|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  4th Meeting: Incheon, KR, 20–26 Apr. 2013 | Document: JCT3V-D0177\_proposed\_text\_r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE4: Advanced residual prediction for multiview coding** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Li Zhang Ying Chen Xiang Li Marta Karczewicz  5775 Morehouse Drive San Diego, CA 92121 USA | Tel: Email: | 1-858-651-6660 [lizhang@qti.qualcomm.com](mailto:lizhang@qti.qualcomm.com)  1-858-845-6589 [cheny@qti.qualcomm.com](mailto:cheny@qti.qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document provides text for ARP Simp #2 of JCT3V-D0177 based on Draft 3 of 3D-HEVC Test Model Description. All the changes are highlighted in green.

H.7.3.9.1 General Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { | **Descriptor** |
| … |  |
| if ( resPredEnableFlag ) |  |
| **weighting\_factor\_index** | ae(v) |
| if ( icEnableFlag ) |  |
| **ic\_flag** | ae(v) |
| … |  |
| } |  |
| } |  |

**weighting\_factor\_index** equal to 0 specifies that residual prediction is not used. weighting\_factor\_indexunequal to 0 specifies that residual prediction is used. When weighting\_factor\_indexis not present, its value shall be inferred to be equal to 0.

The variable resPredEnableFlag specifying whether weighting\_factor\_index is present in the bitstream is derived as

1. resPredEnableFlag = iv\_res\_pred\_flag[ nuh\_layer\_id ] && TempRefExistInSlice
2. && PredMode != MODE\_INTRA && PartMode = = PART\_2Nx2N (H‑xx)
3. The variable TempRefExistInSlice is derived at slice level and set to 1 if there is at least one reference picture from current view in either reference picture list. Otherwise, it is set to 0.

H.8.5.2.1 Derivation process for motion vector components and reference indices

…

~~If iv\_res\_pred\_flag[ nuh\_layer\_id ] is equal to 1, availableFlagIvMC is equal to 1, and PartMode[ xP ][ yP ] is equal to PART\_2Nx2N, ResPredFlag[ xP ][ yP ] is set equal to 0, otherwise ( iv\_res\_pred\_flag[ nuh\_layer\_id ] is equal to 0 or availableFlagIvMC is equal to 0 or PartMode[ xP ][ yP ] is not equal to PART\_2Nx2N ), ResPredFlag[ xP ][ yP ] is set equal to 0.~~

H.8.5.2.2 Decoding process for inter prediction samples

…

The variable nCSL is set equal to nCS and the variable nCSC is set equal to nCS >> 1.

* If VspModeFlag[ xC + xB ][ yC + yB ] is equal to 0, the following ordered steps apply:
  1. …
  2. For X being replaced by either 0 or 1, the following applies:
     + When predFlagLX is equal to 1, the following applies.
       - …
       - Depending on DepthFlag, the following applies:
         * If DepthFlag is equal to 0, when weighting\_factor\_index is equal to 0 or refIdxLX points to an inter-view reference picture, the arrays predSamplesLXL, predSamplesLXCb, and predSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause 8.5.2.2.2 with the luma locations ( xC, yC ), ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, the motion vectors mvLX, mvCLX, and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr given as input. When weighting\_factor\_index is unequal to 0, and both refIdxLX and RefPicList[ 0 ] point to temporal reference pictures, the following steps are invoked:

Motion vectors are scaled to RefPicListX[0],

tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td (H-xxx)

distScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (H-xxx)

mvLX = Clip3( −32768, 32767, Sign( distScaleFactor \* mvLX ) \*

( ( Abs( distScaleFactor \* mvLX ) + 127 ) >> 8 ) ) (H-xxx)

where td and tb are derived as

td = Clip3( −128, 127, DiffPicOrderCnt( currPic, refPicListX[refIdxLX] ) ) (H-xxx)

tb = Clip3( −128, 127, DiffPicOrderCnt( currPic, RefPicListX[ 0 ] ) )

mvCLX is set equal to mvLX.

the inter-view residual prediction process as specified in subclause H.8.5.2.2.6 is invoked with the luma locations ( xC, yC ), ( xB, yB ), the size of the current luma coding block nCS, the prediction list utilization flag predFlagLX, the reference index 0, the motion vectors mvLX, mvCLX, the disparity vector mvDisp and the arrays predSamplesL, predSamplesCb, and predSamplesCr, and the reference view index refViewIdx set equal to 0, as the inputs and the outputs are modified versions of the arrays predSamplesL, predSamplesCb, and predSamplesCr.

* + - * + Otherwise, ...
  + …

~~When ResPredFlag[ xC + xB ][ yC + yB ] is equal to 1, the inter-view residual prediction process as specified in subclause H.8.5.2.2.6 is invoked with the luma locations ( xC, yC ) and ( xC + xB, yC + yB ), the size of the current luma coding block nCS, the variables nPSW and nPSH, and the arrays predSamples~~~~L~~~~, predSamples~~~~Cb~~~~, and predSamples~~~~Cr~~ ~~as the inputs and the outputs are modified versions of the arrays predSamples~~~~L~~~~, predSamples~~~~Cb~~~~, and predSamples~~~~Cr~~~~.~~

H.8.5.2.2.6 Inter-view residual prediction process

The process is only invoked if weight\_factor\_index ~~res\_pred\_flag~~ is unequal to 0 ~~1~~.

Inputs to this process are:

* a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top left luma sample of the current picture,
* a luma location ( xP, yP ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* ~~variables nPSW and nPSH specifying the width and the height, respectively, of the current prediction unit,~~prediction list utilization flag~~s, predFlagL0~~predFlagLX ~~and predFlagL1~~, reference index refIdxLX, luma motion vector mvLX, chroma motion vector mvCLX, and reference view index refViewIdx,
* a (nPSW)x(nPSH) array predSamplesL of luma prediction samples,
* two (nPSW / 2)x(nPSH / 2) arrays predSamplesCb and predSamplesCr of chroma prediction samples.

Output of this process are:

* …

The derivation process for a disparity vector as specified in subclause H.8.5.4 is invoked with the luma locations ( xC, yC ) and ( xP, yP ), the coding block size nCS, the variables nPSW and nPSH, the partition index equal to 0 ~~partIdx~~ and the variable deriveFromDepthFlag being equal to 0, as the inputs and the outputs are the view order index refViewIdx, the flag availableDV and the disparity vector mvDisp~~.[Ed. (GT) partIdx is missing as input to this subclause.].~~

~~[Ed. (GT). In software refViewIdx is set equal to 0. Might be better to reuse disparity and refViewIdx from inter-view motion vector prediction. ].~~

~~Let refResSamples~~~~L~~ ~~be the (PicWidthInSamples~~~~L~~~~)x(PicHeightInSamples~~~~L~~~~) array of luma residual samples ResSamples~~~~L:~~ ~~of the view component with ViewIdx equal to refViewIdx. Let refResSamples~~~~Cb~~ ~~and refResSamples~~~~Cr~~ ~~be the (PicWidthInSamples~~~~L~~~~/ 2)x(PicHeightInSamples~~~~L~~~~/ 2) arrays of Cb and Cr residual samples ResSamples~~~~Cb~~ ~~and ResSamples~~~~Cr~~~~, respectively, for inter-coded coding units for the view component with ViewIdx equal to refViewIdx.~~

~~When the flag availableDV is equal to 0 the whole decoding process of this sub-clause terminates.~~

~~For y proceeding over the values 0..(nPSH – 1) and x proceeding over the values 0..(nPSW – 1), the following ordered steps apply.~~

* 1. ~~The variables xR0, xR1, yR0, yR1 and w0, w1, w2, and w3 are derived by~~
     + 1. ~~xR0 = Clip3( 0, PicWidthInSamples~~~~L~~~~– 1, xP + x + (mvDisp[ 0 ] >> 2 ) ) (H‑233)  
          xR1 = Clip3( 0, PicWidthInSamples~~~~L~~~~– 1, xP + x + (mvDisp[ 0 ] >> 2 ) + 1 ) (H‑234)  
          yR0 = Clip3( 0, PicHeightInSamples~~~~L~~~~– 1, yP + y + (mvDisp[ 1 ] >> 2 ) ) (H‑235)  
          yR1 = Clip3( 0, PicHeightInSamples~~~~L~~~~– 1, yP + y + (mvDisp[ 1 ] >> 2 ) + 1 ) (H‑236)  
          w0 = 4 – mvDisp[ 0 ] + ( ( mvDisp[ 0 ] >> 2 ) << 2 ) (H‑237)  
          w1 = mvDisp[ 0 ] − ( ( mvDisp[ 0 ] >> 2 ) << 2 ) (H‑238)  
          w2 = 4 – mvDisp[ 1 ] + ( ( mvDisp[ 1 ] >> 2 ) << 2 ) (H‑239)  
          w3 = mvDisp[ 1 ] − ( ( mvDisp[ 1 ] >> 2 ) << 2 ) (H‑240)~~
  2. ~~The sample predSamples~~~~L~~~~[ x ][ y ] is modified by~~
     + 1. ~~delta~~~~L1~~ ~~= ( w0 \* refResSamples~~~~L~~~~[ xR0 ][ yR0 ] + w1 \* refResSamples~~~~L~~~~[ xR1 ][ yR0 ] + 4 ) >> 3 (H‑241)  
          delta~~~~L2~~ ~~= ( w0 \* refResSamples~~~~L~~~~[ xR0 ][ yR1 ] + w1 \* refResSamples~~~~L~~~~[ xR1 ][ yR1 ] + 4 ) >> 3 (H‑242)  
          delta~~~~L~~ ~~= ( w2 \* delta~~~~L1~~ ~~+ w3 \* delta~~~~L2~~ ~~+ 4 ) >> 3 (H‑243)  
          predSamples~~~~L~~~~[ x ][ y ] = predSamples~~~~L~~~~[ x ][ y ] + delta~~~~L~~ ~~(H‑244)~~

~~For y proceeding over the values 0..(nPSH / 2 – 1) and x proceeding over the values 0..(nPSW / 2 – 1), the following ordered steps are specified:~~

* 1. ~~The variables xR0, xR1, yR0, yR1 and w0, w1, w2, and w3 are derived by~~
     + 1. ~~xR0 = Clip3( 0, PicWidthInSamples~~~~L~~~~/ 2 – 1, xP / 2 + x + (mvDisp[ 0 ] >> 3 ) ) (H‑245)  
          xR1 = Clip3( 0, PicWidthInSamples~~~~L~~~~/ 2 – 1, xP / 2 + x + (mvDisp[ 0 ] >> 3 ) + 1 ) (H‑246)  
          yR0 = Clip3( 0, PicHeightInSamples~~~~L~~~~/ 2 – 1, yP / 2 + y + (mvDisp[ 1 ] >> 3 ) ) (H‑247)  
          yR1 = Clip3( 0, PicHeightInSamples~~~~L~~~~/ 2 – 1, yP / 2 + y + (mvDisp[ 1 ] >> 3 ) + 1 ) (H‑248)  
          w0 = 8 – mvDisp[ 0 ] + ( (mvDisp[ 0 ] >> 3 ) << 3 ) (H‑249)  
          w1 = mvDisp[ 0 ] − ( (mvDisp[ 0 ] >> 3 ) << 3 ) (H‑250)  
          w2 = 8 – mvDisp[ 1 ] + ( (mvDisp[ 1 ] >> 3 ) << 3 ) (H‑251)  
          w3 = mvDisp[ 1 ] − ( (mvDisp[ 1 ] >> 3 ) << 3 ) (H‑252)~~
  2. ~~The sample predSamples~~~~Cb~~~~[ x ][ y ] is modified by~~
     + 1. ~~delta~~~~Cb1~~ ~~= ( w0 \* refResSamples~~~~Cb~~~~[ xR0 ][ yR0 ] + w1 \* refResSamples~~~~Cb~~~~[ xR1 ][ yR0 ] + 8 ) >> 4 (H‑253)  
          delta~~~~Cb2~~ ~~= ( w0 \* refResSamples~~~~Cb~~~~[ xR0 ][ yR1 ] + w1 \* refResSamples~~~~Cb~~~~[ xR1 ][ yR1 ] + 8 ) >> 4 (H‑254)  
          delta~~~~Cb~~  ~~= ( w2 \* delta~~~~Cb1~~ ~~+ w3 \* delta~~~~Cb2~~ ~~+ 8 ) >> 4 (H‑255)  
          predSamples~~~~Cb~~~~[ x ][ y ] = predSamples~~~~Cb~~~~[ x ][ y ] + delta~~~~Cb~~  ~~(H‑256)~~
  3. ~~The sample predSamples~~~~Cr~~~~[ x ][ y ] is modified by~~
     + 1. ~~delta~~~~Cr1~~ ~~= ( w0 \* refResSamples~~~~Cr~~~~[ xR0 ][ yR0 ] + w1 \* refResSamples~~~~Cr~~~~[ xR1 ][ yR0 ] + 8 ) >> 4 (H‑257)  
          delta~~~~Cr2~~ ~~= ( w0 \* refResSamples~~~~Cr~~~~[ xR0 ][ yR1 ] + w1 \* refResSamples~~~~Cr~~~~[ xR1 ][ yR1 ] + 8 ) >> 4 (H‑258)  
          delta~~~~Cr~~ ~~= ( w2 \* delta~~~~Cr1~~~~+ w3 \* delta~~~~Cr2~~~~) + 8 ) >> 4 (H‑259)  
          predSamples~~~~Cr~~~~[ x ][ y ] = predSamples~~~~Cr~~~~[ x ][ y ] + delta~~~~Cr~~  ~~(H‑260)~~

The arrays predSamplesLXL, predSamplesLXCb, and predSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause H.8.5.2.2.6.1 with the luma locations ( xC, yC ), ( xB, yB ), the width and the height of the current luma prediction block nPbW, nPbH, the motion vectors mvLX, mvCLX, and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr (correspond to decoded sample arrays SL, SCb, SCr of RefPicListX[ 0 ]) given as input.

The following apply to derive the prediction values in the reference view:

* Set the reference picture sample arrays refPicLXL, refPicLXCb, and refPicLXCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause 8.7 for a previously-decoded picture which has the same POC value of current picture in the view component with ViewIdx equal to refViewIdx. The arrays refSamplesLXL, refSamplesLXCb, and refSamplesLXCr are derived by invoking the fractional sample interpolation process in subclause H.8.5.2.2.6.1 specified with the luma locations ( xC, yC ), ( xB, yB ), the width and the width nCS and height nCS, the motion vectors mvDisp and mvDisp, and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr given as input.
* Set refPicLXL, refPicLXCb, and refPicLXCr corresponding to decoded sample arrays SL, SCb, SCr derived in subclause 8.7 for a previously-decoded picture which has the same POC value of RefPicListX[ refIdxLX ] in the view component with ViewIdx equal to refViewIdx. The arrays refPredSamplesLXL, refPredSamplesLXCb, and refPredSamplesLXCr are derived by invoking the fractional sample interpolation process specified in subclause H.8.5.2.2.6.1 with the luma locations ( xC, yC ), ( xB, yB ), the width and the height nCS, nCS, the motion vectors (mvLX[0]+ mvDisp[0], mvLX[1]+ mvDisp[1]), (mvLX[0]+ mvDisp[0], mvLX[1]+ mvDisp[1]) and the reference arrays with refPicLXL, refPicLXCb and refPicLXCr given as input.

The modified prediction samples predSamplescom[ x ][ y ] with x = 0..(nPbW)−1 and y = 0..(nPbH)−1 are derived as follows:

predSamplescom[ x ][ y ] = predSamplescom[ x ][ y ] + ( refSamplesLXcom[ x ][ y ] - refPredSamplesLXcom[ x ][ y ] ) >> shiftVal ) (H‑xx)

where com could be replaced by L, Cb, or Cr. When com is unequal to L, nPbW and nPbH are replaced by nPbW / 2 and nPbH / 2, repsectively. shiftVal is set to 1 or 0 corresponding to weight\_factor\_index equal or unequal to 2.

H.8.5.2.2.6.1 Fractional sample interpolation process

The same process as defined in HEVC subclause 8.5.3.2.2 is invoked with ‘8.5.3.2.2.1’and ‘8.5.3.2.2.2’ replaced by ‘H.8.5.2.2.6.2’.

H.8.5.2.2.6.2 Bilinear interpolation

Inputs to this process are:

– a luma/chroma location in full-sample units ( xIntCOM, yIntCOM ),

– a luma/chroma location offset in fractional-sample units ( xFracCOM, yFracCOM ),

– luma/chroma component samples from the selected reference picture refPicLXCOM.

Output of this process is a predicted chroma sample value predPartLXCOM[ xC, yC ].

Let the variable COM denotes the component of one picture with its value equal to L for luma and C for chroma component, respectively.

In Figure xx, the positions labelled with A, B, C, and D represent COM samples at full-sample locations inside the given two-dimensional array refPicLXCOM of chroma samples. When COM is equal to L, xFracCOM and yFracCOM are multiplied by 2.



Figure xx – Fractional sample position dependent variables in bi-linear interpolation and surrounding integer position samples A, B, C, and D

The sample coordinates specified in the following equations are used for generating the predicted COM sample value predPartLXCOM[ xCOM, yCOM ].

xAC = Clip3( 0, PicWidthInSamplesCOM − 1, xIntCOM ) (x)  
xBC = Clip3( 0, PicWidthInSamplesCOM − 1, xIntCOM + 1 ) (x)xCC = Clip3( 0, PicWidthInSamplesCOM − 1, xIntCOM ) (x)xDC = Clip3( 0, PicWidthInSamplesCOM − 1, xIntCOM + 1 ) (x)

yAC = Clip3( 0, refPicHeightEffectiveCOM − 1, yIntCOM ) (x)  
yBC = Clip3( 0, refPicHeightEffectiveCOM − 1, yIntCOM ) (x)yCC = Clip3( 0, refPicHeightEffectiveCOM − 1, yIntCOM + 1 ) (x)yDC = Clip3( 0, refPicHeightEffectiveCOM − 1, yIntCOM + 1 ) (x)

Given the COM samples A, B, C, and D at full-sample locations specified in above equations, the predicted COM sample value predPartLXCOM[ xC, yC ] is derived as:

predPartLXCOM[ xCOM, yCOM ] = ( ( 8 − xFracCOM ) \* ( 8 − yFracCOM ) \* A + xFracCOM \* ( 8 − yFracCOM ) \* B +  
 ( 8 − xFracC ) \* yFracCOM \* C + xFracCOM \* yFracCOM \* D ) >> 6 (x)

H.9.3.1.1 Initialization process for context variables

Table H 13 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| coding\_unit() | **Syntax element** | **ctxIdxTable** | **initType** | | |
| **0** | **1** | **2** |
| dmm\_dc\_1\_abs dmm\_dc\_2\_abs | Table G‑11 | 0 | 1 | 2 |
| weighting\_factor\_index | Table H‑12 |  | 0..3 | 4..7 |
| … | … |  |  |  |

Table H‑xx – Values of variable initValue for weighting\_factor\_index ctxIdx

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Initialization variable** | weighting\_factor\_index | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **initValue** | 154 | 154 | 154 | 154 | 154 | 154 | 154 | 154 |

H.9.3.2 Binarization process

Table H-27 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

| **Syntax element** | **initType** | **Type of binarization** | **maxBinIdxCtx** | **ctxIdxTable** | **ctxIdxOffset** |
| --- | --- | --- | --- | --- | --- |
| **…** |  |  |  |  |  |
| weighting\_factor\_index | 1 | TU, cMax = 2 | 0 | Table G‑12 | 0 |
| 2 | 1 | Table G‑12 | 4 |

H.9.3.3.1.1 Derivation process of ctxIdxInc using left and above syntax elements

Table H‑31 – Specification of ctxIdxInc using left and above syntax elements

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **condL** | **condA** | **ctxIdxInc** |
| sdc\_flag | sdc\_flag [ xL ][ yL ] | sdc\_flag[ xA ][ yA ] | ( condL && availableL ) + ( condA && availableA ) |
| weighting\_factor\_index | weighting\_factor\_index [ xL ][ yL ] | weighting\_factor\_index [ xA ][ yA ] | ( condL && availableL ) + ( condA && availableA ) |