|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, SE, 11 – 20 July 2012 | Document: JCTVC-J0114 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG10: Video parameter set HEVC base specification** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Ying Chen Ye-Kui Wang  5775 Morehouse Drive San Diego, CA 92121 USA | Tel: Email: | 1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

Video parameter set (VPS) was adopted into HEVC, and includes mainly sequence-level temporal scalability related information. This document proposes a changed VPS syntax as well as the corresponding changes in SPS (including VUI) and slice header syntaxes, to enable the use of VPS in session negotiation as well as to reduce the number of bits needed for the representation of SPSs. It is asserted that since SPSs in many application scenarios are transmitted out-of-band, which means the smaller the overall size of all the SPSs the shorter the initial delay, as out-of-band transmission is reliable at the cost of increased initial delay in error-prone environment. An example design of the VPS for future extensions based on the VPS proposed in this document is included in JCTVC-J0124.

# Introduction

Currently video parameter set (VPS) contains mainly sequence-level temporal scalability related information, with the following syntax:

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **vps\_max\_temporal\_layers\_minus1** | u(3) |
| **vps\_max\_layers\_minus1** | u(5) |
| **video\_parameter\_set\_id** | ue(v) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| for( i = 0; i <= vps\_max\_temporal\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **vps\_num\_reorder\_pics[** i **]** | ue(v) |
| **vps\_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( ) |  |
| } |  |

A significant amount of the information contained in SPSs (including VUI) is typically the same for all the layers or views. When all the information is exactly the same, then per the current design, two copies would be needed, for the base layer/view, one for enhancement layers/views (as they can share the same SPS then). If any part of the SPS information changes from one enhancement layer to another enhancement layer, then the rest of the SPS information needs also to repeated once more. SPSs in many application scenarios are transmitted out-of-band, which means the smaller the overall size of all the SPSs the shorter the initial delay, as out-of-band transmission is reliable at the cost of increased initial delay in error-prone environment, wherein whenever there is a loss then retransmission would have to be applied. If signalled in-band in broadcast or multicast applications, repetitions of SPSs would be needed at RAP access units to enable tune-in.

We thus propose a modified design to avoid repetition of SPS information. The proposed design includes all information that is essential for session negotiation to be easily accessible in two ways. Firstly, such information is gathered together and put as early as possible in the VPS. Secondly, such information can be accessed without the need of entropy decoding.

The proposed syntax and semantics are described below.

# Proposal

## Syntax

### Video parameter set RBSP syntax

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **vps\_max\_temporal\_layers\_minus1** | u(3) |
| **vps\_max\_layers\_minus1** | u(5) |
| profile\_level\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| **video\_parameter\_set\_id** | u(5) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| rep\_format\_info( 0, 0 ) |  |
| bitrate\_framerate\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| **next\_essential\_info\_byte\_offset** | u(12) |
| rep\_format\_info( 0, 1 ) |  |
| for( i = 0; i <= vps\_max\_temporal\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **vps\_num\_reorder\_pics[** i **]** | ue(v) |
| **vps\_max\_latency\_increase[** i **]** | ue(v) |
| } |  |
| hrd\_info( 0, vps\_max\_temporal\_layers\_minus1 ) |  |
| vui\_vps\_set ( 0 ) |  |
| **num\_vps\_short\_term\_ref\_pic\_sets** | ue(v) |
| for( i = 0; i < num\_vps\_short\_term\_ref\_pic\_sets; i++ ) |  |
| short\_term\_ref\_pic\_set( i ) |  |
| optional\_sps\_parameters( ) |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

#### Profile and level information table syntax

|  |  |
| --- | --- |
| profile\_level\_info( index, NumTempLevelMinus1 ) { |  |
| **profile\_space** | u(3) |
| **profile\_idc** | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **profile\_compatability\_flag[** I **]** | u(1) |
| **constraint\_flags** | u(16) |
| **level\_idc** | u(8) |
| **level\_lower\_temporal\_layers\_present\_flag** | u(1) |
| if( level\_lower\_temporal\_layers\_present\_flag ) |  |
| for ( i = 0; i < NumTempLevelMinus1; i++ ) |  |
| **level\_idc**[ i ] | u(8) |
| profileLevelInfoIdx = index |  |
| } |  |

#### Representation format information table syntax

|  |  |
| --- | --- |
| rep\_format\_info( index, partIdx ) { |  |
| if( !partIdx ){ |  |
| **chroma\_format\_idc** | u(2) |
| if( chroma\_format\_idc = = 3 ) |  |
| **separate\_colour\_plane\_flag** | u(1) |
| **bit\_depth\_luma\_minus8** | u(2) |
| **bit\_depth\_chroma\_minus8** | u(2) |
| **pic\_width\_in\_luma\_samples** | u(16) |
| **pic\_height\_in\_luma\_samples** | u(16) |
| } |  |
| else { |  |
| **pic\_cropping\_flag** | u(1) |
| if( pic\_cropping\_flag ) { |  |
| **pic\_crop\_left\_offset** | ue(v) |
| **pic\_crop\_right\_offset** | ue(v) |
| **pic\_crop\_top\_offset** | ue(v) |
| **pic\_crop\_bottom\_offset** | ue(v) |
| } |  |
| **}** |  |
| repFormatInfoIdx = index |  |
| } |  |

#### Bitrate and frame rate information table syntax

|  |  |
| --- | --- |
| bitrate\_framerate\_info( TempLevelLow, TempLevelHigh ){ |  |
| **for( i = TempLevelLow; i <= TempLevelHigh; i++ ) {** |  |
| **bitrate\_info\_present\_flag[ i ]** | u(1) |
| **frm\_rate\_info\_present\_flag[ i ]** | u(1) |
| **if( bitrate\_info\_present\_flag[ i ] ) {** |  |
| **avg\_bitrate**[ i ] | u(16) |
| **max\_bitrate** [ i ] | u(16) |
| } |  |
| if( frm\_rate\_info\_present\_flag[ i ] ) { |  |
| **constant\_frm\_rate\_idc**[ i ] | u(2) |
| **avg\_frm\_rate**[ i ] | u(16) |
| } |  |
| } |  |
| } |  |

#### HRD information table syntax

|  |  |
| --- | --- |
| hrd\_info( TempLevelLow, TempLevelHigh ) { |  |
| for ( i = TempLevelLow, nalHrdPresent = 0, vclHrdPresent = 0;  i <= NumTempLevelMinus1; i++ ) { |  |
| **nal\_hrd\_parameters\_present\_flag[** i **]** | u(1) |
| if( nal\_hrd\_parameters\_present\_flag[ i ] ) { |  |
| hrd\_parameters( nalHrdPresent ) |  |
| nalHrdPresent++ |  |
| } |  |
| **vcl\_hrd\_parameters\_present\_flag[** i **]** | u(1) |
| if( vcl\_hrd\_parameters\_present\_flag[ i ] ) { |  |
| hrd\_parameters( vclHrdPresent ) |  |
| vclHrdPresent++ |  |
| } |  |
| if( nalHrdPresent + vclHrdPresent = = 1 ) { |  |
| **low\_delay\_hrd\_flag** | u(1) |
| **sub\_pic\_cpb\_params\_present\_flag** | u(1) |
| **num\_units\_in\_sub\_tick** | u(32) |
| } |  |
|  |  |
| } |  |

#### VUI VPS set table syntax

|  |  |
| --- | --- |
| vui\_vps\_set( index ) { |  |
| **vui\_video\_parameters\_present\_flag** | u(1) |
| if( vui\_video\_parameters\_present\_flag ) |  |
| vui\_parameters() |  |
| vuiVpsSetIndex = index |  |
| } |  |

#### Optional SPS parameters syntax

|  |  |
| --- | --- |
| optional\_sps\_parameters( ) { |  |
| **pcm\_enabled\_flag** | u(1) |
| if( pcm\_enabled\_flag ) { |  |
| **pcm\_sample\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_sample\_bit\_depth\_chroma\_minus1** | u(4) |
| } |  |
| **log2\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| ~~for( i = 0; i <= sps\_max\_temporal\_layers\_minus1; i++ ) {~~ |  |
| **~~sps\_max\_dec\_pic\_buffering[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| **~~sps\_num\_reorder\_pics[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| **~~sps\_max\_latency\_increase[~~**~~i~~**~~]~~** | ~~ue(v)~~ |
| ~~}~~ |  |
| **restricted\_ref\_pic\_lists\_flag** | u(1) |
| if( restricted\_ref\_pic\_lists\_flag ) |  |
| **lists\_modification\_present\_flag** | u(1) |
| **log2\_min\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| if( pcm\_enabled\_flag ) { |  |
| **log2\_min\_pcm\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_pcm\_coding\_block\_size** | ue(v) |
| } |  |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **scaling\_list\_enable\_flag** | u(1) |
| if( scaling\_list\_enable\_flag ) { |  |
| **sps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( sps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_param( ) |  |
| } |  |
| **chroma\_pred\_from\_luma\_enabled\_flag** | u(1) |
| **transform\_skip\_enabled\_flag** | u(1) |
| **seq\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| **asymmetric\_motion\_partitions\_enabled\_flag** | u(1) |
| **nsrqt\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| if( adaptive\_loop\_filter\_enabled\_flag ) |  |
| **alf\_coef\_in\_slice\_flag** | u(1) |
| if( pcm\_enabled\_flag ) |  |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **~~sps\_temporal\_id\_nesting\_flag~~** | ~~u(1)~~ |
| [Ed. (BB): x y padding syntax missing here, present in HM software ] |  |
| if( log2\_min\_coding\_block\_size\_minus3 = = 0 ) |  |
| **inter\_4x4\_enabled\_flag** | u(1) |
| **long\_term\_ref\_pics\_present\_flag** | u(1) |
| **sps\_temporal\_mvp\_enable\_flag** | u(1) |
| **tiles\_fixed\_structure\_idc** | u(2) |
| } |  |

### Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **~~profile\_space~~** | ~~u(3)~~ |
| **~~profile\_idc~~** | ~~u(5)~~ |
| **~~constraint\_flags~~** | ~~u(16)~~ |
| **~~level\_idc~~** | ~~u(8)~~ |
| ~~for( i = 0; i < 32; i++ )~~ |  |
| **~~profile\_compatability\_flag[~~**~~i~~**~~]~~** | ~~u(1)~~ |
| **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~~** | ~~u(1)~~ |
| **~~sps\_max\_temporal\_layers\_minus1~~** | ~~u(3)~~ |
| **~~pic\_width\_in\_luma\_samples~~** | ~~ue(v)~~ |
| **~~pic\_height\_in\_luma\_samples~~** | ~~ue(v)~~ |
| **~~pic\_cropping\_flag~~** | ~~u(1)~~ |
| ~~if( pic\_cropping\_flag ) {~~ |  |
| **~~pic\_crop\_left\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_right\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_top\_offset~~** | ~~ue(v)~~ |
| **~~pic\_crop\_bottom\_offset~~** | ~~ue(v)~~ |
| ~~}~~ |  |
| **~~bit\_depth\_luma\_minus8~~** | ~~ue(v)~~ |
| **~~bit\_depth\_chroma\_minus8~~** | ~~ue(v)~~ |
| ~~[Ed. (BB): chroma bit depth present in HM software but not used further ]~~ |  |
| **num\_short\_term\_ref\_pic\_sets** | ue(v) |
| **use\_rps\_from\_vps\_flag** | u(1) |
| for( i = 0; i < num\_short\_term\_ref\_pic\_sets; i++){ |  |
| idx = use\_rps\_from\_vps\_flag ? num\_vps\_short\_term\_ref\_pic\_sets + i : i |  |
| short\_term\_ref\_pic\_set( idx ) |  |
| } |  |
| **sps\_parameters\_override\_flag** | u(1) |
| if( sps\_parameters\_override\_flag ) |  |
| optional\_sps\_parameters( ) |  |
| **~~vui\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( vui\_parameters\_present\_flag )~~ |  |
| ~~vui\_parameters( )~~ |  |
| **sps\_extension\_flag** | u(1) |
| if( sps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **sps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

### VUI parameters syntax

|  |  |
| --- | --- |
| vui\_vps\_parameters( ) { | Descriptor |
| **aspect\_ratio\_info\_present\_flag** | u(1) |
| if( aspect\_ratio\_info\_present\_flag ) { |  |
| **aspect\_ratio\_idc** | u(8) |
| if( aspect\_ratio\_idc = = Extended\_SAR ) { |  |
| **sar\_width** | u(16) |
| **sar\_height** | u(16) |
| } |  |
| } |  |
| **overscan\_info\_present\_flag** | u(1) |
| if( overscan\_info\_present\_flag ) |  |
| **overscan\_appropriate\_flag** | u(1) |
| **video\_signal\_type\_present\_flag** | u(1) |
| if( video\_signal\_type\_present\_flag ) { |  |
| **video\_format** | u(3) |
| **video\_full\_range\_flag** | u(1) |
| **colour\_description\_present\_flag** | u(1) |
| if( colour\_description\_present\_flag ) { |  |
| **colour\_primaries** | u(8) |
| **transfer\_characteristics** | u(8) |
| **matrix\_coefficients** | u(8) |
| } |  |
| } |  |
| **chroma\_loc\_info\_present\_flag** | u(1) |
| if( chroma\_loc\_info\_present\_flag ) { |  |
| **chroma\_sample\_loc\_type\_top\_field** | ue(v) |
| **chroma\_sample\_loc\_type\_bottom\_field** | ue(v) |
| } |  |
| **neutral\_chroma\_indication\_flag** | u(1) |
| **field\_seq\_flag** | u(1) |
| **timing\_info\_present\_flag** | u(1) |
| if( timing\_info\_present\_flag ) { |  |
| **num\_units\_in\_tick** | u(32) |
| **time\_scale** | u(32) |
| **fixed\_pic\_rate\_flag** | u(1) |
| } |  |
| **~~nal\_hrd\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( nal\_hrd\_parameters\_present\_flag )~~ |  |
| ~~hrd\_parameters( )~~ |  |
| **~~vcl\_hrd\_parameters\_present\_flag~~** | ~~u(1)~~ |
| ~~if( vcl\_hrd\_parameters\_present\_flag )~~ |  |
| ~~hrd\_parameters( )~~ |  |
| ~~if( nal\_hrd\_parameters\_present\_flag | | vcl\_hrd\_parameters\_present\_flag )~~ |  |
| **~~low\_delay\_hrd\_flag~~** | ~~u(1)~~ |
| **~~sub\_pic\_cpb\_params\_present\_flag~~** | ~~u(1)~~ |
| ~~if( sub\_pic\_cpb\_params\_present\_flag )~~ |  |
| **~~num\_units\_in\_sub\_tick~~** | ~~u(32)~~ |
| ~~}~~ |  |
| **bitstream\_restriction\_flag** | u(1) |
| if( bitstream\_restriction\_flag ) { |  |
| **~~tiles\_fixed\_structure\_flag~~** | ~~u(1)~~ |
| **motion\_vectors\_over\_pic\_boundaries\_flag** | u(1) |
| **max\_bytes\_per\_pic\_denom** | ue(v) |
| **max\_bits\_per\_mincu\_denom** | ue(v) |
| **log2\_max\_mv\_length\_horizontal** | ue(v) |
| **log2\_max\_mv\_length\_vertical** | ue(v) |
| } |  |
| } |  |

### HRD parameters syntax

|  |  |
| --- | --- |
| hrd\_parameters( i ) { | Descriptor |
| if( i = = 0 ) { |  |
| **cpb\_cnt\_minus1** | ue(v) |
| **bit\_rate\_scale** | u(4) |
| **cpb\_size\_scale** | u(4) |
| } |  |
| for( SchedSelIdx = 0; SchedSelIdx <= cpb\_cnt\_minus1; SchedSelIdx++ ) { |  |
| **bit\_rate\_value\_minus1[** i **][** SchedSelIdx **]** | ue(v) |
| **cpb\_size\_value\_minus1[** i **] [** SchedSelIdx **]** | ue(v) |
| if( i = = 0 ) |  |
| **cbr\_flag[** SchedSelIdx **]** | u(1) |
| } |  |
| if( i = = 0 ) { |  |
| **initial\_cpb\_removal\_delay\_length\_minus1** | u(5) |
| **cpb\_removal\_delay\_length\_minus1** | u(5) |
| **dpb\_output\_delay\_length\_minus1** | u(5) |
| **time\_offset\_length** | u(5) |
| **}** |  |
| } |  |

### Slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| ... |  |
| **short\_term\_ref\_pic\_set\_sps\_flag** | u(1) |
| if( !short\_term\_ref\_pic\_set\_sps\_flag ) |  |
| short\_term\_ref\_pic\_set( NumShortTermRefPicSets ) |  |
| else |  |
| **short\_term\_ref\_pic\_set\_idx** | u(v) |
| ... |  |
| } |  |

## Semantics

### Video parameter set RBSP semantics

**profile\_space**, **profile\_idc**, **profile\_compatability\_flag[** i **], constraint\_flags, level\_idc**, **bit\_depth\_luma\_minus8, bit\_depth\_chroma\_minus8, chroma\_format\_idc, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, pic\_cropping\_flag, pic\_crop\_left\_offset, pic\_crop\_right\_offset, pic\_crop\_top\_offset, pic\_crop\_bottom\_offset, temporal\_id\_nesting\_flag** and **separate\_colour\_plane\_flag** have the same semantics of those syntax elements with the same syntax element names in the sequence parameter set as specified in the current HEVC draft specification.

**level\_lower\_temporal\_layers\_present\_flag** equal to 1 specifies that level\_idc\_temporal\_subset[ i ] may be present. level\_lower\_temporal\_layers\_present\_flag equal to 0 specifies that level\_idc\_temporal\_subset[ i ] is not present.

**level\_idc\_temporal\_subset[** i **]** specifies the level to which the bitstream subset consisting of all NAL units with temporal\_id less than or equal to i conforms.

**vps\_temporal\_id\_nesting\_flag, vps\_temporal\_id\_nesting\_flag, vps\_max\_dec\_pic\_buffering[** i **], vps\_num\_reorder\_pics[** i **],** and **vps\_max\_latency\_increase[** i **]** have the same semantics of the following syntax elements respectively in the sequence parameter set of the current HEVC WD: sps\_temporal\_id\_nesting\_flag, sps\_temporal\_id\_nesting\_flag, sps\_max\_dec\_pic\_buffering[ i ], sps\_num\_reorder\_pics[ i ], sps\_max\_latency\_increase[ i ].

**next\_essential\_info\_byte\_offset** specifies the byte offset of the next set of profile and level information and other fixed-length coded information in the VPS NAL unit, starting from the beginning of the NAL unit. Decoders conforming to the base specification of HEVC shall ignore the value of next\_essential\_info\_byte\_offset.

NOTE 1 – In a future extension of this specification, e.g. a scalable coding extension or a 3DV extension, VPS information for non-base layer or view may be included in the VPS NAL unit, after the VPS information for the base layer or view. It is expected that the VPS information for non-base layer or view also starts from fixed-length coded profile, level and other information that are essential for session negotiation. The byte offset specified by next\_essential\_info\_byte\_offset would then help to locate and access that essential information in the VPS NAL unit without the need of entropy decoding, which may not be equipped with some network entities that may desire to access only all the information in the VPS essential for session negotiation.

**nal\_hrd\_parameters\_present\_flag[** i **]** and **vcl\_hrd\_parameters\_present\_flag[** i **]** have the similar semantic **as nal\_hrd\_parameters\_present\_flag, and vcl\_hrd\_parameters\_present\_flag that** are currently present in VUI parameters, but are applicable to the i-th temporal layer representation.

**low\_delay\_hrd\_flag, sub\_pic\_cpb\_params\_present\_flag,** and **num\_units\_in\_sub\_tick** have the semantics of those syntax elements with the same syntax element names in the VUI parameters as specified in the current HEVC draft specification. They are applicable to all temporal layer representations.

**vui\_parameters\_present\_flag** equal to 1 specifies that the vui\_vps( ) syntax structure is present in the VPS. This flag equal to 0 specifies that the vui\_vps( ) syntax element is not present.

**num\_vps\_short\_term\_ref\_pic\_sets** specifies the number of short-term reference picture sets that are specified in the video parameter set.

**bitrate\_info\_present\_flag[** i **]** equal to 1 specifies that the bit rate information for the i-th temporal layer is present in the video parameter set. bitrate\_info\_present\_flag[ i ] equal to 0 specifies that the bit rate information for the i-th temporal layer is not present in the video parameter set.

**frm\_rate\_info\_present\_flag[** i **]** equal to 1 specifies that frame rate information for the i-th temporal layer is present in the video parameter set. frm\_rate\_info\_present\_flag[ i ] equal to 0 specifies that frame rate information for the i-th temporal layer is not present in the video parameter set.

**avg\_bitrate[** i **]** indicates the average bit rate of the i-th temporal layer representation. The average bit rate for the i-th temporal layer representation in bits per second is given by BitRateBPS( avg\_bitrate[ i ] ) with the function BitRateBPS( ) being specified by

BitRateBPS( x ) = ( x & ( 214 − 1 ) ) \* 10( 2 + ( x >> 14 ) )

The average bit rate is derived according to the access unit removal time specified in Annex C of this Recommendation | International Standard. In the following, bTotal is the number of bits in all NAL units of the i-th temporal layer representation, t1 is the removal time (in seconds) of the first access unit to which the VPS applies, and t2 is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies.

With x specifying the value of avg\_bitrate[ i ], the following applies:

– If t1 is not equal to t2, the following condition shall be true:

( x & ( 214 − 1 ) )  = =  Round( bTotal ÷ ( ( t2 − t1 ) \* 10( 2 + ( x >> 14 ) ) ) )

– Otherwise (t1 is equal to t2), the following condition shall be true:

( x & ( 214 − 1 ) )  = =  0

**max\_bitrate\_layer[** i **]** indicates an upper bound for the bit rate of the i-th temporal layer representation in any one-second time window, of access unit removal time as specified in Annex C. The upper bound for the bit rate of the current scalable layer in bits per second is given by BitRateBPS( max\_bitrate\_layer[ i ] ) with the function BitRateBPS( ) being specified in Equation G-369. The bit rate values are derived according to the access unit removal time specified in Annex C of this Recommendation | International Standard. In the following, t1 is any point in time (in seconds), t2 is set equal to t1 + max\_bitrate\_calc\_window[ i ] ÷ 100, and bTotal is the number of bits in all NAL units of the current scalable layer that belong to access units with a removal time greater than or equal to t1 and less than t2. With x specifying the value of max\_bitrate\_layer[ i ], the following condition shall be obeyed for all values of t1:

( x & ( 214 − 1 ) )  >=  bTotal ÷ ( ( t2 − t1 ) \* 10( 2 + ( x >> 14 ) ) )

**constant\_frm\_rate\_idc[** i **]** indicates whether the frame rate of the i-th temporal layer representation is constant. In the following, a temporal segment tSeg is any set of two or more consecutive access units, in decoding order, of the current temporal layer representation, fTotal( tSeg ) is the number of pictures, in the temporal segment tSeg, t1( tSeg ) is the removal time (in seconds) of the first access unit (in decoding order) of the temporal segment tSeg, t2( tSeg ) is the removal time (in seconds) of the last access unit (in decoding order) of the temporal segment tSeg, and avgFR( tSeg ) is the average frame rate in the temporal segment tSeg, which is given by:

avgFR( tSeg)  = =  Round( fTotal( tSeg ) \* 256 ÷ ( t2( tSeg ) − t1( tSeg ) ) )

If the i-th temporal layer representation does only contain one access unit or the value of avgFR( tSeg ) is constant over all temporal segments of the i-th temporal layer representation, the frame rate is constant; otherwise, the frame rate is not constant. constant\_frm\_rate\_idc[ i ] equal to 0 indicates that the frame rate of the i-th temporal layer representation is not constant. constant\_frm\_rate\_idc[ i ] equal to 1 indicates that the frame rate of the i-th temporal layer representation is constant. constant\_frm\_rate\_idc[ i ] equal to  2 indicates that the frame rate of the i-th temporal layer representation may or may not be constant. The value of constant\_frm\_rate\_idc[ i ] shall be in the range of 0 to 2, inclusive.

**avg\_frm\_rate[** i **]** indicates the average frame rate, in units of frames per 256 seconds, of the i-th temporal layer representation. With fTotal being the number of pictures in the i-th temporal layer representation, t1 being the removal time (in seconds) of the first access unit to which the VPS applies, and t2 being the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies, the following applies:

– If t1 is not equal to t2, the following condition shall be true:

avg\_frm\_rate[ i ]  = =  Round( fTotal \* 256 ÷ ( t2 − t1 ) )

– Otherwise (t1 is equal to t2), the following condition shall be true:

avg\_frm\_rate[ i ]  = =  0

The semantics of the syntax elements and syntax structures the optional\_sps\_parameters( ) syntax structure have the same semantics as those syntax elements in the SPS with the same syntax element names as specified in the current HEVC draft.

#### Profile and level information table semantics

profileLevelInfoIdx indicates the index of the profile and level information table.

#### Representation format information table semantics

repFormatInfoIdx indicates the index of the representation format information table.

#### VUI VPS set table semantics

vuiVpsSetIndex indicates the index of the VUI VPS set table.

### Sequence parameter set RBSP semantics

**use\_rps\_from\_vps\_flag** equal to 1specifies that the short-term reference pictures sets included in the sequence parameter set are additive to the short-term reference pictures sets included in the referred video parameter set. use\_rps\_from\_vps\_flag equal to 1specifies that the short-term reference pictures sets included in the sequence parameter set override the short-term reference pictures sets included in the referred video parameter set.

The variable NumShortTermRefPicSets is derived as follows.

NumShortTermRefPicSets = num\_short\_term\_ref\_pic\_sets   
 if( use\_rps\_from\_vps\_flag )  
 NumShortTermRefPicSets += num\_vps\_short\_term\_ref\_pic\_sets

**sps\_parameters\_override\_flag** equal to 1specifies that the values of the syntax elements and syntax structures from pcm\_enabled\_flag through tiles\_fixed\_structure\_idc as specified in the sequence parameter set override the values of the same syntax elements and syntax structures as specified in the referred video parameter set. sps\_parameters\_override\_flag equal to 0 the values of the syntax elements and syntax structures from pcm\_enabled\_flag through tiles\_fixed\_structure\_idc as specified in the referred video parameter set are in use.

### VUI parameters semantics

Each syntax element in VUI parameters has the same semantics as the syntax element with the same name in the VUI parameters syntax as specified in the current HEVC draft specification.

### Slice header semantics

**short\_term\_ref\_pic\_set\_idx** specifies the index to the list of the short-term reference picture sets specified in the active sequence parameter set that shall be used for creation of the reference picture set of the current picture. The syntax element short\_term\_ref\_pic\_set\_idx shall be represented by Ceil( Log2( NumShortTermRefPicSets ) ) bits. The value of short\_term\_ref\_pic\_set\_idx shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, where num\_short\_term\_ref\_pic\_sets is the syntax element from the active sequence parameter set.

The variable StRpsIdx is derived as follows.

if( short\_term\_ref\_pic\_set\_sps\_flag )  
 StRpsIdx = short\_term\_ref\_pic\_set\_idx (7‑36)  
else  
 StRpsIdx = NumShortTermRefPicSets

**tiles\_fixed\_structure\_idc** equal to 0 indicates that each picture parameter set that is referred to by any picture in the coded video sequence has tiles\_or\_entropy\_coding\_sync\_idc equal to 0. tiles\_fixed\_structure\_idc equal to 1 indicates that each picture parameter set that is referred to by any picture in the coded video sequence has the same value of the syntax elements num\_tile\_columns\_minus1, num\_tile\_rows\_minus1, uniform\_spacing\_flag, column\_width[ i ], row\_height[ i ] and loop\_filter\_across\_tiles\_enabled\_flag, when present. tiles\_fixed\_structure\_idcg equal to 2 indicates that tiles syntax elements in different picture parameter sets that are referred to by pictures in the coded video sequence may or may not have the same value. The value of tiles\_fixed\_structure\_idc shall be in the range of 0 to 2, inclusive. When the tiles\_fixed\_structure\_flag syntax element is not present, it is inferred to be equal to 2.

NOTE 12 – The signalling of tiles\_fixed\_structure\_flag equal to 1 is a guarantee to a decoder that each picture in the coded video sequence has the same number of tiles distributed in the same way which might be useful for workload allocation in the case of multi-threaded decoding.

# Patent rights declaration(s)

**Qualcomm Incorporated 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).**