#### 7.3.8.5 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 ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] != MODE\_INTRA | | log2CbSize = = MinCbLog2SizeY ) |  |
| **part\_mode** | ae(v) |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&   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 { |  |
| 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 ] ) ) |  |
| **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, 0 ) |  |
| } |  |
| } |  |
| } |  |
| } |  |

#### Transform tree syntax

|  |  |
| --- | --- |
| transform\_tree( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chroma32x32 ) { | Descriptor |
| if( log2TrafoSize <= ( Log2MaxTrafoSize + 1 ) &&  log2TrafoSize > Log2MinTrafoSize && !chroma32x32 &&  trafoDepth < MaxTrafoDepth && !( IntraSplitFlag && ( trafoDepth = = 0 ) ) ) |  |
| **split\_transform\_flag**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( log2TrafoSize > 2 && ChromaArrayType > 0 && (!chroma32x32 || blkIdx == 0) ) { |  |
| if( trafoDepth = = 0 | | cbf\_cb[ xBase ][ yBase ][ trafoDepth − 1 ] ) { |  |
| **cbf\_cb**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( ChromaArrayType = = 2 && ! split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] ) |  |
| **cbf\_cb**[ x0 ][ y0 + ( 1 << ( log2TrafoSize − 1 ) ) ][ trafoDepth ] |  |
| } |  |
| if( trafoDepth = = 0 | | cbf\_cr[ xBase ][ yBase ][ trafoDepth − 1 ] ) { |  |
| **cbf\_cr**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( ChromaArrayType = = 2 && ! split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] ) |  |
| **cbf\_cr**[ x0 ][ y0 + ( 1 << ( log2TrafoSize − 1 ) ) ][ trafoDepth ] |  |
| } |  |
| } |  |
| if( split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] || log2TrafoSize == Log2MaxTrafoSize + 1 ) { |  |
| x1 = x0 + ( 1  <<  ( log2TrafoSize − 1 ) ) |  |
| y1 = y0 + ( 1  <<  ( log2TrafoSize − 1 ) ) |  |
| chroma32x32 = log2TrafoSize == Log2MaxTrafoSize + 1 && !split\_transform\_flag[x0][y0][ trafoDepth ] |  |
| transform\_tree( x0, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 0, chroma32x32 ) |  |
| transform\_tree( x1, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 1, chroma32x32 ) |  |
| transform\_tree( x0, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 2, chroma32x32 ) |  |
| transform\_tree( x1, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 3, chroma32x32 ) |  |
| } else { |  |
| if( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA | | trafoDepth != 0 | |  cbf\_cb[ x0 ][ y0 ][ trafoDepth ] | | cbf\_cr[ x0 ][ y0 ][ trafoDepth ] | |   ( ChromaArrayType = = 2 &&   ( cbf\_cb[ x0 ][ y0 + ( 1 << ( log2TrafoSize − 1 ) ) ][ trafoDepth ] | |   cbf\_cr[ x0 ][ y0 + ( 1 << ( log2TrafoSize − 1 ) ) ][ trafoDepth ] ) ) ) |  |
| **cbf\_luma**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| transform\_unit( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chroma32x32 ) |  |
| } |  |
| } |  |

#### 7.3.8.10 Transform unit syntax

|  |  |
| --- | --- |
| transform\_unit( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx, chroma32x32 ) { | Descriptor |
| log2TrafoSizeC = log2TrafoSize − ( ChromaArrayType = = 3 ? 0 : (1 – chroma32x32) ) |  |
| if( cbf\_luma[ x0 ][ y0 ][ trafoDepth ] | |   cbf\_cb[ x0 ][ y0 ][ trafoDepth ] | |   cbf\_cr[ x0 ][ y0 ][ trafoDepth ] | |   ( ChromaArrayType = = 2 &&   ( cbf\_cb[ x0 ][ y0 + ( 1 << log2TrafoSizeC ) ][ trafoDepth ] | |   cbf\_cr[ x0 ][ y0 + ( 1 << log2TrafoSizeC ) ][ trafoDepth ] ) ) ) { |  |
| if( cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) { |  |
| **cu\_qp\_delta\_abs** | ae(v) |
| if( cu\_qp\_delta\_abs ) |  |
| **cu\_qp\_delta\_sign\_flag** | ae(v) |
| } |  |
| if( cbf\_luma[ x0 ][ y0 ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0, log2TrafoSize, 0 ) |  |
| if( log2TrafoSize > 2 | | ChromaArrayType = = 3 ) { |  |
| for( tIdx = 0; tIdx < ( ChromaArrayType = = 2 ? 2 : 1 ); tIdx++) |  |
| if( cbf\_cb[ x0 ][ y0 + ( tIdx << log2TrafoSizeC ) ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0 + ( tIdx << log2TrafoSizeC ), log2TrafoSizeC, 1 ) |  |
| for( tIdx = 0; tIdx < ( ChromaArrayType = = 2 ? 2 : 1 ); tIdx++) |  |
| if( cbf\_cr[ x0 ][ y0 + ( tIdx << log2TrafoSizeC ) ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0 + ( tIdx << log2TrafoSizeC ), log2TrafoSizeC, 2 ) |  |
| } else if( blkIdx = = 3 ) { |  |
| for( tIdx = 0; tIdx < ( ChromaArrayType = = 2 ? 2 : 1 ); tIdx++) |  |
| if( cbf\_cb[ xBase ][ yBase ][ trafoDepth ] ) |  |
| residual\_coding( xBase, yBase + ( tIdx << log2TrafoSize ), log2TrafoSize, 1 ) |  |
| for( tIdx = 0; tIdx < ( ChromaArrayType = = 2 ? 2 : 1 ); tIdx++) |  |
| if( cbf\_cr[ xBase ][ yBase ][ trafoDepth ] ) |  |
| residual\_coding( xBase, yBase + ( tIdx << log2TrafoSize ), log2TrafoSize, 2 ) |  |
| } |  |
| } |  |
| } |  |

#### 7.4.9.8 Transform tree semantics

**split\_transform\_flag**[ x0 ][ y0 ][ trafoDepth ] specifies whether a block is split ~~into four blocks with half horizontal and half vertical size~~ for the purpose of transform coding. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

The variable interSplitFlag is derived as follows:

* If max\_transform\_hierarchy\_depth\_inter is equal to 0 and CuPredMode[ x0 ][ y0 ] is equal to MODE\_INTER and PartMode is not equal to PART\_2Nx2N and trafoDepth is equal to 0, interSplitFlag is set equal to 1.
* Otherwise, interSplitFlag is set equal to 0.

When split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] is not present, it is inferred as follows:

* If one or more of the following conditions are true, the value of split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] is inferred to be equal to 1:
* log2TrafoSize is greater than Log2MaxTrafoSize + 1
* IntraSplitFlag is equal to 1 and trafoDepth is equal to 0
* interSplitFlag is equal to 1
* Otherwise, the value of split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] is inferred to be equal to 0.