|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  10th Meeting: Strasbourg, FR, 18–24 Oct. 2014 | Document: JCT3V-J0107 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **On 3D-HEVC HLS and its alignment with MV-HEVC HLS** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Gerhard Tech Karsten Müller | Email: | gerhard.tech@hhi.fraunhofer.de  karsten.mueller@hhi.fraunhofer.de |
| *Source:* | Fraunhofer HHI | | |

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

# Abstract

In this document several items to align the 3D-HEVC HLS with MV-HEVC HLS and to fix minor 3D-HEVC HLS syntax issues are proposed. In particular it is proposed to use the direct\_dependency\_flag syntax element to signal 3D-HEVC specific inter-layer dependencies, to signal the 3D-HEVC tool enabling flags for texture and depth in the SPS, to fix the inclusion of the vps\_extension2 syntax structure, to remove the MvHevcCompatibilityFlag variable, and to fix the mapping of camera parameters to views.

# Fix of layer dependency indication

## Dependency signalling and indication in MV-HEVC

In MV-HEVC inter-layer dependencies are signalled using the direct\_dependency\_flag[ i ][ j ] syntax element. When direct\_dependency\_flag[ LayerIdxInVps[ iNuhLId ] ][ LayerIdxInVps[ jNuhLId ] ] is equal to 1, the layer with nuh\_layer\_id equal to iNuhLId directly depends on layer with nuh\_layer\_id equal to jNuhLId.

Furthermore, three lists are derived from the from direct\_dependency\_flag syntax element which are utilized to indicate which layers are related to the layer with nuh\_layer\_id equal to iNuhLId.

The lists are:

* IdRefLayer[ iNuhLId ], which includes the nuh\_layer\_id values of all direct and indirect reference layers of the layer with nuh\_layer\_id equal to iNuhLId.
* IdDirectRefLayer[ iNuhLId ], which includes the nuh\_layer\_id values of all direct reference layers of the layer with nuh\_layer\_id equal to iNuhLId.
* IdPredictedLayer[ iNuhLId ], which includes the nuh\_layer\_id values of all layers that indirectly or directly predicted from the layer with nuh\_layer\_id equal to iNuhLId.

All three list are frequently used throughout the MV-HEVC specification e.g. for:

1. Layer initialization, to start up the decoding.
2. Imposing constraints on parameter set activation.
3. Parsing of syntax elements.
4. Interpretation of VUI syntax elements.
5. Interpretation of SEI messages.
6. Derivation of layer sets.
7. Imposing of bitstream constraints.
8. POC reset.
9. Derivation of layers which are necessary for decoding an OLS.

Moreover, the list IdDirectRefLayers[ iNuhLId ] is used for the

1. Derivation of reference picture lists

## Problem statement

Additional to the conventional inter-layer prediction as used in MV-HEVC, new kinds of inter-layer prediction (in the following referred to as 3D-HEVC specific inter-layer prediction modes) have been introduced to 3D-HEVC, as e.g. inter-view motion and residual prediction; motion parameter, quadtree and contour prediction from texture to depth; and disparity prediction from depth maps for texture coding.

Enabling flags of 3D-HEVC specific inter-layer prediction modes are signalled in the 3D-HEVC extension of the VPS. However, additional 3D-HEVC specific dependencies are not explicitly signalled by the direct\_dependency\_flag[ i ][ j ] syntax elements, such that the lists IdRefLayer[ iNuhLId ], IdDirectRefLayer[ iNuhLId ], and IdPredictedLayer[ iNuhLId ] do not reflect the "real" dependencies between layers.

This missing alignment with MV-HEVC causes issues related to items 1 to 9 in section 1.1. E.g. for item 1 the decoding of a depth layer could be started before the decoding of the texture layer from which motion parameters are predicted has been started. Or e.g. for item 9 some layers might not be decoded although they are required as reference.

However, since the reference picture list construction is based on the list IdDirectRefLayer[ iNuhLId ] (item 10), pictures which are required only as reference for 3D-HEVC specific inter-layer prediction modes are currently not included to the reference picture lists, which seems to be desirable.

## Analysis of additional 3D-HEVC dependencies

In the following it is analysed which dependencies occur and need to be additionally signalled for 3D-HEVC specific inter-layer prediction modes to fix above mentioned issues with items 1 to 9.

1. NBDV, inter-view motion, and inter-view residual prediction only refer to layers of pictures present in the reference picture lists. Hence, to layers which are already included in the IdDirectRefLayer[ iNuhLId ] list. No additional dependency signalling is required.
2. QTL, MPI, DMM4 are used for depth coding and refer to the texture layer of the current view. The additional dependency to this texture layer needs to be signalled.
3. VSP, DBBP, and depth refinement are used in texture coding and refer to already coded depth layers. The additional dependencies to these depth layers need to be signalled.

It can be seen item 1 to 3, that for a depth layer only the dependency from the texture layer of the same view needs to be signalled additionally, whereas for a texture layer only the dependencies from depth layers of already coded views need to be signalled additionally.

## Proposed fix

It is proposed to guarantee that dependencies are correctly reflected by the direct\_dependency\_flag syntax element and the derived lists IdDirectRefLayer, IdRefLayer, and IdPredictedLayer, such that no issues related to items 1 to 9 in section 1.1 can occur. For this purpose, it is proposed to disable 3D-HEVC specific inter-layer prediction modes, when not all required layer dependencies are signalled by the direct\_dependency\_flag syntax element or when not all required layers are present.

Furthermore, it is proposed to keep the design, that pictures of layers only used for 3D-HEVC specific inter-layer prediction modes are not included in the reference picture lists (item 10). For this purpose, another list denoted as IdDirectRefListLayer[ iNuhLId ] is constructed, which does not include layers which are only used as reference for 3D-HEVC specific inter-layer prediction modes. This list replaces IdRefListLayer[ iNuhLId ] in the specification of the reference picture list construction.

Due to the finding in section 1.3, that depth layers only refer additionally to texture layers and vice versa, IdDirectRefListLayer[ iNuhLId ] can be constructed by discarding depth layers from IdDirectRefLayer[ iNuhLId ] when the layer with iNuhLId is a texture layer or by discarding texture layers from IdDirectRefLayer[ iNuhLId ] when the layer with iNuhLId is a depth layer.

## Syntax and semantics changes

**I.7.4.3.1.1 Video parameter set extension semantics**

...

**direct\_dependency\_flag**[ i ][ j ] equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. direct\_dependency\_flag[ i ][ j ] equal to 1 specifies that the layer with index j is a direct reference layer for the layer with index i. When direct\_dependency\_flag[ i ][ j ] is not present for i and j in the range of 0 to MaxLayersMinus1, it is inferred to be equal to 0.

...

The variables NumDirectRefLayers[ iNuhLId ], IdDirectRefLayer[ iNuhLId ][ d ], NumDirectRefListLayers[ iNuhLId ], IdDirectRefListLayer[ iNuhLId ][ d ], NumRefLayers[ iNuhLId ], IdRefLayer[ iNuhLId ][ r ], NumPredictedLayers[ iNuhLId ], and IdPredictedLayer[ iNuhLId ][ p ] are derived as follows:

for( i = 0; i <= MaxLayersMinus1; i++ ) {  
 iNuhLId = layer\_id\_in\_nuh[ i ]  
 for( j = 0, d = 0, l = 0, r = 0, p = 0; j <= MaxLayersMinus1; j++ ) {  
 jNuhLid = layer\_id\_in\_nuh[ j ]  
 if( direct\_dependency\_flag[ i ][ j ] )  
 IdDirectRefLayer[ iNuhLId ][ d++ ] = jNuhLid  
 if( direct\_dependency\_flag[ i ][ j ] &&   
 VpsDepthFlag[ iNuhLId ] = = VpsDepthFlag[ jNuhLid ] )  
 IdDirectRefListLayer[ iNuhLId ][ l++ ] = jNuhLid  
 if( DependencyFlag[ i ][ j ] )  
 IdRefLayer[ iNuhLId ][ r++ ] = jNuhLid (F‑4)  
 if( DependencyFlag[ j ][ i ] )  
 IdPredictedLayer[ iNuhLId ][ p++ ] = jNuhLid  
 }  
 NumDirectRefLayers[ iNuhLId ] = d  
 NumDirectRefListLayers[ iNuhLId ] = l  
 NumRefLayers[ iNuhLId ] = r  
 NumPredictedLayers[ iNuhLId ] = p  
}

...

**I.7.4.3.1.2 Video parameter set extension 2 semantics**

For lIdx in the range of 0 to MaxLayersMinus1, inclusive the following applies:

– The variable lId is set equal to layer\_id\_in\_nuh[ lIdx ].

– The variables depthOfRefViewsAvailFlag[ lId ] and textOfCurViewAvailFlag[ lId ] are set equal to 0.

– If VpsDepthFlag[ lId ] is equal to 0, the following applies:

[Remark: The range below is 0 to NumDirectRefListLayer[ lId ] − 1, since the derived disparity vector always refers to a view included in IdDirectRefListLayer[ lId ], and therefore only depth layers of those views might be referenced by the texture.]

– When the following is true, for all values of i in the range of 0 to NumDirectRefListLayer[ lId ] − 1, inclusive, depthOfRefViewsAvailFlag[ lId ] is set equal to 1:

– There is a value refLId[ i ] with direct\_dependency\_flag[ lIdx ][ LayerIdxInVps[ refLId[ i ] ] ] equal to 1, ViewOrderIdx[ refLId[ i ] ] equal to ViewOrderIdx[ IdDirectRefListLayers[ i ] ] and VpsDepthFlag[ refLId[ i ] ] equal to 1.

– Otherwise (VpsDepthFlag[ lId ] is equal to 1), the following applies:

– When the following is true, textOfCurViewAvailFlag[ lId ] is set equal to 1.

– There is a value refLId with direct\_dependency\_flag[ lIdx ][ LayerIdxInVps[ refLId ] ] equal to 1, ViewOrderIdx[ refLId ] equal to ViewOrderIdx[ lId ] and VpsDepthFlag[ refLId ] equal to 0.

**iv\_mv\_pred\_flag**[ layerId ] indicates whether inter-view motion parameter prediction is used in the decoding process of the layer with nuh\_layer\_id equal to layerId. iv\_mv\_pred\_flag[ layerId ] equal to 0 specifies that inter-view motion parameter prediction is not used for the layer with nuh\_layer\_id equal to layerId. iv\_mv\_pred\_flag[ layerId ]equal to 1 specifies that inter-view motion parameter prediction may be used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of iv\_mv\_pred\_flag[ layerId ] is inferred to be equal to 0. When NumDirectRefListLayers[ layerId ] is equal to 0, the value of iv\_mv\_pred\_flag[ layerId ] shall be equal to 0.

**iv\_mv\_scaling\_flag[** layerId **]** equal to 1 specifies that motion vectors used for inter-view prediction in a layer with nuh\_layer\_id equal to layerId may be scaled based on ViewId**[** layerId **]** values. iv\_mv\_scaling\_flag**[** layerId **]** equal to 0 specifies that motion vectors used for inter-view prediction in a layer with nuh\_layer\_id equal to layerId are not scaled based on ViewId[ layerId ] values. When not present, the value of iv\_mv\_scaling\_flag**[** layerId **]** is inferred to be equal to 0.

**log2\_sub\_pb\_size\_minus3**[ layerId ] specifies the value of the variable SubPbSize[ layerId ] that is used in the decoding of prediction units using the inter-view merge candidate. The value of log2\_sub\_pb\_size\_minus3 shall be in the range of ( MinCbLog2SizeY − 3 ) to ( CtbLog2SizeY − 3 ), inclusive.

The variable SubPbSize[ layerId ] is derived as specified in the following:

SubPbSize[ layerId ] = VpsDepthFlag( layerId ) ? CtbSizeY : 1 << ( log2\_sub\_pb\_size\_minus3[ layerId ] + 3 ) (I‑6)

**iv\_res\_pred\_flag**[ layerId ]indicates whether inter-view residual prediction is used in the decoding process of the layer with nuh\_layer\_id equal to layerId. iv\_res\_pred\_flag[ layerId ] equal to 0 specifies that inter-view residual prediction is not used for the layer with nuh\_layer\_id equal to layerId. iv\_res\_pred\_flag[ layerId ]equal to 1 specifies that inter-view residual prediction may be used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of iv\_res\_pred\_flag[ layerId ] is to be equal to 0. When NumDirectRefListLayers[ layerId ] is equal to 0, the value of iv\_res\_pred\_flag[ layerId ] shall be equal to 0.

**view\_synthesis\_pred\_flag**[ layerId ] equal to 0 specifies that view synthesis prediction merge candidates are not used for the layer with nuh\_layer\_id equal to layerId. view\_synthesis\_pred\_flag[ layerId ] equal to 1 specifies that view synthesis prediction merge candidates might be used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of view\_synthesis\_pred\_flag[ layerId ] is inferred to be equal to 0. When NumDirectRefListLayers[ layerId ] is equal to 0, the value of view\_synthesis\_pred\_flag[ layerId ] shall be equal to 0. The variable ViewSynthesisPredFlag[ layerId ] is set equal to ( depthOfRefViewsAvailFlag[ layerId ]  &&  view\_synthesis\_pred\_flag[ layerId ] ).

**depth\_based\_blk\_part\_flag**[ layerId ] equal to 0 specifies that depth based block partitioning is not used for the layer with nuh\_layer\_id equal to layerId. depth\_based\_blk\_part\_flag[ layerId ] equal to 1 specifies that depth based block partitioning might be used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of depth\_based\_blk\_part\_flag[ layerId ] is inferred to be equal to 0. The variable DepthBasedBlkPartFlag[ layerId ] is set equal to ( depthOfRefViewsAvailFlag[ layerId ]  &&  depth\_based\_blk\_part\_flag[ layerId ] ).

**depth\_refinement\_flag**[ layerId ] equal to 0 specifies that depth view components are not used in the derivation process for a disparity vector for the layer with nuh\_layer\_id equal to layerId. depth\_refinement\_flag[ layerId ] equal to 1 specifies that depth components are used in the derivation process for a disparity vector for the layer with nuh\_layer\_id equal to layerId. When not present, the value of depth\_refinement\_flag[ layerId ] is inferred to be equal to 0. The variable DepthRefinementFlag[ layerId ] is set equal to ( depthOfRefViewsAvailFlag[ layerId ]  &&  depth\_refinement\_flag[ layerId ] ).

**mpi\_flag**[ layerId ] equal to 0 specifies that motion parameter inheritance is not used for the layer with nuh\_layer\_id equal to layerId. mpi\_flag[ layerId ] equal to 1 specifies that motion parameter inheritance may be used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of mpi\_flag[ layerId ] is inferred to be equal to 0. The variable MpiFlag[ layerId ] is set equal to ( textOfCurViewAvailFlag[ layerId ]  &&  mpi\_flag[ layerId ] ).

**log2\_mpi\_sub\_pb\_size\_minus3**[ layerId ] specifies the value of the variable MpiSubPbSize[ layerId ] that is used in the decoding of prediction units using the texture merge candidate. The value of log2\_mpi\_sub\_pb\_size\_minus3[ layerId ] shall be in the range of ( MinCbLog2SizeY − 3 ) to ( CtbLog2SizeY − 3 ), inclusive.

The variable MpiSubPbSize[ layerId ] is derived as specified in the following:

MpiSubPbSize[ layerId ] = 1 << ( log2\_mpi\_sub\_pb\_size\_minus3[ layerId ] + 3 ) (I‑7)

**dmm\_cpredtex\_flag**[ layerId ]equal to 1 specifies that the depth intra prediction mode INTRA\_DEP\_DMM\_CPREDTEX might be used in coding blocks in layers with layer\_id equal to layerId and coded in intra prediction mode. dmm\_cpredtex\_flag[ layerId ]equal to 0 specifies that the depth intra prediction mode INTRA\_DEP\_DMM\_CPREDTEX is not used in coding blocks in layers with layer\_id equal to layerId and coded in intra prediction mode. When not present, dmm\_cpredtex\_flag[ layerId ] is inferred to be equal to 0. The variable DmmCPredTexFlag[ layerId ] is set equal to ( textOfCurViewAvailFlag[ layerId ]  &&  dmm\_cpredtex\_flag[ layerId ] ).

**intra\_sdc\_dmm\_wfull\_flag**[ layerId ]equal to 1 specifies that SDC and the depth intra prediction mode INTRA\_DEP\_DMM\_WFULL might be used in coding blocks in layers with layer\_id equal to layerId and coded in intra prediction mode. intra\_sdc\_dmm\_wfull\_flag[ layerId ]equal to 0 specifies that SDC and the depth intra prediction mode INTRA\_DEP\_DMM\_WFULL is not used in coding blocks in layers with layer\_id equal to layerId and coded in intra prediction mode. When not present, intra\_sdc\_dmm\_wfull\_flag[ layerId ] is inferred to be equal to 0.

**lim\_qt\_pred\_flag**[ layerId ]equal to 1 specifies that prediction of a limited quadtree is used for the layer with nuh\_layer\_id equal to layerId.. lim\_qt\_pred\_flag[ layerId ] equal to 0 specifies that prediction of a limited quadtree is not used for the layer with nuh\_layer\_id equal to layerId. When not present, the value of lim\_qt\_pred\_flag[ layerId ] is inferred to be equal to 0. The variable LimQtPredFlag[ layerId ] is set equal to ( textOfCurViewAvailFlag[ layerId ]  &&  lim\_qt\_pred\_flag[ layerId ] ).

[Remark: Moreover, view\_synthesis\_pred\_flag, depth\_based\_blk\_part\_flag, depth\_refinement\_flag, mpi\_flag, dmm\_cpredtex\_flag, and lim\_qt\_pred\_flag are replaced by ViewSynthesisPredFlag, DepthBasedBlkPartFlag, DepthRefinementFlag, MpiFlag, DmmCpredTexFlag, and LimQtPredFlag, respectively, throughout the spec text.]

...

**I.7.3.6.1 General slice segment header syntax**

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | **Descriptor** |
| ... |  |
| if( sps\_temporal\_mvp\_enabled\_flag ) |  |
| **slice\_temporal\_mvp\_enabled\_flag** | u(1) |
| } |  |
| if( nuh\_layer\_id > 0 && !default\_ref\_layers\_active\_flag &&  NumDirectRefListLayers[ nuh\_layer\_id ] > 0 ) { |  |
| **inter\_layer\_pred\_enabled\_flag** | u(1) |
| if( inter\_layer\_pred\_enabled\_flag &&   NumDirectRefListLayers[ nuh\_layer\_id ] > 1) { |  |
| if( !max\_one\_active\_ref\_layer\_flag ) |  |
| **num\_inter\_layer\_ref\_pics\_minus1** | u(v) |
| if( NumActiveRefLayerPics != NumDirectRefListLayers[ nuh\_layer\_id ] ) |  |
| for( i = 0; i < NumActiveRefLayerPics; i++ ) |  |
| **inter\_layer\_pred\_layer\_idc[**i ] | u(v) |
| } |  |
| } |  |
| ... |  |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| if( ( weighted\_pred\_flag && slice\_type = = P ) | |  ( weighted\_bipred\_flag && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| else if( nuh\_layer\_id > 0 && !DepthFlag && !MvHevcCompatibilityFlag &&   NumDirectRefListLayers[ nuh\_layer\_id ] > 0 ) { |  |
| **slice\_ic\_enable\_flag** | u(1) |
| if( slice\_ic\_enable\_flag **)** |  |
| **slice\_ic\_disable\_merge\_zero\_idx\_flag** | u(1) |
| } |  |
| if( nuh\_layerId > 0 && DepthFlag && !MvHevcCompatibilityFlag ) |  |
| **slice\_single\_sample\_mode\_enable\_flag** | u(1) |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| } |  |
| ... |  |

**I.7.4.7.1 General slice segment header semantics**

...

**num\_inter\_layer\_ref\_pics\_minus1** plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num\_inter\_layer\_ref\_pics\_minus1 syntax element is Ceil( Log2( NumDirectRefListLayers[ nuh\_layer\_id ] ) ) bits. The value of num\_inter\_layer\_ref\_pics\_minus1 shall be in the range of 0 to NumDirectRefListLayers[ nuh\_layer\_id ] − 1, inclusive.

The variables numRefLayerPics and refLayerPicIdc[ j ] are derived as follows:

for( i = 0, j = 0; i < NumDirectRefListLayers[ nuh\_layer\_id ]; i++ ) {  
 refLayerIdx = LayerIdxInVps[ IdDirectRefListLayer[ nuh\_layer\_id ][ i ] ]  
 if( sub\_layers\_vps\_max\_minus1[ refLayerIdx ] >= TemporalId && ( TemporalId = = 0 | | (I‑51)  
 max\_tid\_il\_ref\_pics\_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh\_layer\_id ] ] > TemporalId ) )  
 refLayerPicIdc[ j++ ] = i  
}  
numRefLayerPics = j

The variable NumActiveRefLayerPics is derived as follows:

if( nuh\_layer\_id = = 0 | | numRefLayerPics = = 0 )  
 NumActiveRefLayerPics = 0  
else if( default\_ref\_layers\_active\_flag )  
 NumActiveRefLayerPics = numRefLayerPics  
else if( !inter\_layer\_pred\_enabled\_flag ) (I‑52)  
 NumActiveRefLayerPics = 0  
else if( max\_one\_active\_ref\_layer\_flag | | NumDirectRefListLayers[ nuh\_layer\_id ] = = 1 )  
 NumActiveRefLayerPics = 1  
else  
 NumActiveRefLayerPics = num\_inter\_layer\_ref\_pics\_minus1 + 1

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

**inter\_layer\_pred\_layer\_idc**[ i ] specifies the variable, RefPicLayerId[ i ], representing the nuh\_layer\_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the inter\_layer\_pred\_layer\_idc[ i ] syntax element is Ceil( Log2( NumDirectRefListLayers[ nuh\_layer\_id ] ) ) bits. The value of inter\_layer\_pred\_layer\_idc[ i ] shall be in the range of 0 to NumDirectRefListLayers[ nuh\_layer\_id ] − 1, inclusive. When i is greater than 0, inter\_layer\_pred\_layer\_idc[ i ] shall be greater than inter\_layer\_pred\_layer\_idc[ i − 1 ]. When not present, the value of inter\_layer\_pred\_layer\_idc[ i ] is inferred to be equal to refLayerPicIdc[ i ].

The variables RefPicLayerId[ i ] for all values of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, are derived as follows:

for( i = 0, j = 0; i < NumActiveRefLayerPics; i++ ) (I‑53)  
 RefPicLayerId[ i ] = IdDirectRefListLayer[ nuh\_layer\_id ][ inter\_layer\_pred\_layer\_idc[ i ] ]

...

# 3D-HEVC coding tool enabling flags

## Problem statement and proposal

Currently, 3D-HEVC coding tool enabling flags are signalled per layer in the VPS. However, conceptually the function of the VPS is to signal information on inter-layer dependencies and relationships. Therefore, it is proposed to signal 3D-HEVC coding tool enabling flags in the SPS extension, which would also be aligned with the signalling of RExt related coding tool enabling flags. Camera parameters stay in the VPS.

Moreover, coding tools enabling flags are signalled per layer, which gives a waste flexibility which is not used in most of the use cases of 3D-HEC and which is furthermore not provided for coding tool enabling flags of the base specification. Therefore, it is proposed to signal only two sets of 3D-HEVC coding tool enabling flags per SPS: One for layers with DepthFlag equal to 0, and the other for layers with DepthFlag equal to 1. Flexibility for different configurations is still given, since different SPSs can be active for different layers.

## Syntax and semantics changes

**Video parameter set extension 2**

|  |  |
| --- | --- |
| vps\_extension2( ) { | **Descriptor** |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_one\_bit** | u(1) |
| ~~for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) {~~ |  |
| ~~layerId = layer\_id\_in\_nuh[ i ]~~ |  |
| ~~if ( layerId != 0 ) {~~ |  |
| **~~iv\_mv\_pred\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~log2\_sub\_pb\_size\_minus3~~**~~[ layerId ]~~ | ~~ue(v)~~ |
| ~~if ( !VpsDepthFlag[ layerId ] ) {~~ |  |
| **~~iv\_res\_pred\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~depth\_refinement\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~view\_synthesis\_pred\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~depth\_based\_blk\_part\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| ~~} else {~~ |  |
| **~~mpi\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~vps\_depth\_modes\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~lim\_qt\_pred\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| **~~vps\_inter\_sdc\_flag~~**~~[ layerId ]~~ | ~~u(1)~~ |
| ~~}~~ |  |
| ~~}~~ |  |
| ~~}~~ |  |
| **cp\_precision** | ue(v) |
| for( i = 0; i < NumViews; i++ ) { |  |
| **cp\_present\_flag**[ i ] | u(1) |
| if( cp\_present\_flag[ i ] ) { |  |
| **cp\_in\_slice\_segment\_header\_flag**[ i ] | u(1) |
| if( !cp\_in\_slice\_segment\_header\_flag[ i ] ) |  |
| for( j = 0; j < i; j++ ) { |  |
| **vps\_cp\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_off**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_scale\_plus\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_off\_plus\_off**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| } |  |
| **~~iv\_mv\_scaling\_flag~~** | ~~u(1)~~ |
| **~~log2\_mpi\_sub\_pb\_size\_minus3~~** | ~~ue(v)~~ |
| } |  |

**Sequence parameter set 3d extension**

|  |  |
| --- | --- |
| sps\_3d\_extension( ) { | **Descriptor** |
| for( depthFlag = 0; depthFlag <= 1; depthFlag++ ) { |  |
| **iv\_mv\_pred\_flag**[ depthFlag ] | u(1) |
| **log2\_sub\_pb\_size\_minus3**[ depthFlag ] | ue(v) |
| if ( !depthFlag ) { |  |
| **iv\_res\_pred\_flag**[ depthFlag ] | u(1) |
| **depth\_refinement\_flag**[ depthFlag ] | u(1) |
| **view\_synthesis\_pred\_flag**[ depthFlag ] | u(1) |
| **depth\_based\_blk\_part\_flag**[ depthFlag ] | u(1) |
| } else { |  |
| **mpi\_flag**[ depthFlag ] | u(1) |
| **vps\_depth\_modes\_flag**[ depthFlag ] | u(1) |
| **lim\_qt\_pred\_flag**[ depthFlag ] | u(1) |
| **vps\_inter\_sdc\_flag**[ depthFlag ] | u(1) |
| } |  |
| } |  |
| **iv\_mv\_scaling\_flag** | u(1) |
| **log2\_mpi\_sub\_pb\_size\_minus3** | ue(v) |
| } |  |

Not present values are inferred to be equal to 0. Moreover, e.g. iv\_mv\_pred\_flag[ layerId ] is replace by the variable

IvMvPredFlag[ layerId ] = ( layerId > 0 ) ? iv\_mv\_pred\_flag[ VpsDepthFlag[ layerId ] ] : 0

throughout the spec text (and accordingly other syntax elements).

The sps\_3d\_extension syntax structure is included according to the SPS extension mechanism used in Annex F utilizing one bit of the sps\_extension\_6bits syntax element.

# Indication of VPS extension 2

## Problem statement

In the current draft it is not specified how to include the vps\_extension2 syntax structure in the VPS.

## Proposed fix

It is proposed to

* rename the vps\_extension2 to vps\_3d\_extension.
* signal an additional vps\_3d\_extesion\_flag when vps\_extension2\_flag is equal to 1.
* signal the vps\_3d\_extension( ) syntax structure when vps\_extension2 and vps\_3d\_extension\_flag are both equal to 1.
* signal an additional vps\_extension3\_flag when vps\_extension2\_flag is equal to 1.
* signal future VPS extensions when vps\_extension2 and vps\_extension3\_flag are both equal to 1.

This allows future HEVC extensions to set vps\_extension2\_flag, vps\_3d\_extension\_flag, and vps\_extension3\_flag to 1, 0, and 1, respectively, when a future VPS extension but not the vps\_3d\_extension( ) is required.

## Proposed syntax and semantics

F.7.3.2.1 Video parameter set RBSP

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| ... |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_extension( ) |  |
| **vps\_extension2\_flag** | u(1) |
| if( vps\_extension2\_flag ) { |  |
| **vps\_3d\_extension\_flag** | u(1) |
| if( vps\_3d\_extension\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_3d\_extension\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_3d\_extension( ) |  |
| } |  |
| **vps\_extension3\_flag** | u(1) |
| if ( vps\_extension3\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| } |  |
| } |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

F.7.4.3.1 Video parameter set RBSP semantics

...

**~~vps\_extension2\_flag~~** ~~equal to 0 specifies that no vps\_extension2( ) syntax structure is present in the VPS RBSP syntax structure. vps\_extension\_flag equal to 1 specifies that the vps\_extension2( ) syntax structure is present in the VPS RBSP syntax structure.~~

~~The variable MvHevcCompatibilityFlag is set equal to !vps\_extension2\_flag.~~

**vps\_extension2\_flag** equal to 0 specifies that no vps\_3d\_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps\_extension2\_flag equal to 1 specifies that the vps\_3d\_extension( ) syntax structure might be present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps\_extension2\_flag shall be equal to 1.

**vps\_3d\_extension\_flag** equal to 0 specifies that no vps\_3d\_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps\_3d\_extension\_flag equal to 1 specifies that the vps\_3d\_extension( ) syntax structure is present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps\_3d\_extension\_flag shall be equal to 1.

The variable MvHevcCompatibilityFlag is set equal to !vps\_3d\_extension\_flag.

**vps\_3d\_extension\_alignment\_bit\_equal\_to\_one** shall be equal to 1.

**vps\_extension3\_flag** equal to 0 specifies that no vps\_extension\_data\_flag syntax elements are present in the VPS RBSP syntax structure. vps\_extension3\_flag shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for vps\_extension3\_flag is reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification shall ignore all data that follow the value 1 for vps\_extension3\_flag in a VPS RBSP.

...

I.7.3.2.1.2 Video parameter set 3d extension ~~2~~ syntax

|  |  |
| --- | --- |
| vps\_3d\_extension~~2~~( ) { | **Descriptor** |
| ~~while( !byte\_aligned( ) )~~ |  |
| **~~vps\_extension\_byte\_alignment\_reserved\_one\_bit~~** | u(1) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| layerId = layer\_id\_in\_nuh[ i ] |  |
| ... |  |

[Remark: Changes in vps\_extension2 are editorial only.]

# Removal of MvHevcCompatibilityFlag

## Problem statement

In the current draft the conformance of a bitstream to MV-HEVC is specified by the MvHevcCompatibilityFlag which is based on the presence of vps\_extension2( ) syntax structure. However, in MV-HEVC the correct decoding process is chosen using the general\_profile\_idc syntax element.

## Proposal

It is proposed to remove the MvHevcCompatibilityFlag from the draft and align the selection of the decoding process and the syntax specification with MV-HEVC.

## Specification changes

**F.8.1.3 Common decoding process for a coded picture**

...

The following applies for the decoding of the current picture:

– If the current picture has nuh\_layer\_id equal to 0, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause F.7 and the decoding process for a coded picture with nuh\_layer\_id equal to 0 as specified in clause F.8.1.4 is invoked.

– Otherwise, the following applies:

– When vps\_base\_layer\_internal\_flag is equal to 0, vps\_base\_layer\_available\_flag is equal to 1, TargetDecLayerSetIdx is in the range of 0 to vps\_num\_layer\_sets\_minus1, inclusive, TemporalId is less than or equal to sub\_layers\_vps\_max\_minus1[ 0 ], the current picture is the first coded picture of an access unit, and a decoded picture with nuh\_layer\_id equal to 0 is provided by external means for the current access unit, clause F.8.1.9 is invoked after the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, but prior to decoding any slice segment of the first coded picture of the access unit.

– For the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, the decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in clause F.8.1.5 is invoked.

– Let lIdx be equal to such value for which the nuh\_layer\_id value of the current picture is equal to LayerSetLayerIdList[ OlsIdxToLsIdx[ TargetOlsIdx ] ][ lIdx ].

– If general\_profile\_idc in the profile\_tier\_level( ) syntax structure VpsProfileTierLevel[ profile\_tier\_level\_idx[ TargetOlsIdx ][ lIdx ] ] is equal to 6, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause G.7 and the decoding process of clause G.8.1.2 is invoked.

– Otherwise, general\_profile\_idc in the profile\_tier\_level( ) syntax structure VpsProfileTierLevel[ profile\_tier\_level\_idx[ TargetOlsIdx ][ lIdx ] ] is equal to 7, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause H.7 and the decoding process of clause H.8.1.2 is invoked.

– Otherwise, general\_profile\_idc in the profile\_tier\_level( ) syntax structure VpsProfileTierLevel[ profile\_tier\_level\_idx[ TargetOlsIdx ][ lIdx ] ] is equal to 8, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause I.8 and the decoding process of clause I.8.1.2 is invoked.

[Remark: Above change assumes that 3D-HEVC is indicated by a profile\_idc of 8.]

– After all slices of the current picture have been decoded, the decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in clause F.8.1.6 is invoked.

...

I.7.4.3.1 Video parameter set RBSP semantics

...

**vps\_extension2\_flag** equal to 0 specifies that no vps\_extension2( ) syntax structure is present in the VPS RBSP syntax structure. vps\_extension\_flag equal to 1 specifies that the vps\_extension2( ) syntax structure is present in the VPS RBSP syntax structure.

~~The variable MvHevcCompatibilityFlag is set equal to !vps\_extension2\_flag.~~

...

Throughout the spec

Replace MvHevcCompatibilityFlag with 0.

# Camera parameter signalling

## Problem statement

In the current 3D-HEVC specification camera parameters are parsed by iterating over the number of views present in the bitstream (VPS) and/or the number of views coded before the current view (VPS/slice header). In the current design it is assumed that the value parsed in the i-th iteration is related to the view with ViewIdx equal to i. However, when ViewIdx values of present views are not consecutive (which is similar to MV-HEVC not required, although they are required to be ascending), this assumption does not hold and the mapping of the value parsed in the i-the iteration to the correct view is broken.

Consider e.g. the case when NumViews is equal to 3 and the three views have ViewIdx values of 0, 3, and 5: Only vps\_cp\_scale[ 0 ][ j ], vps\_cp\_scale[ 1 ][ j ], and vps\_cp\_scale[ 2 ][ j ] values are present and according to semantics related to views with ViewIdx values of 0, 1, and 2, although they should be related to views with ViewIdx equal to 0, 3 and 5.

## Proposed fix

It is proposed to create a list ViewOIdxList of present ViewIdx values and to use this list to correctly map the indices used for parsing to ViewIdx values. Hence, the values parsed in the i-th iteration is related to the view with ViewIdx equal to ViewOIdxList[ i ] instead of relating it to the view with ViewIdx equal to i.

E.g. the above example ViewOIdxList would be equal to { 0, 3, 5 }.

## Syntax and semantics

I.7.4.3.1.1 Video parameter set extension semantics

...

The variable ScalabilityId[ i ][ smIdx ] specifying the identifier of the smIdx-th scalability dimension type of the i-th layer, the variable ViewOrderIdx[ layer\_id\_in\_nuh[ i ] ] specifying the view order index of the i-th layer, the variable VpsDepthFlag[ layer\_id\_in\_nuh[ i ] ] specifying the depth flag of the i-th layer, and the variable ViewScalExtLayerFlag specifying whether the i-th layer is a view scalability extension layer are derived as follows:

NumViews = 1  
 ViewOIdxList[ 0 ] = 0   
 for( i = 0; i <= MaxLayersMinus1; i++ ) {  
 lId = layer\_id\_in\_nuh[ i ]  
 for( smIdx= 0, j = 0; smIdx < 16; smIdx++ )  
 if( scalability\_mask\_flag[ smIdx ] )  
 ScalabilityId[ i ][ smIdx ] = dimension\_id[ i ][ j++ ]  
 VpsDepthFlag[ lId ] = ScalabilityId[ i ][ 0 ]  
 ViewOrderIdx[ lId ] = ScalabilityId[ i ][ 1 ]  
 if( i > 0 && ( ViewOrderIdx[ lId ] != ScalabilityId[ i − 1][ 1 ] ) )  
 ViewOIdxList[ NumViews++ ] = ViewOrderIdx[ lId ]   
 ViewScalExtLayerFlag[ lId ] = ( ViewOrderIdx[ lId ] > 0 )  
 AuxId[ lId ] = ScalabilityId[ i ][ 3 ]  
 }

...

I.7.3.2.1.2 Video parameter set extension 2 syntax

|  |  |
| --- | --- |
| vps\_extension2( ) { | **Descriptor** |
| ... |  |
| **cp\_precision** | ue(v) |
| ~~for( i = 1; i < NumViews; i++ ) {~~ |  |
| for( n = 1; n < NumViews; n++ ) { |  |
| i = ViewOIdxList[ n ] |  |
| **cp\_present\_flag**[ i ] | u(1) |
| if( cp\_present\_flag[ i ] ) { |  |
| **cp\_in\_slice\_segment\_header\_flag**[ i ] | u(1) |
| if( !cp\_in\_slice\_segment\_header\_flag[ i ] ) |  |
| ~~for( j = 0; j < i; j++ ) {~~ |  |
| for( m = 0; m < n; m++ ) { |  |
| j = ViewOIdxList[ m ] |  |
| **vps\_cp\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_off**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_scale\_plus\_scale**[ i ][ j ] | se(v) |
| **vps\_cp\_inv\_off\_plus\_off**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| } |  |
| ... |  |

I.7.3.6.1 General slice segment header syntax

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | Descriptor |
| ... |  |
| if( nuh\_layer\_id > 0 && cp\_in\_slice\_segment\_header\_flag[ ViewIdx ] ) |  |
| ~~for ( j = 0; j < ViewIdx; j++ ) {~~ |  |
| for ( m = 0; ViewOIdxList[ m ] < ViewIdx ; m++ ) { |  |
| j = ViewOIdxList[ m ] |  |
| **cp\_scale**[ j ] | se(v) |
| **cp\_off**[ j ] | se(v) |
| **cp\_inv\_scale\_plus\_scale**[ j ] | se(v) |
| **cp\_inv\_off\_plus\_off**[ j ] | se(v) |
| } |  |
| ... |  |

# Patent rights declaration(s)

**Fraunhofer HHI may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**