Vidyo™
Personal Telepresence

JCTVC-I0230/M2714:
Parameter sets modifications for temporal scalability and extension hooks

Jill Boyce, Wonkap Jang, and Danny Hong

Vidyo

Introduction

- **Motivation for changes to the base specification**
  - Temporal scalability
  - Hooks to enable future extensions

- **Proposed base specification changes**
  - Add Video Parameter Set
    - As proposed in JCTVC-H0388, but with some changes to proposed syntax
  - Modify SPS to contain vps_id
  - Rename reserved_one_5bits syntax in NAL unit header to layer_id_plus1

- **Proposed extension changes**
  - Modifications to the NAL unit header and VPS for future scalability and multiview extensions

Video Parameter Set (VPS)

- **Move temporal scalability related syntax elements from Sequence Parameter Set (SPS) to VPS**
  - max_temporal_layers_minus1
  - temporal_id_nesting_flag
  - max_dec_pic_buffering[i], num_reorder_pics[i], max_latency_increase[i]

- **Motivation**
  - All NAL units in an access unit have the same value of temporal_id
    - "**temporal_id** specifies a temporal identifier for the NAL unit. The value of temporal_id shall be the same for all NAL units of an access unit."
  - Same restriction will likely apply to all layers/views in scalability/multiview extensions
    - All layers/views will refer to same VPS, but have different SPS
    - Avoids need to duplicate in all SPS's
  - In SVC, temporal_id_nesting_flag was in Scalability Information SEI message, which applied to all layers in a coded video sequence

Video Parameter Set (VPS) 2

- **Add fixed length max_layers_minus1 syntax element**
  - Value restricted to 0 in the base spec

- **Move some VUI parameters from SPS to VPS**
  - Discussion of which parameters postponed until general VUI review

- **Add extension flag**

Proposed VPS syntax

video_parameter_set_rbsp( ) {	Descriptor
max_temporal_layers_minus1	u(3)
max_layers_minus1	u(5)
video_parameter_set_id	ue(v)
temporal_id_nesting_flag	u(1)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)
num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

Proposed SPS syntax:

Add video_parameter_set_id, remove temporal scalability params

Vidyo™
Personal Telepresence

seq_parameter_set_rbsp( ) {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
video_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	
~~**max_temporal_layers_minus1**~~	
pic_width_in_luma_samples	
pic_height_in_luma_samples	
if(pcm_enabled_flag)	
pcm_loop_filter_disable_flag	
~~**temporal_id_nesting_flag**~~	~~u(1)~~
if(log2_min_coding_block_size_minus3 == 0)	
inter_4x4_enabled_flag	u(1)
...	

log2_max_pic_order_cnt_lsb_minus4	ue(v)
~~for(i = 0; i <= max_temporal_layers_minus1; i++) {~~	
~~max_dec_pic_buffering[i]~~	~~ue(v)~~
~~num_reorder_pics[i]~~	~~ue(v)~~
~~max_latency_increase[i]~~	~~ue(v)~~
~~}~~	
restricted_ref_pic_lists_flag	u(1)

Proposed Extension Changes

Terminology – as proposed in H0386/H0388

- **A Coded Video Sequence (CVS) contains one or more layers**

- **A Layer is a a spatial, quality, view, or depth layer**
  - Each layer may refer separately to an SPS
  - Each layer uniquely identified by a layer_id

- **Temporal layers are not considered to be layers**
  - Temporal sub-layers are contained within a layer

- **A sub-bitstream is defined similarly as in SVC**
  - A sub-bitstream is derived as a subset from a bitstream by discarding a number of NAL units.

Proposed NAL Unit Header in extension:
Rename reserved_one_5bits syntax element to be layer_id_plus1

nal_unit(NumBytesInNALunit) {	Descriptor
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
if(nal_unit_type == 1 || nal_unit_type == 4 || nal_unit_type == 5) {	
temporal_id	u(3)
~~**reserved_one_5bits**~~	~~u(5)~~
layer_id_plus1	u(5)
nalUnitHeaderBytes += 1	
}	
for(i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
}	
}	

Proposed VPS in extension

- **VPS contains**
  - Mapping of layer_id_plus1 to dependency_id, quality_id, view_id, and depth_flag
  - Dependency relationships of the layers in a bitstream
  - Other information applicable to all slices across all (scalability or view) layers in the entire coded video sequence

- **In MVC, the view dependency information was included in the SPS, and repeated for each view**
  - More appropriate that it be included in the VPS rather than the SPS, because it applies to all views/layers
  - More bitrate efficient because duplication is avoided

Motivations

- **Backwards compatible with existing HEVC design**
  - 5-bit layer_id_plus1 field never equal to '00000'
    - Avoids problems with start code emulation, MPEG-2 systems mapping
  - temporal_id maintained
- **Allows 31 active layers, with flexible allocation to spatial, SNR, view, depth layers**
  - Allows combination of temporal, spatial, SNR, view scalability
- **Application specifications or individual profiles could allocate bits to specific fields, while maintaining flexibility in the standard**
  - Example: 2 bits for view_id, 1 bit for depth_flag, 1 bits for dependency_id
  - Have 4 bits to allocate, not 5, in order to avoid value of '00000'
- **Simple "middle box" could perform bitstream extraction based on layer_id and temporal_id, more sophisticated "middle box" could consider VPS mappings**

Proposed VPS in extension

video_parameter_set_rbsp( ) {	Descriptor
max_temporal_layers_minus1	u(3)
max_layers_minus1	u(5)
temporal_id_nesting_flag	u(1)
video_parameter_set_id	ue(v)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)
num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
bit_equal_to_one /* equal to 1 */	u(1)
if(max_layers_minus1 > 0) {	
extension_type	ue(v)
for(i = 1; i < max_layers_minus1; i++) {	
dependent_flag[i]	u(1)
if (dependent_flag[i]) {	
delta_reference_layer_id_minus1[i]	ue(v)
if(extension_type = = 0 || extension_type = = 2) {	
dependency_id[i]	ue(v)
quality_id[i]	ue(v)
}	
if(extension_type = = 1 || extension_type = = 2) {	
view_id[i]	ue(v)
depth_flag[i]	u(1)
}	
}	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

In base spec:

video_parameter_set_rbsp( ) {	Descriptor
max_temporal_layers_minus1	u(3)
max_layers_minus1	u(5)
video_parameter_set_id	ue(v)
temporal_id_nesting_flag	u(1)
for(i = 0; i <= max_temporal_layers_minus1; i++) {	
max_dec_pic_buffering[i]	ue(v)
num_reorder_pics[i]	ue(v)
max_latency_increase[i]	ue(v)
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

Conclusions

- **Proposed design is coding efficient and flexible**
- **Enables "middle box" to perform bitstream extraction**