|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  15th Meeting: Geneva, CH, 23 Oct. – 1 Nov. 2013 | Document: JCTVC-O0058\_r2 |

|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extensions**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  6th Meeting: Geneva, CH, 25 Oct. – 1 Nov. 2013 | Document: JCT3V-F0035\_r2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **On profile, tier and level information** | | |
| *Status:* | Input Document to JCT-VC and JCT-3V | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Takeshi Tsukuba Tomoyuki Yamamoto Tomohiro Ikai  1-9-2 Nakase, Mihama-ku, Chiba-shi, Chiba 261-8520 JAPAN | Tel: Email: | +81-43-299-8526 [tsukuba.takeshi@sharp.co.jp](mailto:tsukuba.takeshi@sharp.co.jp) |
| *Source:* | SHARP Corporation | | |

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

# Abstract

This contribution proposes to 1) remove **profile\_ref\_minus1**, 2) add inference rules for sub-layer profile and level information and 3) byte alignment for profile\_tier\_level.

It is asserted that the proposal 1 to 3 have the following benefit.

1. Reduce bit amount of **profile\_ref\_minus1[ i ]** (and easy to byte-aligned).
2. Avoid confusing undefined value handling of profile\_tier\_level() especially when it is refered by **profile\_level\_tier\_idx**[ i ].
3. Easy to decode profile\_tier\_level( ) of layer set, which is important when negotiation. (e.g. the general profile or general level is not decodable but some layer set can be decodable).

# Introduction

The profile\_tier\_level( ) for layer sets is decoded as follows:

* Decode a list of profile\_tier\_level( ) with **vps\_profile\_present\_flag, profile\_ref\_minus1**.
* Select decoded profile\_tier\_level( ) by profile\_level\_tier\_idx[i].

However, there are several issues, called Issue A to Issue C in this contribution, on the current design.

1. **profile\_ref\_minus1** constantly uses 6 bits. However, it is not so much useful because, in many cases, the information is inferred from the previous one. Note that encoder can arrange elements of a list of profile\_tier\_level( ).
2. profile\_tier\_level( ), which is referred by **profile\_level\_tier\_idx**[ i ], can be not present. It is very confusing that there are unknown values that can be referred.
3. profile\_tier\_level( ) in vps\_extension is not byte-aligned.

# Proposal

Our proposals are summarised as follows;

1. To solve issue A, remove **profile\_ref\_minus1**.
2. To solve issue B, add the following inference rules:  
   1) When the level information is not present for the i-th sub-layer, it is inferred to be equal to the (i+1)-th sub-layer’s.  
   2) When vps\_profile\_present\_flag[i] of i-th profile\_tier\_level( ) is equal to 1 and the profile information is not present for the i-th sub-layer, it is inferred to be equal to (i+1)-th sub-layer’s.
3. To solve issue C, the following applies:   
   1) Add byte-alignment bits before profile\_tier\_level( ) or extend bit-length of profile\_ref\_minus1[i] from 6 bits to 8 bits without proposal a).

2) Add a syntax **vps\_ptr\_info\_offset** indicating byte-offset from starting of VPS NAL to the syntax(vps\_num\_layer\_sets\_minus1) related to profile\_tier\_level( ) or move syntaxes related to profile\_tier\_level( ) after scalability\_mask\_flag.

For more details, please see the Text change in section 4.

# Conclusion

This contribution proposes to 1) remove **profile\_ref\_minus1**, 2) add inference rules for sub-layer profile and level information and 3) byte alignment for profile\_tier\_level.

It is asserted that the proposal 1 to 3 have the following benefit.

1. Reduce bit amount of **profile\_ref\_minus1[ i ]** (and easy to byte-aligned).
2. Avoid confusing undefined value handling of profile\_tier\_level() especially when it is refered by **profile\_level\_tier\_idx**[ i ].
3. Easy to decode profile\_tier\_level( ) of layer set, which is important when negotiation. (e.g. the general profile or general level is not decodable but some layer set can be decodable).

It is recommended to adopt proposals to SHVC and MV-HEVC.

# Text changes

## Solution regarding issue A

* Remove **profile\_ref\_minus1**.

Changes are highlighted in yellow, and both of removals and typos are stroke through in red.

**F.7.3.2.1.1 Video parameter set extension syntax**

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| …snipped… |  |
| **vps\_number\_layer\_sets\_minus1** | u(10) |
| **vps\_num\_profile\_tier\_level\_minus1** | u(6) |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| ~~if( !vps\_profile\_present\_flag[ i ] )~~ |  |
| **~~profile\_ref\_minus1~~**~~[ i ]~~ | ~~u(6)~~ |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| …snipped… |  |
| } |  |

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

**vps\_profile\_present\_flag**[ i ] equal to 1 specifies that the profile and tier information ~~for layer set i~~ is present in the i-th profile\_tier\_level( ) syntax structure.vps\_profile\_present\_flag[ i ] equal to 0 specifies that profile and tier information is not present in the i-th profile\_tier\_level( ) syntax structure and is inferred to be equal to the profile and tier information of the ( i-1 )-th profile\_tier\_level( ) syntax structure. When the index i is equal to 1, vps\_profile\_present\_flag[i] shall be equal to 1.

**~~profile\_ref\_minus1~~**~~[ i ] specifies that the profile and tier information for the i-th profile\_tier\_level( ) syntax structure is inferred to be equal to the profile and tier information for the (profile\_ref\_minus1[ i ] + 1)-th layer set. The value of profile\_ref\_minus1[ i ] + 1 shall be less than i.~~

**F.7.4.4 Profile, tier and level semantics**

~~The profile\_tier\_level( ) syntax structure provides profile, tier and level information used for a layer set.~~ The profile\_tier\_level( ) syntax structure, specified by profile\_level\_tier\_idx[ i ] for the i–th layer set with i larger than 0, provides profile, tier and level information used for a layer set specified by the index i. ~~When the profile\_tier\_level( ) syntax structure is included in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is specified by the corresponding lsIdx variable in the vps\_extension( ) syntax structure. When the profile\_tier\_level( ) syntax structure is included in a VPS, but not in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0.~~ When the profile\_tier\_level( ) syntax structure is included in an SPS, the layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0.

For interpretation of the following semantics, CVS refers to the CVS subset associated with the layer set to which the profile\_tier\_level( ) syntax structure applies.

When vps\_profile\_present\_flag[ i ] is equal to 0, ~~When~~ the syntax elements general\_profile\_space, general\_tier\_flag, general\_profile\_idc, general\_profile\_compatibility\_flag[ j ], general\_progressive\_source\_flag, general\_interlaced\_source\_flag, general\_non\_packed\_constraint\_flag, general\_frame\_only\_constraint\_flag, general\_reserved\_zero\_44bits ~~are not present for the applicable layer set~~ in the i-th profile\_tier\_level( ) syntax structure~~, they~~ are inferred to be equal to the corresponding values of the ~~layer set~~ prior profile\_tier\_level( ) syntax structure specified by the index ~~(~~~~profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 )~~ (i-1), where the value of i shall be in the range of 1 to (vps\_num\_profile\_tier\_level\_minus1+1), inclusive.

When vps\_profile\_present\_flag[ i ] is equal to 0, the syntax elements sub\_layer\_profile\_space[ i ], sub\_layer\_tier\_flag[ i ], sub\_layer\_profile\_idc[ i ], sub\_layer\_profile\_compatibility\_flag[ i ][ j ], sub\_layer\_progressive\_source\_flag[ i ], sub\_layer\_interlaced\_source\_flag[ i ], sub\_layer\_non\_packed\_constraint\_flag[ i ], sub\_layer\_frame\_only\_constraint\_flag[ i ], sub\_layer\_reserved\_zero\_44bits[ i ] ~~are not present for the applicable layer set~~ in the i-th profile\_tier\_level( ) syntax structure, ~~and they are present in or inferred for the layer set specified by the index (~~~~profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ), they~~ are inferred to be equal to the corresponding values of the ~~layer set~~ prior profile\_tier\_level( ) syntax structure specified by the index ~~(~~~~profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 )~~ (i-1), where the value of i shall be in the range of 1 to (vps\_num\_profile\_tier\_level\_minus1+1), inclusive.

## Solution regarding issue B

In addition to the above changes of issue A, the following changes are applied;

* Add inference rule which indicates that, when the level information is not present for the i-th sub-layer, it is inferred to be equal to the (i+1)-th sub-layer’s or general level information.
* Add inference rule which indicates that, when vps\_profile\_present\_flag is equal to 0 and the profile information is not present for the i-th sub-layer, it is inferred to be equal to the (i+1)-th sub-layer’s or general profile information.

Changes are highlighted in yellow and removals are stroke through in red.

**F.7.4.4 Profile, tier and level semantics**

// inference rule for sub-layer level information

When sub\_layer\_level\_present\_flag[ i ] is equal to 0, sub\_layer\_level\_idc[ i ] is inferred by applying the follows:

for ( i = maxNumSubLayersMinus1; i >= 0; i-- ) {

if (!sub\_layer\_level\_present\_flag[ i ])

sub\_layer\_level\_idc[ i ] = (i == maxNumSubLayersMinus1) ? general\_level\_idc: sub\_layer\_level\_idc[ i+1 ]

}

}

// inference rule for sub-layer profile information

When vps\_profile\_present\_flag[ i ] is equal to 1 and sub\_layer\_profile\_present\_flag[ j ] is equal to 0 and the index j is less than vps\_max\_sub\_layers\_minus1, sub\_layer\_profile\_space[ j ], sub\_layer\_tier\_flag[ j ], sub\_layer\_profile\_idc[ j ], sub\_layer\_profile\_compatibility\_flag[ j ][ k ], sub\_layer\_progressive\_source\_flag[ j ], sub\_layer\_interlaced\_source\_flag[ j ], sub\_layer\_non\_packed\_constraint\_flag[ i ], sub\_layer\_frame\_only\_constraint\_flag[ j ], and sub\_layer\_reserved\_zero\_44bits[ j ] are infered to be equal to the corresponding value of the (j+1)-th sub-layer.

When vps\_profile\_present\_flag[ i ] is equal to 1 and sub\_layer\_profile\_present\_flag[ j ] is equal to 0 and the index j is equal to vps\_max\_sub\_layers\_minus1, sub\_layer\_profile\_space[ j ], sub\_layer\_tier\_flag[ j ], sub\_layer\_profile\_idc[ j ], sub\_layer\_profile\_compatibility\_flag[ j ][ k ], sub\_layer\_progressive\_source\_flag[ j ], sub\_layer\_interlaced\_source\_flag[ j ], sub\_layer\_non\_packed\_constraint\_flag[ i ], sub\_layer\_frame\_only\_constraint\_flag[ j ], and sub\_layer\_reserved\_zero\_44bits[ j ] are infered to be equal to the corresponding value of general\_profile\_space, general\_tier\_flag, general\_profile\_idc, general\_profile\_compatibility\_flag[ j ], general\_progressive\_source\_flag, general\_interlaced\_source\_flag, general\_non\_packed\_constraint\_flag, general\_frame\_only\_constraint\_flag, general\_reserved\_zero\_44bits, and general\_level\_idc, respectively.

## Solution regarding issue C

The following changes are applied:

* Replace the syntax elements (e.g. splitting\_flag to vps\_vui\_present\_flag).
* Option A) add a syntax **vps\_ptr\_info\_offset** indicating byte-offset from starting of VPS NAL to the syntax(vps\_num\_layer\_sets\_minus1) related to profile\_tier\_level( )
* Option B) move syntaxes related to profile\_tier\_level( ) after scalability\_mask\_flag.

The following changes are also applied;

1. Option1) Add byte-alignment bits between those syntaxes, with signalling vps\_profile\_present\_flag[i] and profile\_tier\_level( ) in vps\_extension( ) separately
2. Option2) Extend bit-length of profile\_ref\_minus1[i] from 6 bits to 8 bits

Changes are highlighted in cyan and yellow and removals are stroke through in red.

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

**vps\_ reserved\_zero\_6bits** shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for vps\_ reserved\_zero\_6bits are reserved for future use by ITU-T | ISO/IEC. Although the value of vps\_ reserved\_zero\_6bits is required to be equal to 0 in this version of this Specification, decoders shall allow other values of vps\_ reserved\_zero\_6bits to appear in the syntax.

**vps\_ptr\_info\_offset** specifies the byte offset, starting from the beginning of the VPS NAL unit, of the set of fixed-length coded information starting from vps\_number\_layer\_sets\_minus1, in the VPS NAL unit. Emulation prevention bytes that appear in the VPS NAL unit are counted for purposes of byte offset identification.

|  |  |
| --- | --- |
| vps\_extension( ) { // Option A | Descriptor |
| **avc\_base\_layer\_flag** | u(1) |
| **splitting\_flag** | u(1) |
| **vps\_nuh\_layer\_id\_present\_flag** | u(1) |
| **max\_tid\_ref\_present\_flag** | u(1) |
| **all\_ref\_layers\_active\_flag** | u(1) |
| **rep\_format\_idx\_present\_flag** | u(1) |
| **max\_one\_active\_ref\_layer\_flag** | u(1) |
| **cross\_layer\_irap\_aligned\_flag** | u(1)  **Byte-aligned** |
| **single\_layer\_for\_non\_irap\_flag** | u(1) |
| **vps\_vui\_present\_flag** | u(1) |
| **vps\_reserved\_6bits** | u(6) |
| **vps\_vui\_offset** | u(16) |
| **vps\_ptr\_info\_offset** | u(16) |
| **~~splitting\_flag~~** | ~~u(1)~~ |
| for( i = 0, NumScalabilityTypes = 0; i < 16; i++ ) { |  |
| **scalability\_mask\_flag**[ i ] | u(1) |
| NumScalabilityTypes += scalability\_mask\_flag[ i ] |  |
| } |  |
| for( j = 0; j < ( NumScalabilityTypes − splitting\_flag ); j++ ) |  |
| **dimension\_id\_len\_minus1**[ j ] | u(3) |
| **~~vps\_nuh\_layer\_id\_present\_flag~~** | ~~u(1)~~ |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| if( vps\_nuh\_layer\_id\_present\_flag ) |  |
| **layer\_id\_in\_nuh**[ i ] | u(6) |
| if( !splitting\_flag ) |  |
| for( j = 0; j < NumScalabilityTypes; j++ ) |  |
| **dimension\_id**[ i ][ j ] | u(v) |
| } |  |
| if( NumViews > 1 ) |  |
| **view\_id\_len\_minus1** | u(4) |
| for( i = 0; i < NumViews; i++ ) |  |
| **view\_id\_val**[ i ] | u(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| **direct\_dependency\_flag**[ i ][ j ] | u(1) |
| **~~max\_tid\_ref\_present\_flag~~** | ~~u(1)~~ |
| if( max\_tid\_ref\_present\_flag ) |  |
| for( i = 0; i < vps\_max\_layers\_minus1; i++ ) |  |
| **max\_tid\_il\_ref\_pics\_plus1**[ i ] | u(3) |
| **~~all\_ref\_layers\_active\_flag~~** | ~~u(1)~~ |
| … |  |
| // option 1) |  |
| while(!byte\_aligned() ) |  |
| **vps\_aliginment\_bit\_equal\_to\_one** | u(1) |
| **vps\_number\_layer\_sets\_minus1** | u(10) |
| **vps\_num\_profile\_tier\_level\_minus1** | u(6) |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| } |  |
| for( i = ((8-(vps\_num\_profile\_tier\_level\_minus1 + 1)%8)%8); i >0 ;i-- ) { |  |
| **vps\_reseved\_zero\_bit**[ i ] | u(1) |
| } | **Byte-aligned** |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **~~vps\_profile\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~if( !vps\_profile\_present\_flag[ i ] )~~ |  |
| **~~profile\_ref\_minus1~~**~~[ i ]~~ | ~~u(6)~~ |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| // option 2) |  |
| while(!byte\_aligned() ) |  |
| **vps\_aliginment\_bit\_equal\_to\_one** | u(1) |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| if( !vps\_profile\_present\_flag[ i ] ) |  |
| **profile\_ref\_minus1**[ i ] | ~~u(6)~~ u(8) |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| numOutputLayerSets = vps\_number\_layer\_sets\_minus1 + 1 |  |
| **more\_output\_layer\_sets\_than\_default\_flag** | u(1) |
| if( more\_output\_layer\_sets\_than\_default\_flag ) { |  |
| **num\_add\_output\_layer\_sets\_minus1** | u(10) |
| numOutputLayerSets += num\_add\_output\_layer\_sets\_minus1 + 1 |  |
| } |  |
| if( numOutputLayerSets > 1 ) |  |
| **default\_one\_target\_output\_layer\_flag** | u(1) |
| for( i = 1; i < numOutputLayerSets; i++ ) { |  |
| if( i > vps\_number\_layer\_sets\_minus1 ) { |  |
| **output\_layer\_set\_idx\_minus1**[ i ] | u(v) |
| lsIdx = output\_layer\_set\_idx\_minus1[ i ] + 1 |  |
| for( j = 0 ; j < NumLayersInIdList[ lsIdx ] − 1; j++) |  |
| **output\_layer\_flag**[ i ][ j ] | u(1) |
| } |  |
| **profile\_level\_tier\_idx**[ i ] | u(v) |
| } |  |
| **~~rep\_format\_idx\_present\_flag~~** | ~~u(1)~~ |
| if( rep\_format\_idx\_present\_flag ) |  |
| **vps\_num\_rep\_formats\_minus1** | u(4) |
| for( i = 0; i <= vps\_num\_rep\_formats\_minus1; i++ ) |  |
| rep\_format( ) |  |
| if( rep\_format\_idx\_present\_flag ) |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| if( vps\_num\_rep\_formats\_minus1 > 0 ) |  |
| **vps\_rep\_format\_idx**[ i ] | u(4) |
| **~~max\_one\_active\_ref\_layer\_flag~~** | ~~u(1)~~ |
| **~~cross\_layer\_irap\_aligned\_flag~~** | ~~u(1)~~ |
| **direct\_dep\_type\_len\_minus2** | ue(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| if( direct\_dependency\_flag[ i ][ j ] ) |  |
| **direct\_dependency\_type**[ i ][ j ] | u(v) |
| **~~single\_layer\_for\_non\_irap\_flag~~** | ~~u(1)~~ |
| **~~vps\_vui\_present\_flag~~** | ~~u(1)~~ |
| if( vps\_vui\_present\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_vui\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_vui( ) |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| vps\_extension( ) { // Option B | Descriptor |
| **avc\_base\_layer\_flag** | u(1) |
| **splitting\_flag** | u(1) |
| **vps\_nuh\_layer\_id\_present\_flag** | u(1) |
| **max\_tid\_ref\_present\_flag** | u(1) |
| **all\_ref\_layers\_active\_flag** | u(1) |
| **rep\_format\_idx\_present\_flag** | u(1) |
| **max\_one\_active\_ref\_layer\_flag** | u(1) |
| **cross\_layer\_irap\_aligned\_flag** | u(1) |
| **single\_layer\_for\_non\_irap\_flag** | u(1) |
| **vps\_vui\_present\_flag** | u(1) |
| **vps\_reserved\_6bits** | u(6) |
| **vps\_vui\_offset** | u(16) |
| **~~splitting\_flag~~** | ~~u(1)~~ |
| **vps\_number\_layer\_sets\_minus1** | u(10)  **Byte-aligned** |
| **vps\_num\_profile\_tier\_level\_minus1** | u(6) |
| ...// scalalbility\_mask\_flag |  |
| // option 1) |  |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| } |  |
| for( i = ((8-(vps\_num\_profile\_tier\_level\_minus1 + 1)%8)%8); i >0 ;i-- ) { |  |
| **vps\_reseved\_zero\_bit**[ i ] | u(1) |
| } |  |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **~~vps\_profile\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| ~~if( !vps\_profile\_present\_flag[ i ] )~~ |  |
| **~~profile\_ref\_minus1~~**~~[ i ]~~ | ~~u(6)~~ |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| // option 2) |  |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **~~vps\_profile\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| if( !vps\_profile\_present\_flag[ i ] ) |  |
| **profile\_ref\_minus1**[ i ] | ~~u(6)~~ u(8) |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| for( j = 0; j < ( NumScalabilityTypes − splitting\_flag ); j++ ) |  |
| **dimension\_id\_len\_minus1**[ j ] | u(3) |
| **~~vps\_nuh\_layer\_id\_present\_flag~~** | ~~u(1)~~ |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| if( vps\_nuh\_layer\_id\_present\_flag ) |  |
| **layer\_id\_in\_nuh**[ i ] | u(6) |
| if( !splitting\_flag ) |  |
| for( j = 0; j < NumScalabilityTypes; j++ ) |  |
| **dimension\_id**[ i ][ j ] | u(v) |
| } |  |
| if( NumViews > 1 ) |  |
| **view\_id\_len\_minus1** | u(4) |
| for( i = 0; i < NumViews; i++ ) |  |
| **view\_id\_val**[ i ] | u(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| **direct\_dependency\_flag**[ i ][ j ] | u(1) |
| **~~max\_tid\_ref\_present\_flag~~** | ~~u(1)~~ |
| if( max\_tid\_ref\_present\_flag ) |  |
| for( i = 0; i < vps\_max\_layers\_minus1; i++ ) |  |
| **max\_tid\_il\_ref\_pics\_plus1**[ i ] | u(3) |
| **~~all\_ref\_layers\_active\_flag~~** | ~~u(1)~~ |
| **… // move syntaxes related to profile\_tier\_level() to top part in vps\_extension()** |  |
| numOutputLayerSets = vps\_number\_layer\_sets\_minus1 + 1 |  |
| **more\_output\_layer\_sets\_than\_default\_flag** | u(1) |
| if( more\_output\_layer\_sets\_than\_default\_flag ) { |  |
| **num\_add\_output\_layer\_sets\_minus1** | u(10) |
| numOutputLayerSets += num\_add\_output\_layer\_sets\_minus1 + 1 |  |
| } |  |
| if( numOutputLayerSets > 1 ) |  |
| **default\_one\_target\_output\_layer\_flag** | u(1) |
| for( i = 1; i < numOutputLayerSets; i++ ) { |  |
| if( i > vps\_number\_layer\_sets\_minus1 ) { |  |
| **output\_layer\_set\_idx\_minus1**[ i ] | u(v) |
| lsIdx = output\_layer\_set\_idx\_minus1[ i ] + 1 |  |
| for( j = 0 ; j < NumLayersInIdList[ lsIdx ] − 1; j++) |  |
| **output\_layer\_flag**[ i ][ j ] | u(1) |
| } |  |
| **profile\_level\_tier\_idx**[ i ] | u(v) |
| } |  |
| **~~rep\_format\_idx\_present\_flag~~** | ~~u(1)~~ |
| if( rep\_format\_idx\_present\_flag ) |  |
| **vps\_num\_rep\_formats\_minus1** | u(4) |
| for( i = 0; i <= vps\_num\_rep\_formats\_minus1; i++ ) |  |
| rep\_format( ) |  |
| if( rep\_format\_idx\_present\_flag ) |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| if( vps\_num\_rep\_formats\_minus1 > 0 ) |  |
| **vps\_rep\_format\_idx**[ i ] | u(4) |
| **~~max\_one\_active\_ref\_layer\_flag~~** | ~~u(1)~~ |
| **~~cross\_layer\_irap\_aligned\_flag~~** | ~~u(1)~~ |
| **direct\_dep\_type\_len\_minus2** | ue(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| if( direct\_dependency\_flag[ i ][ j ] ) |  |
| **direct\_dependency\_type**[ i ][ j ] | u(v) |
| **~~single\_layer\_for\_non\_irap\_flag~~** | ~~u(1)~~ |
| **~~vps\_vui\_present\_flag~~** | ~~u(1)~~ |
| if( vps\_vui\_present\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_vui\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_vui( ) |  |
| } |  |
| } |  |

**// option 2)**

**profile\_ref\_minus1**[ i ] specifies that the profile and tier information for the i-th profile\_tier\_level( ) syntax structure is inferred to be equal to the profile and tier information for the (profile \_ref\_minus1[ i ] + 1)-th profile\_tier\_level( ) syntax structure ~~layer set~~. The value of profile\_ref\_minus1[ i ] + 1 shall be less than i.

# Reference

1. Y. Wang, et.al, “AHG9: On VPS and SPS in HEVC 3DV and scalable extensions”, JCTVC-M0268, Incheon, KR, 18–26 Apr. 2013.
2. J. Chen, et.al, “SHVC Draft 3”, JCTVC-N1008, Vienna, AT, 25 July–2 Aug. 2013.
3. G. Tech, et.al, “MV-HEVC Draft Text 5”, JCT3V-E1004, Vienna, AT, 27 July–2 Aug. 2013.

# Patent rights declaration(s)

**SHARP Corporation 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).**