I.7.3.8.5.2 Coding unit extension syntax

|  |  |
| --- | --- |
| cu\_extension( x0 , y0 , log2CbSize ) { | **Descriptor** |
| … |  |
| for( j = 0; j < nCbS; j = j + pbOffset ) |  |
| for( k = 0; k < nCbS; k = k + pbOffset ) |  |
| if( DmmFlag[ x0 + k ][ y0 + j ] | | sdc\_flag[ x0 ][ y0 ] ) { |  |
| if(sdc\_flag[ x0 ][ y0 ] && !DmmFlag[ x0 + k ][ y0 + j ]) |  |
| **seg\_ pred\_flag**[ x0 + k ][ y0 + j ] | ae(v) |
| dcNumSeg = nSegNum[ x0 + k ][ y0 + j ] |  |
| } |  |
| if(( CuPredMode[ x0 ][ y0 ] = = MODE\_INTRA && sdc\_flag[ x0 ][ y0 ] ) || seg\_pred\_flag[ x0 + k ][ y0 + j ]) |  |
| **depth\_dc\_flag**[ x0 + k ][ y0 + j ] | ae(v) |
| if(!seg\_pred\_flag[ x0 + k ][ y0 + j ]) |  |
| dcNumSeg = DmmFlag[ x0 + k ][ y0 + j ] ? 2 : 1 |  |
| … |  |
| } |  |

I.7.4.9.5.2 Coding unit extension semantics

…

**seg\_pred\_flag**[ x0 ][ y0 ] equal to 1 specifies that multiple segmental prediction is applied. seg\_pred\_flag[ x0 ][ y0 ] equal to 0 specifies that multiple segmental prediction is not applied. When not present, seg\_pred\_flag[ x0 ][ y0 ] is inferred to be equal to 0.

A variable nSegNum[ x0 ][ y0 ] is set equal to seg\_pred\_flag [ x0 ][ y0 ] + 1.

**…**

**depth\_dc\_abs**[ x0][ y0 ][ i ], **depth\_dc\_sign\_flag**[ x0 ][ y0 ][ i ]are used to derive DcOffset[ x0 ][ y0 ][ i ]. When not present, the values of depth\_dc\_abs[ x0][ y0 ][ i ] and depth\_dc\_sign\_flag[ x0 ][ y0 ][ i ] are inferred to be equal to 0.

If seg\_pred\_flag[ x0 ][ y0 ] is equal to 0, the variable dcOffsetMin is set equal to 2 – dcNumSeg;

Otherwise (seg\_pred\_flag[ x0 ][ y0 ] is equal to 1), dcOffsetMin is derived as follows

If i < dcNumSeg  - 1, DcOffsetMin = 0;

Otherwise ( i >= dcNumSeg - 1 ), a variable sumOffset is set equal to the sum of DcOffset[ x0 ][ y0 ][ m ] for all m from 0 to dcNumSeg – 2. dcOffsetMin = sumOffset == 0 ? 1 : 0.

The variable DcOffset[ x0 ][ y0 ][ i ] is derived as specified in the following:

DcOffset[ x0 ][ y0 ][ i ] =   
( 1 − 2 \*depth\_dc\_sign\_flag[ x0 ][ y0 ][ i ] ) \* ( depth\_dc\_abs[ x0 ][ y0 ][ i ] + dcOffsetMin ~~− dcNumSeg +2~~ ) (‑37)

I.8.4.4.1 General decoding process for intra blocks

…

5. The general intra sample prediction process as specified in subclause  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.

6. When sdc\_flag is equal to 1 and seg\_pred\_flag[ xCb ][ yCb ] is equal to 1, the derivation process for segmental prediction as specified in subclause I.8.8.2 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the transform block size nTbS, and predSamples as inputs, and the output is the modified predSamples.

…

I.8.5.3.3.5 Full sample interpolation process

…

* The prediction luma sample value predSampleLXL[ xL ][ yL ] is derived as specified in the following:

predSampleLXL[ xL ][ yL ] = refPicLXL[ xIntL ][ yIntL ] (‑190)

When sdc\_flag is equal to 1 and seg\_pred\_flag[ xCb ][ yCb ] is equal to 1, the derivation process for segmental prediction as specified in subclause I.8.8.2 is invoked with the block location ( xCb, yCb ), the block size nPbW, and predSampleLXL as inputs, and the output is the modified predSampleLXL.

…

I.8.5.4.1 General

…

Otherwise (sdc\_flag is equal to 1), if seg\_pred\_flag[ xCb ][ yCb ] is equal to 0, for x in the range of 0 to nCbSL − 1 and y in the range of 0 to nCbSL − 1, resSamplesL[ x ][ y ] is set equal to DcOffset[ xCb ][ yCb ][ 0 ]. Otherwise (seg\_pred\_flag[ xCb ][ yCb ] is equal to 1 ), for x in the range of 0 to nCbSL − 1 and y in the range of 0 to nCbSL − 1, resSamplesL[ x ][ y ] is set equal to 0.

…

I.8.8.2 Derivation process for segmental prediction

This process is only invoked when seg\_pred\_flag[ xCb ][ yCb ] is equal to 1.

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 nTbS specifying the block size,
* a reference sample array refSamples[ x ][ y ], with x, y =0..nTbS − 1.

Output of this process is:

* the modified reference sample array refSamples[ x ][ y ], with x, y =0..nTbS − 1.

A variable predAvg is derived as

predAvg = refSamples[ 0 ][ 0 ]+ refSamples[ 0 ][ nTbS-1 ] + refSamples[ nTbS-1 ][ 0 ]+ refSamples[ nTbS-1 ][ nTbS-1 ];

predAvg = ( predAvg + 2) >>2.

A variable segIdx[ x ][ y ] is set equal to 0 for x = 0 .. nTbS-1, y = 0 .. nTbS -1.

The following applies for x = 0.. nTbS -1, y = 0 .. nTbS-1:

* + If refSamples[ x ][ y ] < predAvg, segIdx[ x ][ y ] is set equal to 0;
  + Otherwise ( refSamples[ x ][ y ] >= predAvg), segIdx[ x ][ y ] is set equal to 1.

A variable sampleCount[ j ][ k ] is set equal to 0 for k = 0 .. (1  <<  BitDepthY ) -1, and j = 0 .. nSegNum[ xCb ][ yCb ] - 1.

A variable mostCount[ j ] is set equal to 0 for j from 0 to nSegNum[ xCb ][ yCb ] – 1.

A variable segPred[ j ] is set equal to 0 for j from 0 to nSegNum[ xCb ][ yCb ] – 1.

For y in the range of 0 to nTbS-1, inclusive, the following applies:

* + For x in the range of 0 to nTbS-1, inclusive, the following applies:
    - j = segIdx[ x ][ y ].
    - sampleCount[ j ][ refSamples [ x ][ y ] ]++.
    - When sampleCount[ j ][ refSamples [ x ][ y ]] > mostCount[ j ], mostCount[ j ] is set equal to sampleCount[ j ][ refSamples [ x ][ y ]] and segPred[ j ] is set equal to refSamples [ x ][ y ].

For a variable k in the range 0 to nSegNum[ xCb ][ yCb ] – 1, the following applies,

A variable recSamp[ k ] is derived as recSamp[ k ] = DltFlag[ nuh\_layer\_id ] ? Idx2DepthValue[Clip1Y( DepthValue2Idx( segPred[ k ]) + DcOffset[ xCb ][ yCb ][ k ])] : Clip1Y( segPred[ k ] + DcOffset[ xCb ][ yCb ][ k ]).

For y in the range of 0 to nTbS-1, inclusive, the following applies:

* + For x in the range of 0 to nTbS-1, inclusive, the following applies:
    - refSamples[ x ][ y ] is set equal to recSamp[ segIdx[ x ][ y ]].

Table ‑12 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxTable** | **initType** | | |
| **0** | **1** | **2** |
| ... |  |  |  |  |
| dim\_not\_present\_flag |  | 0 | 1 | 2 |
| seg\_pred\_flag | Table I-21’ | 0 | 1 | 2 |
| … |  |  |  |  |

Table ‑21’ – Values of initValue for seg\_pred\_flag ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **ctxIdx of** seg\_pred\_flag | | |
| **0** | **1** | **2** |
| **initValue** | 154 | 154 | 154 |

Table ‑21 – Syntax elements and associated binarizations

| **Syntax structure** | **Syntax element** | **Binarization** | |
| --- | --- | --- | --- |
| **Process** | **Input parameters** |
| cu\_extension( ) | … | … | … |
| sdc\_flag | FL | cMax = 1 |
| seg\_pred\_flag | FL | cMax = 1 |
| … |  |  |

Table ‑23 –Assignment of ctxInc to syntax elements with context coded bins

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>=5** |
| ... | .. | .. | … | … | … | … |
| sdc\_flag | 0 | na | na | na | na | na |
| seg\_pred\_flag | 0 | na | na | na | na | na |
| … |  |  |  |  |  |  |