l1[ ][ ] | TR | cMax = num\_ref\_idx\_l1\_active\_minus1, cRiceParam = 0 |
| mvp\_l1\_flag[ ][ ] | FL | cMax = 1 |

Note: The following part is changed only if 4x4 intraBC PU is suppoted.

##### **9.3.3.6 Binarization process for part\_mode**

**Table 9‑36 – Binarization for part\_mode [Ed. The typesetting of this table is odd.]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CuPredMode** [ xCb ][ yCb ] | **part\_mode** | **PartMode** | **Bin string** | | | |
| log2CbSize >  MinCbLog2SizeY | | log2CbSize  = =  MinCbLog2SizeY | |
| !amp\_enabled\_flag | amp\_enabled\_flag | log2CbSize  = =  3 && ! intra\_block\_copy\_enabled\_flag | log2CbSize > 3 || intra\_block\_copy\_enabled\_flag |
| MODE\_INTRA | 0 | PART\_2Nx2N | - | - | 1 | 1 |
| 1 | PART\_NxN | - | - | 0 | 0 |
| MODE\_INTER | 0 | PART\_2Nx2N | 1 | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 011 | 01 | 01 |
| 2 | PART\_Nx2N | 00 | 001 | 00 | 001 |
| 3 | PART\_NxN | - | - | - | 000 |
| 4 | PART\_2NxnU | - | 0100 | - | - |
| 5 | PART\_2NxnD | - | 0101 | - | - |
| 6 | PART\_nLx2N | - | 0000 | - | - |
| 7 | PART\_nRx2N | - | 0001 | - | - |

##### **9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag**

##### **9.3.4.2.1 General**

**Table 9‑39 – Assignment of ctxInc to syntax elements with context coded bins**

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>= 5** |
|  |  |  |  |  |  |  |
| intra\_bc\_merge\_flag | 0 | na | na | na | na | na |