**Draft Text Specification**

The proposed text changes are based on the document JCTVC-P1005-v1.doc for the Intra BC block vector difference coding in SCCE1 Test 3.6. 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 ] ) |  |
| **pred\_mode\_flag** | ae(v) |
| 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 &&   ~~!intra\_bc\_flag[ x0 ][ y0 ] &&~~   log2CbSize >= Log2MinIpcmCbSizeY &&  log2CbSize <= Log2MaxIpcmCbSizeY ) |  |
| **pcm\_flag**[ x0 ][ y0 ] | ae(v) |
| if( pcm\_flag[ x0 ][ y0 ] ) { |  |
| while( !byte\_aligned( ) ) |  |
| **pcm\_alignment\_zero\_bit** | f(1) |
| pcm\_sample( x0, y0, log2CbSize ) |  |
| ~~} else if( intra\_bc\_flag[ x0 ][ y0 ] ) {~~ |  |
| ~~mvd\_coding( x0, y0, 2)~~ |  |
| ~~if( PartMode = = PART\_2NxN )~~ |  |
| ~~mvd\_coding( x0, y0 + ( nCbS / 2 ), 2)~~ |  |
| ~~else if( PartMode = = PART\_Nx2N )~~ |  |
| ~~mvd\_coding( x0 + ( nCbS / 2 ), y0, 2)~~ |  |
| ~~else if( PartMode = = PART\_NxN ) {~~ |  |
| ~~mvd\_coding( x0 + ( nCbS / 2 ), y0, 2)~~ |  |
| ~~mvd\_coding( x0, y0 + ( nCbS / 2 ), 2)~~ |  |
| ~~mvd\_coding( x0 + ( nCbS / 2 ), y0 + ( nCbS / 2 ), 2)~~ |  |
| ~~}~~ |  |
| } else { |  |
| pbOffset = ( PartMode = = PART\_NxN ) ? ( nCbS / 2 ) : nCbS |  |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| **prev\_intra\_luma\_pred\_flag**[ x0 + i ][ y0 + j ] | ae(v) |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| if( prev\_intra\_luma\_pred\_flag[ x0 + i ][ y0 + j ] ) |  |
| **mpm\_idx**[ x0 + i ][ y0 + j ] | ae(v) |
| else |  |
| **rem\_intra\_luma\_pred\_mode**[ x0 + i ][ y0 + j ] | ae(v) |
| if( ChromaArrayType = = 3 ) |  |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( i = 0; i < nCbS; i = i + pbOffset ) |  |
| **intra\_chroma\_pred\_mode**[ x0 + i ][ y0 + j ] | ae(v) |
| else if( ChromaArrayType != 0 ) |  |
| **intra\_chroma\_pred\_mode**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } else { |  |
| if( PartMode = = PART\_2Nx2N ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else if( PartMode = = PART\_2NxN ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS, nCbS / 2 ) |  |
| } else if( PartMode = = PART\_Nx2N ) { |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS ) |  |
| } else if( PartMode = = PART\_2NxnU ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 4 ), nCbS, nCbS \* 3 / 4 ) |  |
| } else if( PartMode = = PART\_2NxnD ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS \* 3 / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS \* 3 / 4 ), nCbS, nCbS / 4 ) |  |
| } else if( PartMode = = PART\_nLx2N ) { |  |
| prediction\_unit( x0, y0, nCbS / 4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 4 ), y0, nCbS \* 3 / 4, nCbS ) |  |
| } else if( PartMode = = PART\_nRx2N ) { |  |
| prediction\_unit( x0, y0, nCbS \* 3 / 4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS \* 3 / 4 ), y0, nCbS / 4, nCbS ) |  |
| } else { /\* PART\_NxN \*/ |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| } |  |
| } |  |
| if( !pcm\_flag[ x0 ][ y0 ] ) { |  |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !( PartMode = = PART\_2Nx2N && merge\_flag[ x0 ][ y0 ] ) | |   ( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && intra\_bc\_flag[ x0 ][ y0 ] ) ) |  |
| **rqt\_root\_cbf** | ae(v) |
| if( rqt\_root\_cbf ) { |  |
| MaxTrafoDepth = ( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ?   ( max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag ) :   max\_transform\_hierarchy\_depth\_inter ) |  |
| transform\_tree( x0, y0, x0, y0, log2CbSize, 0, 0 ) |  |
| } |  |
| } |  |
| } |  |
| } |  |

**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\*/ |  |
| intra\_bc\_bvd\_coding( x0, y0, 2) |  |
| } 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) |
| } |  |
| } |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| intra\_bc\_bvd\_coding ( x0, y0, refList ) { | **Descriptor** |
| **intra\_bc\_abs\_bvd\_greater0\_flag[ 0 ]** | ae(v) |
| **intra\_bc\_abs\_bvd\_greater0\_flag[ 1 ]** | ae(v) |
| if(intra\_bc\_abs\_bvd\_greater0\_flag[ 0 ] ) { |  |
| **intra\_bc\_abs\_bvd\_minus1[ 0 ]** | ae(v) |
| **intra\_bc\_bvd\_sign\_flag[ 0 ]** | ae(v) |
| } |  |
| if(intra\_bc\_abs\_bvd\_greater0\_flag[ 1 ] ) { |  |
| **intra\_bc\_abs\_bvd\_minus1**[ 1 ] | ae(v) |
| **intra\_bc\_bvd\_sign\_flag**[ 1 ] | ae(v) |
| } |  |
| } |  |

**Semantics**

**intra\_bc\_abs\_bvd\_greater0\_flag**[ compIdx ] specifies whether the absolute value of a block vector component difference is greater than 0.

**intra\_bc\_abs\_bvd\_minus1**[ compIdx ] plus 1 specifies the absolute value of a block vector component difference.

When intra\_bc\_abs\_bvd\_minus1[ compIdx ] is not present, it is inferred to be equal to −1.

**intra\_bc\_bvd\_sign\_flag**[ compIdx ] specifies the sign of a block vector component difference as follows:

If intra\_bc\_bvd\_sign\_flag[ compIdx ] is equal to 0, the corresponding block vector component difference has a positive value.

Otherwise (intra\_bc\_bvd\_sign\_flag[ compIdx ] is equal to 1), the corresponding block vector component difference has a negative value.

When intra\_bc\_bvd\_sign\_flag[ compIdx ] is not present, it is inferred to be equal to 0.

The block vector difference BvdIntra [ compIdx ] for compIdx = 0..1 is derived as follows:

BvdIntra [ compIdx ] = intra\_bc\_abs\_bvd\_greater0\_flag [ compIdx ] \*  
 ( intra\_bc\_abs\_bvd\_minus1 [ compIdx ] + 1 ) \* ( 1 − 2 \* intra\_bc\_bvd\_sign\_flag [ compIdx ] )

| **Table 9‑34 – Syntax elements and associated binarizations** | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
| slice\_segment\_data( ) | end\_of\_slice\_segment\_flag | FL | cMax = 1 |
| end\_of\_sub\_stream\_one\_bit | FL | cMax = 1 |
| sao( ) | sao\_merge\_left\_flag | FL | cMax = 1 |
| sao\_merge\_up\_flag | FL | cMax = 1 |
| sao\_type\_idx\_luma | TR | cMax = 2, cRiceParam = 0 |
| sao\_type\_idx\_chroma | TR | cMax = 2, cRiceParam = 0 |
| sao\_offset\_abs[ ][ ][ ][ ] | TR | cMax = ( 1  <<  ( Min( bitDepth, 10 ) − 5 ) ) − 1, cRiceParam = 0 |
| sao\_offset\_sign[ ][ ][ ][ ] | FL | cMax = 1 |
| sao\_band\_position[ ][ ][ ] | FL | cMax = 31 |
| sao\_eo\_class\_luma | FL | cMax = 3 |
| sao\_eo\_class\_chroma | FL | cMax = 3 |
| 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 |
| pred\_mode\_flag | FL | cMax = 1 |
| part\_mode | 9.3.3.5 | ( xCb, yCb ) = ( x0, y0), log2CbSize |
| pcm\_flag[ ][ ] | FL | cMax = 1 |
| prev\_intra\_luma\_pred\_flag[ ][ ] | FL | cMax = 1 |
| mpm\_idx[ ][ ] | TR | cMax = 2, cRiceParam = 0 |
| rem\_intra\_luma\_pred\_mode[ ][ ] | FL | cMax = 31 |
| intra\_chroma\_pred\_mode[ ][ ] | 9.3.3.6 | - |
| rqt\_root\_cbf | FL | cMax = 1 |
| prediction\_unit( ) | merge\_flag[ ][ ] | FL | cMax = 1 |
| merge\_idx[ ][ ] | TR | cMax = MaxNumMergeCand − 1, cRiceParam = 0 |
| inter\_pred\_idc[ x0 ][ y0 ] | 9.3.3.7 | 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 |
| transform\_tree( ) | split\_transform\_flag[ ][ ][ ] | FL | cMax = 1 |
| cbf\_luma[ ][ ][ ] | FL | cMax = 1 |
| cbf\_cb[ ][ ][ ] | FL | cMax = 1 |
| cbf\_cr[ ][ ][ ] | FL | cMax = 1 |
| mvd\_coding( ) | abs\_mvd\_greater0\_flag[ ] | FL | cMax = 1 |
| abs\_mvd\_greater1\_flag[ ] | FL | cMax = 1 |
| abs\_mvd\_minus2[ ] | EG1 | - |
| mvd\_sign\_flag[ ] | FL | cMax = 1 |
| intra\_bc\_bvd\_coding ( ) | intra\_bc\_abs\_bvd\_greater0\_flag[ ] | FL | cMax = 1 |
| intra\_bc\_abs\_bvd\_minus1 [ ] | EG3 | - |
| intra\_bc\_bvd\_sign\_flag [ ] | FL | cMax = 1 |
| transform\_unit( ) | cu\_qp\_delta\_abs | 9.3.3.8 | - |
| cu\_qp\_delta\_sign\_flag | FL | cMax = 1 |
| cu\_chroma\_qp\_adjustment\_flag | FL | cMax = 1 |
| cu\_chroma\_qp\_adjustment\_idc | TR | cMax = chroma\_qp\_adjustment\_table\_size\_minus1, cRiceParam = 0 |
| cross\_comp\_pred( ) | log2\_res\_scale\_abs\_plus1 | TR | cMax = 4, cRiceParam = 0 |
| res\_scale\_sign\_flag | FL | cMax = 1 |
| residual\_coding( ) | transform\_skip\_flag[ ][ ][ ] | FL | cMax = 1 |
| explicit\_rdpcm\_flag[ ][ ][ ] | FL | cMax = 1 |
| explicit\_rdpcm\_dir\_flag[ ][ ][ ] | FL | cMax = 1 |
| last\_sig\_coeff\_x\_prefix | TR | cMax = ( log2TrafoSize << 1 ) − 1, cRiceParam = 0 |
| last\_sig\_coeff\_y\_prefix | TR | cMax = ( log2TrafoSize << 1 ) − 1, cRiceParam = 0 |
| last\_sig\_coeff\_x\_suffix | FL | cMax = ( 1  <<  ( ( last\_sig\_coeff\_x\_prefix  >>  1 ) − 1 ) − 1 ) |
| last\_sig\_coeff\_y\_suffix | FL | cMax = ( 1  <<  ( ( last\_sig\_coeff\_y\_prefix  >>  1 ) − 1 ) − 1 ) |
| coded\_sub\_block\_flag[ ][ ] | FL | cMax = 1 |
| sig\_coeff\_flag[ ][ ] | FL | cMax = 1 |
| coeff\_abs\_level\_greater1\_flag[ ] | FL | cMax = 1 |
| coeff\_abs\_level\_greater2\_flag[ ] | FL | cMax = 1 |
| coeff\_abs\_level\_remaining[ ] | 9.3.3.9 | current sub-block scan index i, baseLevel |
| coeff\_sign\_flag[ ] | FL | cMax = 1 |