**Draft Text Specification**

The proposed text changes are based on the document JCTVC-R1005-v3.doc for the PU level IntraBC signalling in CE2 Test 2a. The changes are marked in yellow.

**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( intra\_block\_copy\_enabled\_flag )~~ |  |
| **~~intra\_bc\_flag~~**~~[ x0 ][ y0 ]~~ | ~~ae(v)~~ |
| if( slice\_type != I ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~ | | intra\_block\_copy\_enabled\_flag ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( palette\_enabled\_flag && ChromaArrayType = = 3 &&   CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~) |  |
| **palette\_mode\_flag**[ x0 ][ y0 ] | ae(v) |
| if( palette\_mode\_flag[ x0 ][ y0 ] ) |  |
| palette\_coding( x0, y0, nCbS ) |  |
| else { |  |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | ~~intra\_bc\_flag[ x0 ][ y0 ] | |~~  log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ~~&& !intra\_bc\_flag[ x0 ][ y0 ]~~) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&   log2CbSize >= Log2MinIpcmCbSizeY &&  log2CbSize <= Log2MaxIpcmCbSizeY ) |  |
| … |  |

#### 7.3.8.6 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 {~~if( intra\_bc\_flag[ x0 ][ y0 ] ) /\* Intra BC\*/~~ |  |
| if( intra\_block\_copy\_enabled\_flag && slice\_type != I && ( PartMode < PART\_NxN  | | MinCbLog2SizeY > 3 && PartMode = = PART\_NxN ))) |  |
| **intra\_bc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( intra\_bc\_flag[ x0 ][ y0 ] ) { /\* Intra BC\*/ |  |
| bvd\_coding( x0, y0, 2 ) |  |
| **bvp\_flag**[ x0 ][ y0 ] | ae(v) |
| } else { /\* MODE\_INTER \*/ |  |
| **merge\_flag**[ x0 ][ y0 ] | ae(v) |
| if( merge\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { |  |
| if( slice\_type = = B ) |  |
| **inter\_pred\_idc**[ x0 ][ y0 ] | ae(v) |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L1 ) { |  |
| if( num\_ref\_idx\_l0\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l0**[ x0 ][ y0 ] | ae(v) |
| mvd\_coding( x0, y0, 0 ) |  |
| **mvp\_l0\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L0 ) { |  |
| if( num\_ref\_idx\_l1\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l1**[ x0 ][ y0 ] | ae(v) |
| if( mvd\_l1\_zero\_flag &&   inter\_pred\_idc[ x0 ][ y0 ] = = PRED\_BI ) { |  |
| MvdL1[ x0 ][ y0 ][ 0 ] = 0 |  |
| MvdL1[ x0 ][ y0 ][ 1 ] = 0 |  |
| } else |  |
| mvd\_coding( x0, y0, 1 ) |  |
| **mvp\_l1\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**Semantics**

#### Coding unit semantics

**~~intra\_bc\_flag~~**~~[ x0 ][ y0 ] equal to 1 specifies that the current coding unit is coded in intra block copying mode. intra\_bc\_flag[ x0 ][ y0 ] equal to 0 specifies that the current coding unit is coded according to pred\_mode\_flag. When not present, the value of intra\_bc\_flag is inferred to be equal to 0. The array indices x0 and y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.~~

#### Prediction unit semantics

**intra\_bc\_flag**[ x0 ][ y0 ] equal to 1 specifies that the current prediction unit is coded in intra block copying mode. intra\_bc\_flag[ x0 ][ y0 ] equal to 0 specifies that the current prediction unit is coded according to pred\_mode\_flag. The array indices x0 and 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 not present, the value of intra\_bc\_flag is inferred as follows:

* + if pred\_mode\_flag is equal to 0 and intra\_block\_copy\_enabled\_flag is equal to 1
    - if slice\_type is equal to I, intra\_bc\_flag is set equal to 1;
    - if PartMode is equal to PART\_NxN and MinCbLog2SizeY is equal to 3, intra\_bc\_flag is set equal to 1;
  + otherwise, intra\_bc\_flag is set equal to 0

**Decoding process**

## Decoding process for coding units coded in intra prediction mode

## 8.4.1 General decoding process for coding units coded in intra prediction mode

……

1. The derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xCb, yCb ) as input.

The variables nCbS, nPbSw, and nPbSh are derived as follows:

nCbS = 1  <<  log2CbSize

nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑xx)

nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑xx)

Depending upon PartMode, the variable numPartitions is derived as follows:

* If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.
* Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.
* Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.

The array of block vectors bvIntra is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):

* The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).
* The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).
* The variable yPb is set equal to yCb + nPbSh \* ( blkIdx / 2 )

1. When intra\_bc\_flag[ xPb ][ yPb ] is equal to 1, the derivation process for block vector components in intra block copying prediction mode as specified in subclause 8.4.4 is invoked with the luma location ( xCb, yCb ), ( xPb, yPb ), the luma coding block size nCbS, the luma prediction block width nPbSw, the luma prediction block height nPbSh, and variable log2CbSize as inputs, and the output being bvIntra.
2. If cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the following applies:

……

### Derivation process for block vector components in intra block copying prediction mode

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 sample of the current picture,
* the luma coding block size block nCbS,
* two variables nPbSw and nPbSh specifying the width and the height of the luma prediction block,a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is the (nPbSw)x(nPbSh) array.

~~The variables nCbS, nPbSw, and nPbSh are derived as follows:~~

~~nCbS = 1  <<  log2CbSize~~ (8‑25)

~~nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑25)~~

~~nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑25)~~

The variable BvpIntra[ compIdx ] specifies a block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

~~Depending upon PartMode, the variable numPartitions is derived as follows:~~

~~– If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.~~

~~– Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.~~

~~– Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.~~

~~The array of block vectors bvIntra is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):~~

* ~~The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).~~
* ~~The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).~~
* ~~The variable yPb is set equal to yCb + nPbSh \* ( blkIdx / 2 )~~

1. The following ordered steps apply, for the variable compIdx proceeding over the values 0..1:
2. The variable LastBvIntra[ 0 ][ compIdx ] and LastBvIntra[ 1 ][ compIdx ]specifies the last two block vector predictor. If this process is invoked for the first time for the current coding tree unit, LastBvIntra[ compIdx ] is derived as follows:

LastBvIntra[ 0 ][ 0 ] = −2\* nCbS; LastBvIntra[ 0 ][ 1 ] = 0

LastBvIntra[ 1 ][ 0 ] = −nCbS; LastBvIntra[ 1 ][ 1 ] = 0

~~Depending upon the number of times this process has been invoked for the current coding tree unit,~~ subclause 8.4.4.1 is invoked with the luma coding block location ( xCb, yCb ), the coding block size nCbS, the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbSw, the luma prediction block height nPbSh, the last block vectors LastBvIntra, and the partition index blkIdx as inputs, and the block vector predictor BvpIntra[ xPb ][ yPb ] as the output, and bvIntra[ xPb ][ yPb ][ compIdx ] is set equal to BvdIntra[ xPb ][ yPb ][ compIdx ] + BvpIntra[ xPb ][ yPb ][ compIdx ] [Ed. (GJS): Needs further formatting cleanup.]

### ……

### Decoding process for intra blocks

#### General decoding process for intra blocks

### ……

1. When controlParaACT is not equal to 1, depending upon the value of predModeIntraBc, the following applies:

– When predModeIntraBc is equal to 0, the general intra sample prediction process as specified in subclause 8.4.4.2.1 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the intra prediction mode predModeIntra, the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

– Otherwise (predModeIntraBc is equal to 1 or CuPredMode[ xTb0 ][ yTb0 ] is equal to MODE\_INTER ),

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 is set equal to ( xTb0, yTb0 )

The variables nCbS, nPbSw, and nPbSh are derived as follows:

nCbS = 1  <<  log2TrafoSize

nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑xx)

nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑xx)

Depending upon PartMode, the variable numPartitions is derived as follows:

* If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.
* Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.
* Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.

The array of predSamples is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):

* The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).
* The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).
* The variable yPb is set equal to yCb + nPbSh \* ( blkIdx / 2 )
* If predModeIntraBc is equal to 1, the intra block copying process as specified in subclause  is invoked with the ~~transform~~ prediction block location ~~( xTb0, yTb0 + yTbOffset )~~(xPb, yPb), ~~the transform block size nTbS, the variable trafoDepth,~~ the width and the height of the luma prediction block nPbSw and nPbSh, the variable bvIntra, and the variable cIdx as inputs, and the output is an ~~(nTbS)x(nTbS)~~ (nPbSw)x(nPbSh) array predSamples.

……

### Inter prediction process

……

Depending on the value of PartMode, the following applies:

– If PartMode is equal to PART\_2Nx2N, the decoding process for prediction units in inter prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( 0, 0 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL, the height of the luma prediction block nPbH set equal to nCbSL, and a partition index partIdx set equal to 0 as inputs, and the outputs are an (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, two (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.

– Otherwise, if PartMode is equal to PART\_2NxN, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb + yBl ][ yCb + yBl ] is equal to 0, the decoding process for prediction units in inter prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( 0, 0 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL, the height of the luma prediction block nPbH set equal to nCbSL  >>  1, and a partition index partIdx set equal to 0 as inputs, and the outputs are an (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, two (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.
2. When intra\_bc\_flag[ xCb + yBl ][ yCb + yBl ] is equal to 0, the The decoding process for prediction units in inter prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( 0, nCbSL  >>  1 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL, the height of the luma prediction block nPbH set equal to nCbSL  >>  1, and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, the two modified (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.

– Otherwise, if PartMode is equal to PART\_Nx2N, the following ordered steps apply:

1. When intra\_bc\_flag[ xCb + yBl ][ yCb + yBl ] is equal to 0, the decoding process for prediction units in inter prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( 0, 0 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL  >>  1, the height of the luma prediction block nPbH set equal to nCbSL, and a partition index partIdx set equal to 0 as inputs, and the outputs are an (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, two (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.
2. When intra\_bc\_flag[ xCb + yBl ][ yCb + yBl ] is equal to 0, the decoding process for prediction units in inter prediction mode as specified in subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xBl, yBl ) set equal to ( nCbSL  >>  1, 0 ), the size of the luma coding block nCbSL, the width of the luma prediction block nPbW set equal to nCbSL  >>  1, the height of the luma prediction block nPbH set equal to nCbSL, and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified (nCbSL)x(nCbSL) array predSamplesL and when ChromaArrayType is not equal to 0, the two modified (nCbSwC)x(nCbShC) arrays predSamplesCb and predSamplesCr.

– Otherwise, if PartMode is equal to PART\_2NxnU, the following ordered steps apply:

……

**Parsing process**

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** |
| sao( ) | sao\_merge\_left\_flag sao\_merge\_up\_flag |  | 0 | 1 | 2 |
| sao\_type\_idx\_luma sao\_type\_idx\_chroma |  | 0 | 1 | 2 |
| coding\_quadtree( ) | split\_cu\_flag[ ][ ] |  | 0..2 | 3..5 | 6..8 |
| coding\_unit( ) | cu\_transquant\_bypass\_flag |  | 0 | 1 | 2 |
| cu\_skip\_flag |  |  | 0..2 | 3..5 |
| ~~intra\_bc\_flag[ ][ ]~~ | ~~Table 9‑38~~ | ~~0~~ | ~~1~~ | ~~2~~ |
| palette\_mode\_flag[ ][ ] |  | 0 | 1 | 2 |
| pred\_mode\_flag |  |  | 0 | 1 |
| part\_mode |  | 0 | 1..4 | 5..8 |
| prev\_intra\_luma\_pred\_flag[ ][ ] |  | 0 | 1 | 2 |
| intra\_chroma\_pred\_mode[ ][ ] |  | 0 | 1 | 2 |
| rqt\_root\_cbf |  |  | 0 | 1 |
| cu\_residual\_act\_flag |  | 0 | 1 | 2 |
| prediction\_unit( ) | merge\_flag[ ][ ] |  |  | 0 | 1 |
| merge\_idx[ ][ ] |  |  | 0 | 1 |
| intra\_bc\_flag[ ][ ] | Table 9‑38 | 0 | 1 | 2 |
| inter\_pred\_idc[ ][ ] |  |  | 0..4 | 5..9 |
| ref\_idx\_l0[ ][ ], ref\_idx\_l1[ ][ ] |  |  | 0..1 | 2..3 |
| mvp\_l0\_flag[ ][ ], mvp\_l1\_flag[ ][ ], bvp\_flag[ ][ ] |  |  | 0 | 1 |

| Table 9‑38 – Syntax elements and associated binarizations | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
| coding\_quadtree( ) | split\_cu\_flag[ ][ ] | FL | cMax = 1 |
| coding\_unit( ) | cu\_transquant\_bypass\_flag | FL | cMax = 1 |
| cu\_skip\_flag | FL | cMax = 1 |
| ~~intra\_bc\_flag~~ | ~~FL~~ | ~~cMax = 1~~ |
| palette\_mode\_flag | FL | cMax = 1 |
| …… |  |  |
| palette\_coding( ) | previous\_palette\_entry\_flag[] | FL | cMax = 1 |
| …… |  |  |
| prediction\_unit( ) | merge\_flag[ ][ ] | FL | cMax = 1 |
| merge\_idx[ ][ ] | TR | cMax = MaxNumMergeCand − 1, cRiceParam = 0 |
| intra\_bc\_flag | FL | cMax = 1 |
| 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 |
| bvp\_flag[ ][ ] | FL | cMax = 1 |

Table 9‑40 – Binarization for part\_mode

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