JCTVC-L0226: VPS AND VPS_EXTENSION UPDATES

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application claims priority to the U.S. Provisional Patent Application Ser. No. 61/748,893, filed January 4, 2013 and titled, "JCTVC-L0XXX: VPS AND VPS_EXTENSION UPDATES"; and U.S. Patent Application Ser. No. 14/035,711, filed September 24, 2013 and titled "JCTVC-L0226: VPS AND VPS_EXTENSION UPDATES". Each of the above referenced applications is hereby incorporated herein by reference in its entirety for all purposes.

FIELD OF THE INVENTION

The present invention relates to the field of video encoding. More specifically, the present invention relates to high efficiency video coding.

BACKGROUND OF THE INVENTION

The Video Parameter Set (VPS) has been added as metadata to describe the overall characteristics of coded video sequences, including the dependencies between temporal sublayers. The primary purpose of this is to enable the compatible extensibility of the standard in terms of signaling at the systems layer, e.g., when the base layer of a future extended scalable or multiview bitstream would need to be decodable by a legacy decoder, but for which additional information about the bitstream structure that is only relevant for the advanced decoder would be ignored.

SUMMARY OF THE INVENTION

The VPS and vps_extension( ) syntax structures are updated with some cleanups for the HEVC Extensions in scalable video coding, multi-view coding and 3D video coding areas. In addition, four options of adding syntaxes to support mixed video sequences in various layers for the VPS extension are described.

According to a first aspect of the present invention, there is provided a method for information processing a device comprising: decoding a bit stream based on video parameter set (VPS) syntax structure wherein a byte-alignment syntax is under a condition of a VPS extension flag in the VPS syntax structure.

According to a second aspect of the present invention, there is provided an information processing apparatus comprising: a decoder for decoding a bit stream based on video parameter set (VPS) syntax structure wherein a byte-alignment syntax is under a condition of a VPS extension flag in the VPS syntax structure.

In one embodiment, a method programmed in a non-transitory memory of a device comprises decoding content and accessing information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information. A video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag. Byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag. The video parameter set data includes a raw byte sequence payload trailing bits value. The video parameter set data is determined without using byte alignment syntaxes. The video parameter set data is determined using two syntaxes for mixed sequence signaling support. The video parameter set data is determined using a source mixed codec flag syntax parameter. The video parameter set data is determined using a source mixed video present flag syntax parameter. The video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter. The method further comprises encoding the content. The device comprises a personal computer, a laptop computer, a computer workstation, a server, a mainframe computer, a handheld computer, a personal digital assistant, a cellular/mobile telephone, a smart appliance, a gaming console, a digital camera, a digital camcorder, a camera phone, a smart phone, a portable music player, a tablet computer, a mobile device, a video player, a video disc writer/player, a high definition disc writer/player, an ultra high definition disc writer/player), a television, a home entertainment system, or a smart watch.

In another embodiment, a method programmed in a non-transitory memory of a device comprises providing content and enabling access of information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information. A video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag. Byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag. The video parameter set data includes a raw byte sequence payload trailing bits value. The video parameter set data is determined without

using byte alignment syntaxes. The video parameter set data is determined using two syntaxes for mixed sequence signaling support. The video parameter set data is determined using a source mixed codec flag syntax parameter. The video parameter set data is determined using a source mixed video present flag syntax parameter.

The video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter. The method further comprises encoding the content. The device comprises a personal computer, a laptop computer, a computer workstation, a server, a mainframe computer, a handheld computer, a personal digital assistant, a cellular/mobile telephone, a smart appliance, a gaming console, a digital camera, a digital camcorder, a camera phone, a smart phone, a portable music player, a tablet computer, a mobile device, a video player, a video disc writer/player, a high definition disc writer/player, an ultra high definition disc writer/player), a television, a home entertainment system, or a smart watch.

In another embodiment, an apparatus comprises a non-transitory memory for storing an application, the application for decoding content and accessing information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information and a processing component coupled to the memory, the processing component configured for processing the application. A video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag. Byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag. The video parameter set data includes a raw byte sequence payload trailing bits value. The video parameter set data is determined without using byte alignment syntaxes. The video parameter set data is determined using two syntaxes for mixed sequence signaling support. The video parameter set data is determined using a source mixed codec flag syntax parameter. The video parameter set data is determined using a source mixed video present flag syntax parameter. The video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter. The apparatus further comprises encoding the content.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a flowchart of a method of utilizing a modified VPS syntax according to some embodiments.

FIG. 2 illustrates a block diagram of an exemplary computing device configured to implement the modified VPS method according to some embodiments.

FIG. 3 illustrates a general diagram of an HEVC encoder according to some embodiments.

FIG. 4 illustrates a general diagram of an HEVC decoder according to some embodiments.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The Video Parameter Set (VPS) syntax structure in the current Draft International Standard (DIS) specification for High Efficiency Video Coding (HEVC) is shown below.

video_parameter_set_rbsp() {	Descriptor
vps_video_parameter_set_id	u(4)
vps_reserved_three_2bits	u(2)
vps_reserved_zero_6bits	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
bit_rate_pic_rate_info(0, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)

for (i= (vps_sub_layer_ordering_info_present_flag ?
0 : vps_max_sub_layers_minus1);
i <= vps_max_sub_layers_minus1; i++) {	

vps_max_dec_pic_buffering[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase[i]	ue(v)
}	
vps_max_nuh_reserved_zero_layer_id	u(6)
vps_num_op_sets_minus1	

for(i=1;i<vps_num_op_sets_minus1;i++)	
operation_point_set(i)	
vps_num_hrd_parameters	ue(v)
for(i=0;i<num_hrd_parameters;i++) {	
hrd_op_set_idx[i]	ue(v)
if(i>0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

The if(vps_extension_flag) section above is updated below to support various HEVC extensions.

video_parameter_set_rbsp() {	Descriptor
vps_video_parameter_set_id	u(4)
vps_reserved_three_2bits	u(2)
vps_max_layers_minus1 //vps_reserved_zero_6bits in version-1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
next_essential_info_byte_offset // vps_reserved_0xffff_16bits in version-1	u(16)

profile_tier_level(1, vps_max_sub_layers_minus1)	
bit_rate_pic_rate_info(0, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)

for (i= (vps_sub_layer_ordering_info_present_flag ?
0 : vps_max_sub_layers_minus1);
i <= vps_max_sub_layers_minus1; i++) {	

vps_max_dec_pic_buffering[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase[i]	ue(v)
}	
vps_max_nuh_reserved_zero_layer_id	u(6)
vps_num_op_sets_minus1	
for(i=1;i<vps_num_op_sets_minus1;i++)	
operation_point_set(i)	
vps_num_hrd_parameters	ue(v)
for(i=0;i<num_hrd_parameters;i++) {	
hrd_op_set_idx[i]	ue(v)
if(i>0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
vps_extension_flag /bit_equal_to_one in JCTVC-K1007_v1	u(1)
if(vps_extension_flag) {	
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)

vps_extension( )	
}	
rbsp_trailing_bits( )	
}	

The updates include:

1) The vps_reserved_zero_6bits in version 1 is changed to vps_max_num_layers_minus1. However, in the vps_extension( ) syntax structure, this parameter is used as vps_max_layers_minus1. This is also consistent with a similar parameter vps_max_sub_layers_minus1 in version-1. Thus, vps_max_layers_minus1 is used in VPS.

2) For HEVC extensions, the "vps_extension( )" syntax structure as defined previously is inserted in VPS under the condition of vps_extension_flag.

3) The vps_reserved_0xffff_16bits in version-1 is replaced by next_essential_info_byte_offset. This new syntax parameter is able to be used to locate the esential information available in "vps_extension( )" syntax structure.

4) The syntax parameter next_essential_info_byte_offset also includes the byte-alignment related syntaxes near the top of the "vps_extension( )" syntax structure. These byte-alignment syntaxes are taken out and placed inside the VPS under the condition of vps_extension_flag, just before the "vps_syntax( )" structure, for cleanliness.

5) The "rbsp_trailing_bits" contain a number of byte alignment bits for the VPS_rbsp( ), and such bit-counts are able to be determined as follows: nByteAlignmentBits = ((vps_extension_length + 7) / 8) * 8 - vps_extension length, where vps_extension_length is the total number of bits in vps_extension( ), and these are able to be computed easily as vps_extension( ) syntax structure so far contains fixed-length unsigned integer bits.

The syntax structure of vps_extension( ) is shown below:

vps_extension() {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_codec_flag	u(1)

scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=1; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension Ids	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
num_dimensions_minus1[i]	u(4)
for (j=0; j<=num_dimensions_minus1; j++) {	
dimension_type[i][j]	u(4)
dimension_id[i][j]	u(v8)
}	
}	
for(i=1;i<vps_max_layers_minus1;i++) {	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=1;i<vps_max_layers_minus1;i++) {	
//layer dependency	
num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)
}	

}	

While reviewing the above syntax structure and semantics description, a number of issues are discussed below as cleanups or for editing purposes.

For the scalability_mask syntax parameter, the following parameter table is listed, while the corresponding entries show the bit-location for the register-type data of "scalability_mask," but it is not described well.

Scalability_mask	Scalability dimension
0	none (base HEVC)
1	spatial
2	quality

3	depth
4	multiview
5	unspecified
6-15	Reserved

In place of the above table for scalability_mask, the following register-type description is better for a clearer understanding:

5	4	3	2	1	0	Bit-positions
unspecified	multi-view	depth	quality	spatial	none	scalability dimension

15	14	13	12	11	10	9	8	7	6
reserved									

Also, the modified syntax structure of vps_extension( ) shown below, indicates some helpful cleanups.

vps_extension( ) {	Descriptor
avc_base_codec_flag	u(1)
scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=0; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension IDs	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for (j=0; j<=NumScalabilityTypes; j++) {	
v = dimension_id_len_minus1[j]+1	
dimension_id[i][j]	u(v)
}	
}	
for(i=1;i<vps_max_layers_minus1;i++)	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=0;i<vps_max_layers_minus1;i++) {	
//layer dependency	

num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)
}	
}	

1) The byte-alignments-related syntaxes are removed.

2) The undefined parameter "num_dimensions_minus1" is replaced by NumScalabilityTypes."

3) Two of the "for" loops have been modified to start from 0 instead of 1. As an example, for "layer specific info," the "for(i=1; i<=vps_max_layers_minus1; i++){...}" loop, if a coded video sequence containing stereo-view with depth-view (2+1=3 layers), there are 3 layers. So, vps_max_layers_minus1 = 3 - 1 = 2. If the current "for" loop is followed,

i = 1 => Layer-1 = depth-view;

i = 2 (vps_max_layers_minus1) => Layer-2 = View-1

where is i=3 => Layer-3 = View-2?

As a correction, this loop is "for(i=0; i<=vps_max_layers_minus1; i++){...}" for two cases, except for the "profile_tier_level(...)."

4) The semantics description for the syntax parameter "dimension_id[i][j]" is also not available with respect to "scalability dimension" and "dimension_id_len_minus1[j]." An exemplary table is shown below to show the relationship among these parameters:

Scalability dimension	dimension_id	dimension_id_len(example)

none(base HEVC)	-	-
spatial	dependency_id	4
quality	quality_id	4
depth	depth_flag	1
multiview	view_idx	4
unspecified	-	-

Reserved	-	-

The VPS and vps_extension( ) syntax structures are updated for the HEVC Extensions in scalable video coding, multi-view coding and 3D video coding areas.

VPS contains metadata to describe the overall characteristics of coded video, enables standards extension compatibility in terms of systems layer signaling so that a legacy decoder is able to ignore additional information about the extension bitstream structure. The current VPS syntax structure and its extension contain information for multi-layered video sequences, where a layer is able to be a base layer (for HEVC and its extensions), or is able to be a scalable enhancement layer (spatial or quality) for SHVC, or a view layer (texture view or depth view) for multi-view coding of HEVC extension (MHVC). All such layers are able to have their sub-layers also (temporal). Each video layer is able to be depending upon its neighboring lower layers or it is able to be independent with no inter-layer predictions.

The motivation behind "mixed sequences signaling support" is to develop new layer-specific properties in the VPS extension syntax structure, as suggested in "JCTVC-K_notes." The layered syntax agreement for independent layers in VPS are able to be suitably exploited to support a variety of mixed contents (source-types, coding types and others) in the high level syntax structure for HEVC extensions.

Some possible application examples in mixed broadcasting or network transitions for a VPS with 2-layers are:

Frame structured video in layer-1 and field structured video in layer-2: each video layer has its respective SPS and more. Legacy mon-view (2-D) decoders are able to user either of these 2 layers for its base-layer application.

Frame structured video (2-D, mon-view) in layer-1 and frame-compatible (frame-packing) video (3D, stereo-view) in layer-2: legacy mono-view (2-D) decoder uses the layer-1 video, while an advanced stereo-view (3D) frame-compatible video decoder is able to use layer-2 for 3D stereo-view video application.

The combination of mixed video types will be present in the video bitstreams to

support two such decoders (legacy and advanced).

Many such combinations of coded video content with varied source-types and coding types are able to co-exist in the current and future applications where either legacy or advanced decoders will sort out the respective bitstream for decoding.

The inherent layer-based descriptions of VPS are able to be extended to support such applications, at least for the systems level solutions (ad-insertions, splicing and more).

There are four possible options of adding syntax for mixed sequences signaling in VPS extension.

Option 1: 2 syntaxes for mixed sequences signaling support

Option 2: Option 1 plus source_mixed_codec_flag syntax parameter

Option 3: Option 2 plus mixed_video_present_flag syntax parameter

Option 4: specific applications support plus mixed_video_present_flag syntax parameter and use it as a condition for added syntax/semantics for this option.

Option 1

vps_extension( ) {	Descriptor
avc_base_codec_flag	u(1)
source_mixed_video_type	u(4)
scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=0; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension IDs	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)

for (j=0; j<=NumScalabilityTypes; j++) {	
v = dimension_id_len_minus1[j]+1	
dimension_id[i][j]	u(v)
}	
if(source_mixed_video_type > 0)	
source_mixed_video_idc[i]	u(4)
}	
for(i=0;i<vps_max_layers_minus1;i++)	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=0;i<vps_max_layers_minus1;i++) {	
//layer dependency	
num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)
}	
}	

The semantics description for these two added syntax parameters are shown in the tables below:

source_mixed_video_type	Mixed video combination sources (1st/2nd/3rd...)

0	None
1	Frame / Field
2	Frame / Frame-Compatible
3	Field / Frame-Compatible
4	Frame-Compatible / Stereo-view
5	Frame / Field / Frame-Compatible

6	Frame / Frame-Compatible / Stereo-view
7...15	Undefined

source_mixed_video_idc	Mixed video combination order
0	First source-video-type present in Layer-i
1	Second source-video-type present in Layer-i
2	A mixture of First and Second source-video-types present in Layer-i
3	Third source-video-type present in Layer-i
4	A mixture of First, Second and Third source-video-types present in Layer-i
5...15	Undefined

Option 2

vps_extension( ) {	Descriptor
avc_base_codec_flag	u(1)
source_mixed_video_type	u(4)
scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	

dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=0; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension IDs	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for (j=0; j<=NumScalabilityTypes; j++) {	
v = dimension_id_len_minus1[j]+1	
dimension_id[i][j]	u(v)
}	
if(source_mixed_video_type > 0) {	
source_mixed_video_idc[i]	u(4)
source_mixed_codec_flag[i]	u(1)
}	
}	
for(i=0;i<vps_max_layers_minus1;i++)	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=0;i<vps_max_layers_minus1;i++) {	
//layer dependency	
num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)

}	
}	

The semantics description for the added new syntax is the same avc_base_codec_flag as follow:

avc_base_codec_flag / source_mixed_codec_flag	Mixed video coding-type
0	HEVC
1	AVC

Option 3

vps_extension( ) {	Descriptor
avc_base_codec_flag	u(1)
mixed_video_present_flag	u(1)
if (mixed_video_present_flag)	
source_mixed_video_type	u(4)
scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=0; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension IDs	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)

for (j=0; j<=NumScalabilityTypes; j++) {	
v = dimension_id_len_minus1[j]+1	
dimension_id[i][j]	u(v)
}	
if(mixed_video_present_flag) {	
source_mixed_video_idc[i]	u(4)
source_mixed_codec_flag[i]	u(1)
}	
}	
for(i=0;i<vps_max_layers_minus1;i++)	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=0;i<vps_max_layers_minus1;i++) {	
//layer dependency	
num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)
}	
}	

mixed_video_present_flag syntax parameter equals 1 indicates presence of mixed video in VPS layers. If 0, no such mixed sequences are present in the layers.

The presence of this flag is able to save some added bits in case of "no mixed video-type."

source_mixed_video_type	Mixed video combination sources (1st/2nd/3rd...)
0	Frame / Field
1	Frame / Frame-Compatible
2	Field / Frame-Compatible
3	Frame-Compatible / Stereo-view
4	Frame / Field / Frame-Compatible

5	Frame / Frame-Compatible / Stereo-view
6...15	Undefined

Option 4

vps_extension( ) {	Descriptor
avc_base_codec_flag	u(1)
mixed_video_present_flag	u(1)
scalability_mask	u(16)
for(i=0;i<NumScalabilityTypes;i++) {	
dimension_id_len_minus1[i]	u(3)
}	
vps_nuh_layer_id_present_flag	u(1)
//layer specific information	
for (i=0; i<=vps_max_layers_minus1; i++) {	
//mapping of layer ID to scalability dimension IDs	
if (vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
for (j=0; j<=NumScalabilityTypes; j++) {	
v = dimension_id_len_minus1[j]+1	

dimension_id[i][j]	u(v)
}	
if(mixed_video_present_flag) {	
source_scan_type_info_idc[i]	u(4)
source_2d_3d_info_idc[i]	u(1)
}	
}	
for(i=0;i<vps_max_layers_minus1;i++)	
profile_tier_level(1,vps_max_sub_layers_minus1)	
for(i=0;i<vps_max_layers_minus1;i++) {	
//layer dependency	
num_direct_ref_layers[i]	u(6)
for(j=0; j<num_direct_ref_layers[i];j++)	
ref_layer_id[i][j]	u(6)
}	
}	

mixed_video_present_flag syntax parameter equals 1 indicates presence of mixed video in VPS layers. If 0, no such mixed sequences are present in the layers.

In case of "mixed_video_present_flag = 1," two new syntaxes are present for each layer and these two syntax parameters address two specific application examples with mixed video sources:

Frame / Field structured scan-types and Frame for 2D video, and

Frame-packing arrangement (FPA) video for 3D video.

The semantics description for these two new syntaxes are explained in the tables below:

source_scan_type_info_idc[i]	Video Layer Characteristics
0	All pictures in layer i have interlaced scan type
1	All pictures in layer i have progressive scan type
2	Pictures in layer i have unknown scan type
3	Pictures in layer i have a mixture of interlaced and progressive scan type

source_2d_3d_info_idc[i]	Video Layer Characteristics
0	All pictures in layer i are 2D frames
1	All pictures in layer i are packed in frame compatible 3D format or frame packing arrangement (FPA)
2	Pictures in layer i have unknown FPA
3	Pictures in layer i have a mixture of frame compatible 3D and 2D frames

The syntax modifications described herein extend the layer-based properties of VPS to address various emerging applications in the System layers (content editing, splicing, ad-insertion) as well as in the overall decoding path for better communication and system integration purposes.

Figure 1 illustrates a flowchart of a method of utilizing a modified VPS syntax according to some embodiments. In the step 100, content (e.g., a video) is encoded. The VPS data generated using the modified VPS syntax is also included with the encoded content. In the step 102, decoding is implemented. For example, a video is decoded using any decoder. In the step 104, the VPS data is accessible. While decoding the VPS data is able to be used for additional processing. The VPS data contains various HEVC Extension information in scalable video coding, multi-view coding and 3D video coding areas. In some embodiments, the VPS data includes mixed signaling information. In some embodiments, additional or fewer steps are implemented. In some embodiments, the order of the steps is modified.

Figure 2 illustrates a block diagram of an exemplary computing device configured to implement the modified VPS method according to some embodiments. The computing device 200 is able to be used to acquire, store, compute, process, communicate and/or display information such as images and videos. In general, a hardware structure suitable for implementing the computing device 200 includes a network interface 202, a memory 204, a processor 206, I/O device(s) 208, a bus 210 and a storage device 212. The choice of processor is not critical as long as a suitable processor with sufficient speed is chosen. The memory 204 is able to be any conventional computer memory known in the art. The storage device 212 is able to include a hard drive, CDROM, CDRW, DVD, DVDRW, Blu-ray®, flash memory card or any other storage device. The computing device 200 is able to include one or more network interfaces 202. An example of a network interface includes a network card connected to an Ethernet or other type of LAN. The I/O device(s) 208 are able to include one or more of the following: keyboard, mouse, monitor, screen, printer, modem, touchscreen, button interface and other devices. Modified VPS application(s) 230 used to perform the modified VPS method are likely to be stored in the storage device 212 and memory 204 and processed as applications are typically processed. More or fewer components shown in Figure 2 are able to be included in the computing device 200. In some embodiments, modified VPS hardware 220 is included. Although the computing device 200 in Figure 2 includes applications 230 and hardware 220 for the modified VPS method, the modified VPS method is able to be implemented on a computing device in hardware, firmware, software or any combination thereof. For example, in some embodiments, the modified VPS applications 230 are programmed in a memory and executed using a processor. In another example, in some embodiments, the modified VPS hardware 220 is programmed hardware logic including gates specifically designed to implement the modified VPS method.

In some embodiments, the modified VPS application(s) 230 include several applications and/or modules. In some embodiments, modules include one or more sub-modules as well. In some embodiments, fewer or additional modules are able to be included.

Examples of suitable computing devices include a personal computer, a laptop computer, a computer workstation, a server, a mainframe computer, a handheld computer, a personal digital assistant, a cellular/mobile telephone, a smart appliance, a gaming console, a digital camera, a digital camcorder, a camera phone, a smart phone, a portable music

player, a tablet computer, a mobile device, a video player, a video disc writer/player (e.g., DVD writer/player, high definition disc writer/player, ultra high definition disc writer/player), a television, a home entertainment system, smart jewelry (e.g., smart watch) or any other suitable computing device.

Figure 3 illustrates a general diagram of an HEVC encoder according to some embodiments. The encoder 300 includes a general coder control component, a transform scaling and quantization component, a scaling and inverse transform component, an intra-picture estimation component, a filter control analysis component, an intra-picture prediction component, a deblocking and SAO filters component, a motion compensation component, a motion estimation component, and a header formatting and CABAC component. An input video signal is received by the encoder 300 and is split into Coding Tree Units (CTUs). The HEVC encoder components process the video data using the modified VPS and generate a coded bitstream.

Figure 4 illustrates a general diagram of an HEVC decoder according to some embodiments. The decoder 400 includes an entropy decoding component, an inverse quantization component, an inverse transform component, a current frame component, an intra prediction component, a previous frames component, a motion compensation component, a deblocking filter, an SAO component and an adaptive loop filter. An input bitstream (e.g., a coded video) is received by the decoder 400, and a decoded bitstream is generated for display. In some embodiments, the modified VPS is utilized while decoding.

To utilize the modified VPS method, devices are able to access parameters in VPS and its extension for scalable video coding, multi-view coding, 3D video coding and mixed video sequences. The modified VPS method is automatically used when performing video processing or other times. The modified VPS method is able to be implemented automatically without user involvement.

In operation, the VPS and vps_extension( ) syntax structures are updated with some cleanups for the HEVC Extensions in scalable video coding, multi-view coding and 3D video coding areas. In addition, four options of adding syntaxes to support mixed video sequences in various layers for the VPS extension are described. The VPS is generated using the modifed syntax structure.

U.S. Patent Application No. 14/035,680, filed September 24, 2013, titled "VIDEO PARAMETER SET (VPS) SYNTAX RE-ORDERING FOR EASY ACCESS OF EXTENSION PARAMETERS" and U.S. Patent Application No. 14/035,740, filed

September 24, 2013, titled "JCTVC-L0227: VPS_EXTENSION WITH UPDATES OF PROFILE-TIER-LEVEL SYNTAX STRUCTURE" are hereby incorporated by reference in their entireties for all purposes.

SOME EMBODIMENTS OF JCTVC-L0226: VPS AND VPS_EXTENSION UPDATES

1. A method programmed in a non-transitory memory of a device comprising:
   a. decoding content; and
   b. accessing information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information.

2. The method of clause 1 wherein a video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag.

3. The method of clause 1 wherein byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag.

4. The method of clause 1 wherein the video parameter set data includes a raw byte sequence payload trailing bits value.

5. The method of clause 1 wherein the video parameter set data is determined without using byte alignment syntaxes.

6. The method of clause 1 wherein the video parameter set data is determined using two syntaxes for mixed sequence signaling support.

7. The method of clause 6 wherein the video parameter set data is determined using a source mixed codec flag syntax parameter.

8. The method of clause 7 wherein the video parameter set data is determined using a source mixed video present flag syntax parameter.

9. The method of clause 1 wherein the video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter.

10. The method of clause 1 further comprising encoding the content.

11. The method of clause 1 wherein the device comprises a personal computer, a laptop computer, a computer workstation, a server, a mainframe computer, a handheld computer, a personal digital assistant, a cellular/mobile telephone, a smart appliance, a gaming console, a digital camera, a digital camcorder, a camera phone, a smart phone, a portable music player, a tablet computer, a mobile device, a video player, a video disc writer/player, a high definition disc writer/player, an ultra high definition disc writer/player), a television, a home entertainment system, or a smart watch.

12. A method programmed in a non-transitory memory of a device comprising:
    a. providing content; and
    b. enabling access of information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information.

13. The method of clause 12 wherein a video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag.

14. The method of clause 12 wherein byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag.

15. The method of clause 12 wherein the video parameter set data includes a raw byte sequence payload trailing bits value.

16. The method of clause 12 wherein the video parameter set data is determined without using byte alignment syntaxes.

17. The method of clause 12 wherein the video parameter set data is determined using two syntaxes for mixed sequence signaling support.

18. The method of clause 17 wherein the video parameter set data is determined using a source mixed codec flag syntax parameter.

19. The method of clause 18 wherein the video parameter set data is determined using a source mixed video present flag syntax parameter.

20. The method of clause 12 wherein the video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter.

21. The method of clause 12 further comprising encoding the content.

22. The method of clause 12 wherein the device comprises a personal computer, a laptop computer, a computer workstation, a server, a mainframe computer, a handheld computer, a personal digital assistant, a cellular/mobile telephone, a smart appliance, a gaming console, a digital camera, a digital camcorder, a camera phone, a smart phone, a portable music player, a tablet computer, a mobile device, a video player, a video disc writer/player, a high definition disc writer/player, an ultra high definition disc writer/player), a television, a home entertainment system, or a smart watch.

23. An apparatus comprising:
    a. a non-transitory memory for storing an application, the application for:
        i. decoding content; and
        ii. accessing information related to the content, wherein the information comprises video parameter set data, further wherein the video parameter set data comprises mixed signaling information; and
    b. a processing component coupled to the memory, the processing component configured for processing the application.

24. The apparatus of clause 23 wherein a video parameter set function used in determining the video parameter set data is under a condition of a video parameter set extension flag.

25. The apparatus of clause 23 wherein byte-alignment syntaxes used in determining the video parameter set data are under a condition of a video parameter set extension flag.

26. The apparatus of clause 23 wherein the video parameter set data includes a raw byte sequence payload trailing bits value.

27. The apparatus of clause 23 wherein the video parameter set data is determined without using byte alignment syntaxes.

28. The apparatus of clause 23 wherein the video parameter set data is determined using two syntaxes for mixed sequence signaling support.

29. The apparatus of clause 28 wherein the video parameter set data is determined using a source mixed codec flag syntax parameter.

30. The apparatus of clause 29 wherein the video parameter set data is determined using a source mixed video present flag syntax parameter.

31. The apparatus of clause 23 wherein the video parameter set data is determined using specific application support and a source mixed video present flag syntax parameter.

32. The apparatus of clause 23 further comprising encoding the content.

The present invention has been described in terms of specific embodiments incorporating details to facilitate the understanding of principles of construction and operation of the invention. Such reference herein to specific embodiments and details thereof is not intended to limit the scope of the claims appended hereto. It will be readily apparent to one skilled in the art that other various modifications may be made in the embodiment chosen for illustration without departing from the spirit and scope of the invention as defined by the claims.

CLAIMS

1. A method for information processing a device comprising:

decoding a bit stream based on video parameter set (VPS) syntax structure wherein a byte-alignment syntax is under a condition of a VPS extension flag in the VPS syntax structure.

2. The method of claim 1 wherein a VPS function is under a condition of a VPS extension flag in VPS syntax structure.

3. The method of claim 1 wherein VPS extension function is present after byte-alignment syntaxes in VPS syntax structure.

4. An information processing apparatus comprising:

a decoder for decoding a bit stream based on video parameter set (VPS) syntax structure wherein a byte-alignment syntax is under a condition of a VPS extension flag in the VPS syntax structure.

5. The apparatus of claim 4 wherein a VPS function is under a condition of a VPS extension flag in VPS syntax structure.

6. The apparatus of 5 wherein VPS extension function is present after byte-alignment syntaxes in VPS syntax structure.