



# JCTVC-M0046: High-level syntax modifications for SHVC

*Jill Boyce, Danny Hong, Wonkap Jang*

# Proposals for SHVC High-level syntax

1. Add syntax element in slice header to signal inter-layer texture and motion prediction from different direct dependent reference layers, for TextureRL (IntraBL) mode only
2. Add syntax elements in the slice segment header for enhancement layer slice skipping
3. Add syntax elements in the SPS extension to enable individual inter-layer prediction coding tools
4. Add syntax elements in the slice segment header for adaptive and default inter-layer texture prediction and enabling motion prediction

# 1. Add syntax elements in slice header to signal inter-layer prediction from different direct dependent reference layers



- Background
  - VPS extension includes syntax elements to signal one or more direct dependent reference layers to be used for inter-layer prediction in enhancement layer
  - Current SHM software and test model only support the use of one direct dependent reference layer
    - For ref index mode, the ref\_idx syntax element can be used to signal different reference layers
    - For TextureRL (IntraBL) mode, no syntax elements in test model to signal different reference layers for the slice
  - SVC uses ref\_layer\_dq\_id syntax element in slice header in scalable extension to indicate which dq\_id layer used for inter-layer prediction

# 1. Add syntax elements in slice header to signal inter-layer prediction from different direct dependent reference layers

- Proposal
  - Add conditional syntax element in the slice segment header in enhancement layers to indicate which directly dependent reference layer used for inter-layer prediction, in TextureRL (IntraBL) mode
  - Only allows one reference layer to be selected per slice, but can be any of the directly dependent reference layers
  - Proposed syntax element only present when necessary
    - Spatial/SNR scalability enhancement layer slice
    - More than one directly dependent reference layer
    - Inter-layer texture or motion enabled
  - u(v) coding used
    - Numbers of bits based on number of directly dependent reference layers



# 1. Proposed syntax

**Green** for changes proposed in JCTVC-M0045, **yellow** for current proposal

slice_segment_header( ) {	Descriptor
...	
if (nuh_layer_id > 0) {	
...	
if( DependencyId [ nuh_layer_id ] > 0) {	
if( NumDirectRefLayers[ nuh_layer_id ] > 1 && ( InterLayerTextureRlEnableFlag    InterLayerMotionPredictionEnableFlag ) )	
inter_layer_pred_layer_idc	u(v)
}	
}	
if( slice_segment_header_extension_present_flag ) {	
slice_segment_header_extension_length	ue(v)
for( i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[ i ]	u(8)
}	
byte_alignment( )	
}	

# 1. Proposed semantics

**inter\_layer\_pred\_layer\_idc** specifies that the reference layer picture, rIPic, used for inter layer texture prediction or inter layer motion prediction is the coded picture with nuh\_layer\_id equal to RefLayerId[ nuh\_layer\_id ][ inter\_layer\_pred\_layer\_idc ]. The number of bits used to represent inter\_layer\_pred\_layer\_idc is  $\text{Ceil}(\text{Log2}(\text{NumDirectRefLayers}[\text{nuh\_layer\_id}]))$ .

When not present, inter\_layer\_pred\_layer\_idc is inferred to be equal to 0.

When inter\_layer\_pred\_layer\_idc present, a coded picture with nuh\_layer\_id equal to RefLayerId[ nuh\_layer\_id ][ inter\_layer\_pred\_layer\_idc ] shall be present in the current access unit.

## 2. Add syntax elements in the slice segment header for enhancement layer slice skipping

- SVC contains syntax elements to indicate if an enhancement layer is skipped, and how many MBs are contained within the skipped slice
  - Has been found to be useful in SVC products
    - Encoder that is CPU or bitrate starved
    - Decoder or middle box for error resiliency
- Propose similar slice skip syntax elements for SHVC
  - Methods proposed for both ref index and textureRL (intra BL) modes
  - Syntax and semantics are proposed
  - Decoding method for skipped slice is left undefined
    - May be defined separately
    - Depends on exact tools included in design
    - Out of scope of current proposal

## 2. Proposed syntax element discussion

- Two syntax elements proposed, `scalable_slice_skip_flag` and `num_ctbs_in_skipped_slice`
- Conditionally present
  - In spatial/SNR enhancement layers
  - At least one directly dependent reference layer
  - Inter-layer texture prediction enabled
    - Either ref index or TextureRL (IntraBL) mode
- `u(v)` coding used for `num_ctbs_in_skipped_slice` for coding efficiency
  - SVC used `ue(v)` coding for similar syntax element
  - Skipped slices likely to be used when available bitrate very low

## 2. Proposed slice segment header syntax

slice_segment_header( )	Descriptor
...	
if( nuh_layer_id > 0 ) {	
if( DependencyId[ nuh_layer_id ] > 0 ) {	
if ( NumDirectRefLayers[ nuh_layer_id ] > 0 ) {	
if( sps_inter_layer_texture_rl_enable_flag    sps_inter_layer_ref_idx_enable_flag )	
scalable_slice_skip_flag	u(1)
if( scalable_slice_skip_flag )	
num_ctbs_in_skipped_slice	u(v)
}	
}	
}	
if( slice_segment_header_extension_present_flag ) {	
slice_segment_header_extension_length	ue(v)
for( i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[ i ]	u(8)
}	
byte_alignment( )	
}	



## 2. Proposed slice segment header semantics

**scalable\_slice\_skip\_flag** equal to 1 specifies that `slice_segment_data( )` is not present for the current slice.  
**scalable\_slice\_skip\_flag** equal to 0 specifies that `slice_segment_data( )` is present for the current slice.

**num\_ctbs\_in\_skipped\_slice** specifies the number of coding tree blocks for slice with **scalable\_skip\_slice\_flag** equal to 1. The number of bits used to represent **num\_ctbs\_in\_skipped\_slice** is  $\text{Ceil}(\text{Log2}(\text{PicWidthInCtbsY} * \text{PicHeightInCtbsY}))$

[Ed. (JB): Need decoding process description for CUs in skipped slice.]

## 2. Proposed slice segment layer syntax

slice_segment_layer_rbsp( ) {	<b>Descriptor</b>
slice_segment_header( )	
if (!scalable_slice_skip_flag)	
slice_segment_data( )	
rbsp_slice_segment_trailing_bits( )	
}	

### 3. Add syntax elements in SPS extension to enable individual inter-layer prediction coding tools

- SHM1 Test Model includes `sps_inter_layer_mfm_enable_flag` in extension to indicate enabling of motion field mapping process in ref index framework
- Propose that syntax elements be similarly added in SPS extension to individually signal enabling of the other scalability coding tools included in the Test Model
  - ref index texture prediction
  - texture RL (IntraBL) prediction
  - inter layer motion prediction
- SHM Test Model includes corresponding place holder variables to indicate which individual inter-layer prediction coding tools are enabled, without defining how values of those variables are set
- SPS extension seems most appropriate location

## 3. Proposed SPS extension syntax

Proposal in **yellow**, previous proposal in JCTVC-M0045 in **green**.

Proposed to be conditional present, but may be always present

sps_extension( ) {	Descriptor
for( i = 0; i < 3; i++ )	
extension_type_flag[ i ]	u(1)
if ( extension_type_flag[ 0 ] ) {	
inter_view_mv_vert_constraint_flag	u(1)
// additional syntax elements	
}	
if ( extension_type_flag[ 1 ] ) {	
sps_inter_layer_mfm_enable_flag	u(1)
sps_inter_layer_ref_idx_enable_flag	u(1)
sps_inter_layer_texture_rl_enable_flag	u(1)
sps_inter_layer_motion_prediction_enable_flag	u(1)
}	
if ( extension_type_flag[ 2 ] ) {	
// additional syntax elements	
}	
}	

## 3. Proposed SPS extension semantics

**sps\_inter\_layer\_ref\_idx\_enable\_flag** equal to 1 specifies the inter-layer reference index mapping process is applied as specified in subclause G.8.

sps\_inter\_layer\_ref\_idx\_enable\_flag equal to 0 specifies the inter-layer reference index mapping process is not applied. When not present, the value of sps\_inter\_layer\_ref\_idx\_enable\_flag is inferred to be equal to 0.

**sps\_inter\_layer\_texture\_rl\_enable\_flag** equal to 1 specifies that the inter-layer texture\_rl process is applied as specified in subclause H.8.

sps\_inter\_layer\_texture\_rl\_enable\_flag equal to 0 specifies the inter-layer texture\_rl process is not applied. When not present, the value of sps\_inter\_layer\_texture\_rl\_enable\_flag is inferred to be equal to 0.

When sps\_inter\_layer\_ref\_idx\_enable\_flag equal to 1,  
sps\_inter\_layer\_texture\_rl\_enable\_flag shall be equal to 0.

**sps\_inter\_layer\_motion\_prediction\_enable\_flag** equal to 1 specifies the inter-layer motion prediction process is applied as specified in subclause H.8.5.3.

sps\_inter\_layer\_motion\_prediction\_enable\_flag equal to 0 specifies the inter-layer motion prediction process is not applied. When not present, the value of sps\_inter\_layer\_motion\_prediction\_enable\_flag is inferred to be equal to 0.

When sps\_inter\_layer\_mfm\_enable\_flag equal to 1,  
sps\_inter\_layer\_motion\_prediction\_enable\_flag shall be equal to 0.

## 4. Add syntax elements in slice segment header for adaptive and default inter-layer texture prediction and enabling motion prediction

- New slice segment header syntax elements proposed for SHVC, similar to those found in SVC
  - Adaptive and default inter-layer texture prediction
  - Enabling inter-layer motion prediction
  - Proposed for both ref index or TextureRL (IntraBL) modes
- Similar tools found useful in SVC implementations
  - Encoded slice can always, sometimes, or never use inter-layer texture prediction, and sometimes or never use inter-layer motion prediction
- Proposed syntax elements are conditionally present
  - In spatial/SNR enhancement layer
  - At least one directly dependent reference layer
  - Inter-layer texture prediction enabled, for adaptive\_inter\_layer\_texture\_flag and default\_inter\_layer\_texture\_flag
  - Inter-layer motion prediction enabled for enable\_inter\_layer\_motion\_flag



# 4. Proposed slice segment header syntax

Green for previous proposals, yellow for current proposal

slice_segment_header( )	Descriptor
...	
if( nuh_layer_id > 0 ) {	
if( DependencyId[ nuh_layer_id ] > 0 ) {	
if ( NumDirectRefLayers[ nuh_layer_id ] > 0 ) {	
if( sps_inter_layer_texture_rl_enable_flag    sps_inter_layer_ref_idx_enable_flag )	
scalable_slice_skip_flag	
if( scalable_slice_skip_flag )	
num_ctbs_in_skipped_slice	u(v)
else {	
if( sps_inter_layer_texture_rl_enable_flag	
sps_inter_layer_ref_idx_enable_flag ) {	
adaptive_inter_layer_texture_flag	u(1)
if( !adaptive_inter_layer_texture_flag )	
default_inter_layer_texture_flag	u(1)
}	
if( sps_inter_layer_motion_prediction_enable_flag	
sps_inter_layer_mfm_enable_flag )	
enable_inter_layer_motion_flag	u(1)
}	
}	
}	
if( slice_segment_header_extension_present_flag ) {	
slice_segment_header_extension_length	ue(v)
for( i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[ i ]	u(8)
}	
byte_alignment( )	



## 4. Proposed slice segment header semantics

**adaptive\_inter\_layer\_texture\_flag** equal to 1 specifies if the inter-layer texture prediction mode is signaled for coding units in the current slice. **adaptive\_inter\_layer\_texture\_flag** equal to 0 specifies the inter-layer texture prediction mode is not signaled for coding units in the current slice. When not present, the value of **adaptive\_inter\_layer\_texture\_flag** is inferred to be equal to 0.

**default\_inter\_layer\_texture\_flag** equal to 1 specifies if inter-layer texture prediction is performed for the current slice. **default\_inter\_layer\_texture\_flag** equal to 0 specifies inter-layer texture prediction is not performed for the current slice. When not present, the value of **default\_inter\_layer\_texture\_flag** is inferred to be equal to 0.

The variables **InterLayerTextureRLEnableFlag** and **InterLayerRefIdxEnableFlag** are derived as follows.

**InterLayerTextureRLEnableFlag** = (**adaptive\_inter\_layer\_texture\_flag** || **default\_inter\_layer\_texture\_flag**)  
&& **sps\_inter\_layer\_texture\_rl\_enable\_flag**

**InterLayerRefIdxEnableFlag** = (**adaptive\_inter\_layer\_texture\_flag** || **default\_inter\_layer\_texture\_flag**)  
&& **sps\_inter\_layer\_ref\_idx\_enable\_flag**

When **InterLayerTextureRLEnableFlag** equal to 1, the Decoding process for coding units coded in textureRL prediction mode is applied as specified in subclause H.8.8. When **InterLayerRefIdxEnableFlag** equal to 1, the inter-layer ref index prediction process is applied as specified in subclause G.8.3.

**enable\_inter\_layer\_motion\_flag** equal to 1 specifies that inter-layer motion prediction may be performed for the current slice. **enable\_inter\_layer\_motion\_flag** equal to 0 specifies that inter-layer motion prediction is not performed for the current slice.

The variables **InterLayerMotionPredictionEnableFlag** and **InterLayerMfmEnableFlag** are derived as follows.

**InterLayerMotionPredictionEnableFlag** = **sps\_inter\_layer\_motion\_prediction\_enable\_flag** && **enable\_inter\_layer\_motion\_flag**

**InterLayerMfmEnableFlag** = **sps\_inter\_layer\_mfm\_enable\_flag** && **enable\_inter\_layer\_motion\_flag**

When **InterLayerMotionPredictionEnableFlag** equal to 1, the derivation process for motion vector components and reference indices process is applied as specified in subclause H.8.5.3.1. **InterLayerMfmEnableFlag** equal to 1 specifies the motion field mapping process is applied as part of the resampling process for reference pictures specified in subclause G.8.1.2

# 4. Proposed CU syntax and semantics for TextureRL (IntraBL) mode

coding_unit( x0, y0, log2CbSize ) {	Descriptor
...	
if( cu_skip_flag[ x0 ][ y0 ] )	
prediction_unit( x0, y0, nCbS, nCbS )	
else {	
if ( nuh_layer_id > 0 && InterLayerTextureRIEnableFlag && adaptive_inter_layer_texture_flag )	
texture_rl_flag[ x0 ][ y0 ]	ae(v)
if( !texture_rl_flag[ x0 ][ y0 ] ) {	
if( slice_type != I )	
pred_mode_flag	ae(v)
...	

**texture\_rl\_flag[ x0 ][ y0 ]** specifies the prediction mode for the current coding unit. texture\_rl\_flag[ x0 ][ y0 ] equal to 1 specifies that the prediction samples are derived using reconstructed samples of the reference layer representation as specified in subclause H.8.8; texture\_rl\_flag[ x0 ][ y0 ] equal to 0 specifies that the current coding unit is coded in either intra prediction or inter prediction mode.

When texture\_rl\_flag[ x0 ][ y0 ] not present, it is inferred to be equal to default\_inter\_layer\_texture\_flag.

When texture\_rl\_flag is equal to 1, the variable PredMode[ x ][ y ] and PartMode[ x ][ y ] are derived as follows.

for  $x = x0..x0 + nCbS - 1$  and  $y = y0..y0 + nCbS - 1$ ,

PredMode[ x ][ y ] = MODE\_TEXTURE\_RL

(H-xx)

PartMode[ x ][ y ] = PART\_2Nx2N

(H-xx)

~~When texture\_rl\_flag[ x0 ][ y0 ] not present, it is inferred to be equal to 0.~~

# 4. Proposed ref pic list construction process for ref index mode

- adaptive\_inter\_layer\_texture\_flag and default\_inter\_layer\_texture\_flag values do not affect the syntax table when the ref\_idx mode is used, but affect the reference picture list construction.
- When the adaptive\_inter\_layer\_texture\_flag is equal to 1, the method of list construction described in the SHM Test Model is used
- When the adaptive\_inter\_layer\_texture\_flag is equal to 0 and default\_inter\_layer\_texture\_flag is equal to 0, the single layer HEVC list construction method is used and the inter-layer reference picture set is not included in the list
- When the adaptive\_inter\_layer\_texture\_flag is equal to 0 and default\_inter\_layer\_texture\_flag is equal to 1, a new list construction method is proposed where only the inter-layer reference picture set is included in the list
- Details provided in contribution document

# Background:

## SVC slice header in scalable extension

slice_header_in_scalable_extension() {	C	Descriptor
...		
if( !no_inter_layer_pred_flag && quality_id == 0 ) {		
<b>ref_layer_dq_id</b>	2	ue(v)
...		
if( !no_inter_layer_pred_flag ) {		
<b>slice_skip_flag</b>	2	u(1)
if( slice_skip_flag )		
<b>num_mbs_in_slice_minus1</b>	2	ue(v)
else {		
<b>adaptive_base_mode_flag</b>	2	u(1)
if( !adaptive_base_mode_flag )		
<b>default_base_mode_flag</b>	2	u(1)
if( !default_base_mode_flag ) {		
<b>adaptive_motion_prediction_flag</b>	2	u(1)
if( !adaptive_motion_prediction_flag )		
<b>default_motion_prediction_flag</b>	2	u(1)
}		
<b>adaptive_residual_prediction_flag</b>	2	u(1)
if( !adaptive_residual_prediction_flag )		
<b>default_residual_prediction_flag</b>	2	u(1)
}		
if( adaptive_tcoeff_level_prediction_flag )		
<b>tcoeff_level_prediction_flag</b>	2	u(1)
}		
...		