|  |  |
| --- | --- |
| **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-D0116 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Proposed text for JCT3V-Dxxxx** | | |
| *Status:* | Input document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Jin Young Lee, Min Woo Park, Chanyul Kim (Samsung Electronics Co. Ltd.)  Igor Kovliga, Alexey Fartukov, Mikhail Mishurovskiy (Samsung R&D Institute Russia) | Tel: Email: | [jinyoung79.lee@samsung.com](mailto:jinyoung79.lee@samsung.com) [i.kovliga@samsung.com](mailto:i.kovliga@samsung.com) |
| *Source:* | Samsung Electronics Co. Ltd. and Samsung R&D Institute Russia | | |

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

*In 7.3.5,Macroblock layer syntax, replace the syntax table with:*

|  |  |  |
| --- | --- | --- |
| macroblock\_layer( ) { | **C** | **Descriptor** |
| if ( IvmpEnabledFlag ) |  |  |
| **mb\_ivmp\_flag** |  | u(1) | ae(v) |
| if (!mb\_ivmp\_flag) { |  |  |
| if( DepthFlag && disp\_flag ) |  |  |
| **mb\_disp\_flag** |  | u(1) | ae(v) |
| if(!mb\_disp\_flag){ |  |  |
| **mb\_type** | 2 | ue(v) | ae(v) |
| if( nal\_unit\_type = = 21 && !DepthFlag   && slice\_type = = B   && direct\_spatial\_mv\_pred\_flag && VspRefExist  && mb\_type = = B\_Direct\_16x16 ) |  |  |
| **mb\_direct\_type\_flag** | 2 | u(1) | ae(v) |
| if( alc\_sps\_enable\_flag && nal\_unit\_type = = 21 &&  slice\_type = = P && !DepthFlag &&  ( mb\_type = = P\_L0\_16x16 ||   mb\_type = = P\_L0\_L0\_16x8 ||  mb\_type = = P\_L0\_L0\_8x16) |  |  |
| **mb\_alc\_flag** | 2 | u(1) | ae(v) |
| } |  |  |
| } |  |  |
| … |  |  |

**mb\_alc\_flag** equal to 1 specifies that the adaptive luminance compensation mode is in use for the current macroblock. mb\_alc\_flag equal to 0 specifies that the adaptive luminance compensation mode is not in use for the current macroblock. When mb\_alc\_flag is not present, it is inferred to be equal to 0. When mb\_alc\_flag is equal to 1, the current macroblock shall be coded as P\_L0\_16x16, P\_L0\_L0\_16x8, P\_L0\_L0\_8x16.

J.8.3 3DVC inter prediction, inter-view prediction, view synthesis prediction and adaptive luminance compensation

Processes described in this subclause are invoked when decoding P and B macroblock types, and when nal\_unit\_type is equal to 21.

The variable log2Div is specified to be equal to BitDepthY + 6.

The function Disparity( depthSample, srcIndex, refIndex ) is specified to return disparityValue specified as follows.

disparityValue = ( NdrInverse[ depthSample ] \* DisparityScale[ dps\_id ][ srcIndex ][ refIndex ] +   
 ( DisparityOffset[ dps\_id ][ srcIndex ][ refIndex ] << BitDepthY ) +   
 ( 1 << ( log2Div – 1 ) ) ) >> log2Div

When DepthFlag is equal to 0 and dmvp\_flag is equal to 1, the variables DepthRefPicList0, DepthRefPicList1 for B slices, and DepthCurrPic are specified as follows. The variable DepthRefPicList0 is specified to consist of the depth view components of the view component pairs for which the texture view components are in RefPicList0 in the order that RefPicList0[ i ] and DepthRefPicList0[ i ] form a view component pair for any value of i = 0.. num\_ref\_idx\_l0\_active\_minus1. The variable DepthRefPicList1 is specified for B slices to consist of the depth view components of the view component pairs for which the texture view components are in RefPicList1 in the order that RefPicList1[ i ] and DepthRefPicList1[ i ] form a view component pair for any value of i = 0.. num\_ref\_idx\_l1\_active\_minus1. The variable DepthCurrPic is specified to be the upsampled decoded sample array of the depth view component of the view component pair for which the texture view component is the current texture view component.

The specifications in subclause 8.4 apply with the following changes.

– if nal\_unit\_type is equal to 21, DepthFlag is equal to 0, and dmvp\_flag is equal to 1, the following applies:

– If mb\_alc\_skip\_flag is equal to 0 and mb\_alc\_flag is equal to 0, subclause J.8.3.1 is invoked instead of subclause 8.4.1.

– If mb\_alc\_skip\_flag is equal to 1 or mb\_alc\_flag is equal to 1, subclause J.8.3.5 is invoked instead of subclause 8.4.1 for the derivation of motion vector components and reference indices, and subclause J.8.3.6 is invoked instead of subclause 8.4.3 for the derivation of prediction weights. [Ed.(MH): “subclause J.8.3.6 is invoked instead of subclause 8.4.3 for the derivation of prediction weights” should be moved further down in the logic of subclause 8.4, that is between steps 3 and 4 in subclause 8.4.]

– Otherwise, if nal\_unit\_type is equal to 21, DepthFlag is equal to 0, dmvp\_flag is equal to 0, and mb\_alc\_skip\_flag or mb\_alc\_flag is equal to 1, subclause J.8.3.5 is invoked instead of subclause 8.4.1 for the derivation of motion vector components and reference indices, and subclause J.8.3.6 is invoked instead of subclause 8.4.3 for the derivation of prediction weights. [Ed.(MH): “subclause J.8.3.6 is invoked instead of subclause 8.4.3 for the derivation of prediction weights” should be moved further down in the logic of subclause 8.4, that is between steps 3 and 4 in subclause 8.4.]

– When nal\_unit\_type is equal to 21, DepthFlag is equal to 1, and mb\_ivmp\_flag is equal to 1, subclause J.8.3.3 is invoked instead of subclause 8.4.1.

– The following additional step applied between steps 3 and 4 in subclause 8.4.

– When (nal\_unit\_type is equal to 21 and depth\_weighted\_pred\_flag is equal to 1 and (slice\_type % 5) is equal to 0 or 3) or (depth\_weighted\_bipred\_flag is equal to 1 and (slice\_type % 5) is equal to 1), the derivation process for prediction weights in depth-range-based weighted prediction in subclause J.8.3.4 is invoked.

– The following sentence “The decoding process for Inter prediction samples as specified in subclause 8.4.2 is invoked.” in step 4 of subclause 8.4 is replaced by “The decoding process for Inter or view synthesis prediction samples as specified in subclause J.8.3.2 is invoked.”

J.8.3.6 Derivation process for prediction weights in adaptive luminance compensation

Inputs to this process are:

– reference index refIdxL0

– the luma sample array of the selected reference picture refPicL0L.

– the current partition given by its partition index mbPartIdx and its sub-macroblock partition index subMbPartIdx

– Luma4x4BlkIdx

– the width and height partWidth, partHeight of this partition in luma-sample units

– a luma motion vector mvL0 given in quarter-luma-sample units

– array cSL containing already constructed luma samples prior to deblocking filter process.

Outputs of this process are:

– variables for weighted prediction of the current partition logWDC, W0C, W1C,O0C, O1C, with C being replaced by L and, when ChromaArrayType is not equal to 0, Cb and Cr. [Ed. Some subscripts are incorrectly used or missing in this subclause and its siblings. For example W0C should be W0C.]

The variables W1C , O1C are derived as follows for C is equal to L, Cb or Cr:

W1C = 0

O1C = 0

The variables W0C , O0C are derived as follows for C if equal to Cb or Cr:

logWDC = 15

W0C = 1

O0C=0.

When C is equal to L for luma samples, subclauses J.8.3.6.1 through J.8.3.6.5 are invoked sequentially to derive LogWDL, W0C, and O0C.

J.8.3.6.1 Defining of coordinates and sizes of a luma block to be predicted

Let ( xM, yM ) be equal to the output of the subclause 6.4.1 (the location of upper-left luma sample for the current macroblock with address mbAddr relative to the upper-left sample of the picture).

Let ( xP, yP ) be equal to the output of the subclause 6.4.2.1 (the location of upper-left luma sample for the macroblock partition mbPartIdx).

Let ( xB, yB ) be equal to the output of the subclause 6.4.2.2 (the location of upper-left luma sample for the 4x4 luma block defined by Luma4x4BlkIdx that can be 0...15) relative to the top-left sample of the sub-macroblock.

The vairables xT, yT, xBlockWidth, yBlockHeight are set as follows:

– xT is set equal to xM + xP;

– yT is set equal to yM +yP;

– xBlockWidth is set equal to MbPartWidth( mb\_type );

– yBlockHeight is set equal to MbPartHeight( mb\_type );

If one or more of the following conditions are true, W0C is set to 1 and logWDC is set to 15.

– (mvL0[ 0 ] +((xT– 1)<<2)) is smaller than 0

– (mvL0[ 1 ] +((yT– 1)<<2)) is smaller than 0

– (mvL0[ 0 ] +((xT + xBlockWidth)<<2)) is larger or equal to (PicWidthInSamplesL<<2);

– (mvL0[ 1 ] +((yT + yBlockHeight)<<2)) is larger or equal to (PicHeightInSamplesL<<2).

Otherwise LRef, URef, LRec, URec sample values are derived as it is specified in J.8.3.6.2 and J.8.3.6.3 followed by calculation of variables NeighborRefSum, NeighborSum and W0L, O0L specified in the subclause J.8.3.6.4 and J.8.3.6.5 correspondently.

J.8.3.6.2 Deriving of left and up reference samples of the current block

LRec and URec blocks belong to an (PicWidthInSamplesL)x(PicHeightInSamplesL) array cSL containing constructed luma samples prior to the deblocking filter process.

Each luma sample LRec[ 0, yL ] (0<=yL< yBlockHeight) is defined as follows:

LRec[ 0, yL ] = cSL[ xT – 1, yT + yL ]

Each luma sample URec[ xL, 0 ] (0<=xL< xBlockWidth) is defined as follows:

URec[ xL, yL ] = cSL[ xT + xL, yT – 1]

J.8.3.6.3 Deriving of left and up reference samples of the reference block

For each luma sample location (0, yL) such as: 0<=yL< yBlockHeight inside LRef block, sample value LRef [0, yL] is derived by the following ordered steps:

1. The variables xIntL, yIntL, xFracL, and yFracL are derived by:

xIntL = xT + (mvL0[ 0 ] >> 2) -1 (J-8-XX)

yIntL = yT + (mvL0[ 1 ] >> 2) + yL (J-8-XX)

xFracL= mvL0[ 0 ] & 3 (J-8-XX)

yFracL= mvL0[ 1 ] & 3 (J-8-XX)

2. LRef[ 0, yL ] sample is derived as an output of the process specified in the 8.4.2.2.1 sub-clause with ( xIntL, yIntL), ( xFracL, yFracL ) and refPicL0L given as input.

For each luma sample location ( xL, 0 ) such as: 0<=xL< xBlockWidthnside URef block sample value URef[ xL, 0 ] is derived by the following ordered steps:

1. The variables xIntL, yIntL, xFracL, and yFracL are derived by:

xIntL = xT + (mvL0[ 0 ] >> 2) + xL (J-8-XX)

yIntL = yT + (mvL0[ 1 ] >> 2) -1 (J-8-XX)

xFracL = mvL0[ 0 ] & 3 (J-8-XX)

yFracL= mvL0[ 1 ] & 3 (J-8-XX)

1. URef[ xL, yL ] sample is derived as an output of the process specified in the 8.4.2.2.1 sub-clause with ( xIntL, yIntL ), ( xFracL, yFracL ) and refPicL0L given as input.

J.8.3.6.4 Deriviation of NeighborRefSum and NeighborSum

Both NeighborRefSum and NeighborSum are set to 1 and further calculated as follows:

for( j=0; j< yBlockHeight;j++ )  
 if ( Abs( LRec[ 0, j ] - LRef[ 0,j ] ) < 31 ) {

NeighborRefSum = NeighborRefSum + LRef[ 0, j ]

NeighborSum = NeighborSum + LRec[ 0, j ]

}

for( i=0; i< xBlockWidth; i++ )  
 if ( Abs( URec[ i, 0 ] - URef[ i ,0 ] ) < 31 ) {

NeighborRefSum = NeighborRefSum + URef[ i, 0 ] NeighborSum = NeighborSum + URec[ i, 0 ]

}

J.8.3.6.5 Deriviation of prediction weights

LogWDL is set to 15. O0L is set to 0 and W0L is derived as follows:

If (NeighborSum>>4) is equal to (NeighborRefSum >>4), W0L is set to 1.

Otherwise, W0L is equal to ((1<< LogWDL) \* NeighborSum + (NeighborRefSum>>1)) / NeighborRefSum.