# Removal of Partitioning Derivation Process

I.7.4.9.5 Coding unit semantics

**dbbp\_flag**[ x0 ][ y0 ] equal to 1 specifies that depth based block partition is used for the current coding unit. dbbp\_flag[ x0 ][ y0 ] equal to 0 specifies that depth based block partition is not used for the current coding unit. When not present, the value of dbbp\_flag[ x0 ][ y0 ] is inferred to be equal to 0.

~~When dbbp\_flag[ x0 ][ y0 ] is equal to 1, PartMode is set equal to PART\_2NxN.~~

I.8.5.1 General decoding process for coding units coded in inter prediction mode

The decoding process for coding units coded in inter prediction mode consists of following ordered steps:

* 1. When iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 1, or iv\_res\_pred\_flag[ nuh\_layer\_id ] is equal to 1 or view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 1, following applies:

If DepthFlag is equal to 0 the derivation process for disparity vectors as specified in subclause I.8.5.5 is invoked with the luma locations ( xCb, yCb ), the coding block size nCbSL as inputs.

Otherwise (DepthFlag is equal to 1), the derivation process for disparity vectors for depth layers as specified in subclause I.8.5.6 is invoked with the luma locations ( xCb, yCb ), the coding block size nCbSL as inputs.

* 1. ~~When dbbp\_flag is equal to 1, the derivation process for a modified partitioning mode as specified in subclause I.8.5.7 is invoked with the luma locations ( xCb, yCb ) and the coding block size nCbS~~~~L~~ ~~as the inputs.~~
  2. The inter prediction process as specified in subclause I.8.5.2 is invoked with the luma location ( xCb, yCb ) and the luma coding block size log2CbSize as inputs, and the outputs are three arrays predSamplesL, predSamplesCb, and predSamplesCr.
  3. When dbbp\_flag is equal to 1, the arrays predSamplesL, predSamplesCb, and predSamplesCr are set equal to PredSamplesDbbpL, PredSamplesDbbpCb, and PredSamplesDbbpCr, respectively. [ Ed. (GT): See note in section I.8.5.3.1].

The decoding process for the residual signal of coding units coded in inter prediction mode specified in subclause I.8.5.3.3.8 is invoked with the luma location ( xCb, yCb ) and the luma coding block size log2CbSize as inputs, and the outputs are three arrays resSamplesL, resSamplesCb, and resSamplesCr.

The reconstructed samples of the current coding unit are derived as follows:

The picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the luma coding block location ( xCb, yCb ), the variable nCurrS set equal to nCbSL, the variable cIdx set equal to 0, the (nCbSL)x(nCbSL) array predSamples set equal to predSamplesL, and the (nCbSL)x(nCbSL) array resSamples set equal to resSamplesL as inputs.

The picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the chroma coding block location ( xCb / 2, yCb / 2 ), the variable nCurrS set equal to nCbSC, the variable cIdx set equal to 1, the (nCbSC)x(nCbSC) array predSamples set equal to predSamplesCb, and the (nCbSC)x(nCbSC) array resSamples set equal to resSamplesCb as inputs.

The picture reconstruction process prior to in-loop filtering for a colour component as specified in subclause 8.6.5 is invoked with the chroma coding block location ( xCb / 2, yCb / 2 ), the variable nCurrS set equal to nCbSC, the variable cIdx set equal to 2, the (nCbSC)x(nCbSC) array predSamples set equal to predSamplesCr, and the (nCbSC)x(nCbSC) array resSamples set equal to resSamplesCr as inputs.

**~~I.8.5.7 Derivation process for a modified partitioning 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 variable nCbS specifying the size of the current luma coding block.~~

~~The derivation process for a depth predicted contour pattern as specified in subclause I.8.5.8 is invoked with the sampling interval sampInt equal to 4, the sample location ( xTb, yTb ) equal to ( xCb, yCb ), and the block size nTbS equal to nCbS as inputs, and the output is a binary partition pattern wedgePattern[ x ][ y ].~~

~~For p in the range of 0 to 1, inclusive and i in the range of 0 to 1, inclusive, partSum[ p ][ i ] is set equal to 0.~~

~~for( y = 0; y < nCbS ; y += 4 )  
 for( x = 0; x < nCbS ; x += 4 ) {   
 segFlag = wedgePattern[ x ][ y ]  
 partSum[ 0 ][ ( x < ( nCbS  >>  1 ) ) ? segFlag : !segFlag ]++  
 partSum[ 1 ][ ( y < ( nCbS  >>  1 ) ) ? segFlag : !segFlag  ]++ (I‑288)  
 }~~

~~The variable partFlag is derived as specified in the following:~~

~~partFlag = 0  
 maxPartSum = 0  
 for( p = 0; p < 2; p++ )  
 for( i = 0; i < 2; i++ ) {  
 if( partSum[ p ][ i ] > maxPartSum ) { (I‑289)  
 maxPartSum = partSum[ p ][ i ]  
 partFlag = p  
 }  
 }~~

~~The variables x0 and y0 are derived and the variable PartMode is modified depending on partFlag as specified in the following:~~

~~x0 = partFlag ? xCb : xCb + nCbS / 2~~

~~y0 = partFlag ? yCb + nCbS / 2 : yCb~~

~~PartMode = partFlag ? SIZE\_2NxN : SIZE\_Nx2N~~

~~The following applies:~~

~~MergeIdx[ x0 ][ y0 ] = merge\_idx[ xCb ][ yCb + nCbS / 2 ] (I‑290)~~

~~MergeFlag[ x0 ][ y0°] = merge\_flag[ xCb ][ yCb + nCbS / 2 ] (I‑291)~~

~~InterPredIdc[ x0 ][ y0°] = inter\_pred\_idc[ xCb ][ yCb + nCbS / 2 ] (I‑292)~~

~~For X in the range of 0 to 1, inclusive, the following applies:~~

~~PuRefIdxLX[ x0 ][ y0°] = ref\_idx\_lX[ xCb ][ yCb + nCbS / 2 ] (I‑293)~~

~~MvpLXFlag[ x0 ][ y0°] = mvp\_lX\_flag[ xCb ][ yCb + nCbS / 2 ] (I‑294)~~

~~MvdLX[ x0 ][ y0°] = MvdLX[ xCb ][ yCb + nCbS/2 ] (I‑295)~~

# Signaling of DBBP Flag for non-square Partitioning

I.7.3.8.5 Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { | **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 ) && !predPartModeFlag ) |  |
| **part\_mode** | ae(v) |
| ~~if( depth\_based\_blk\_part\_flag[ nuh\_layer\_id ]   && CuPredMode[ x0 ][ y0 ]  != MODE\_INTRA )~~ |  |
| **~~dbbp\_flag~~**~~[ x0 ][ y0 ]~~ | ~~ae(v)~~ |
| … |  |
| } |  |

|  |  |
| --- | --- |
| cu\_extension( x0 , y0 , log2CbSize ) { | **Descriptor** |
| if ( rpEnableFlag ) |  |
| **iv\_res\_pred\_weight\_idx** | ae(v) |
| if ( icEnableFlag && iv\_res\_pred\_weight\_idx = = 0 ) |  |
| **ic\_flag** | ae(v) |
| if( depth\_based\_blk\_part\_flag[ nuh\_layer\_id ] && PartMode != PART\_NxN   && PartMode != PART\_2Nx2N ) |  |
| **dbbp\_flag**[ x0 ][ y0 ] | ae(v) |
| … |  |
| } |  |