|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  11th Meeting: Shanghai, CN, 10– 19 Oct. 2012 | Document: JCTVC-K0254r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG9: Parsing profile and level information** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Takeshi Tsukuba\*  Tomoyuki Yamamoto\*  Sachin Deshpande\*\*  \*1-9-2 Nakase, Mihama-ku, Chiba-shi, Chiba 261-8520  JAPAN  \*\*5750 Pacific Rim Blvd,  Camas, WA 98607,  USA. | Tel: Email: | +81-43-299-8526  [tsukuba.takeshi@sharp.co.jp](mailto:tsukuba.takeshi@sharp.co.jp) [yamamoto.tomoyuki@sharp.co.jp](mailto:yamamoto.tomoyuki@sharp.co.jp)  [sdeshpande@sharplabs.com](mailto:sdeshpande@sharplabs.com) |
| *Source:* | SHARP Corporation, Sharp Labs of America, Inc. | | |

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

# Abstract

This contribution proposes a change to make the syntax of profile\_tier\_level() more parsing friendly. The proposal separates the part of sub-layer profile/level present flags and the part of the sub-layer profile/level information, which keeps profile\_tier\_level() syntax byte-aligned.

This contribution also proposes a change to move resereved\_zero\_12bits syntax element before profile\_tier\_level() section in VPS and extend its bit-length to 16 bits for maintaining byte-alignment. The change has the advantage that when that syntax is later used to signal the byte offset of the next set of fixed-length coded information in the video parameter set NAL unit, it is easier to parse the value signaled in those 16 bits without the need to parse the entire profile\_tier\_level() section.

# Problem Statement

In HEVC DIS [1], there remains a drawback of parsing profile\_tier\_level() information:

1. The profile and level information has been designed to fulfill byte-aligned thoughtfully. However at the sub-layer related syntax that design fails to function, which is highlighted in orange in Table.1. The design inconsistency may cause unnecessary hardware and software implementation complexity. Byte aligned information is readable by human and is beneficial for development and test.[Solution 1, 2]
2. Sub-layer profile and level present flags and those information are mixed / interleaved in the syntax structure and whole is encoded before vps\_reserved\_zero\_12bits. When a decoder extracts the specified sub-layer profile and level information or the syntax following profile\_tier\_level(), it needs to parse all of the syntax of the preceding sub-layers since it cannot easily estimate the amount of bits related to sub-layer profile and level information.[Solution 1, 3]
3. When ProfilePresentFlag is equal to 0, signalling of sub-layer profile present flags is redundant. [Solution 2]

Therefore, the current profile\_tier\_level() syntax structure as well as its position in VPS should be fixed to be designed decoder-friendly and development-friendly.

Table : Profile, tier and level syntax in HM8

|  |  |
| --- | --- |
| profile\_tier\_level( ProfilePresentFlag, MaxNumSubLayersMinus1 ) { | **Descriptor** |
| if( ProfilePresentFlag ) { |  |
| **general\_profile\_space** | u(2) |
| **general\_tier\_flag** | u(1) |
| **general\_profile\_idc** | u(5) |
| for( i = 0; i < 32; i++ ) |  |
| **general\_profile\_compatibility\_flag**[ i ] | u(1) |
| **general\_reserved\_zero\_16bits** | u(16) |
| } |  |
| **general\_level\_idc** | u(8) |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **sub\_layer\_profile\_present\_flag**[ i ] | u(1) |
| **sub\_layer\_level\_present\_flag**[ i ] | u(1) |
| if( ProfilePresentFlag && sub\_layer\_profile\_present\_flag[ i ] ){ |  |
| **sub\_layer\_profile\_space**[ i ] | u(2) |
| **sub\_layer\_tier\_flag**[ i ] | u(1) |
| **sub\_layer\_profile\_idc**[ i ] | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **sub\_layer\_profile\_compatibility\_flag**[ i ][ j ] | u(1) |
| **sub\_layer\_reserved\_zero\_16bits**[ i ] | u(16) |
| } |  |
| if( sub\_layer\_level\_present\_flag[ i ] ) |  |
| **sub\_layer\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |

# Proposal

To extract or skip parsing sub-layer profile and level information more easily, we propose the following two solutions. In addition, to parse next\_essential\_info\_byte\_offset more easily, we propose the following third solution.

## Solution 1:

The first solution is to split the part of sub-layer profile/level present flags and the part of sub-layer profile/level information so that present flags are collectively signaled before the associated information. In addition, to make all the syntax of profile\_tier\_level() byte-aligned, bit\_equal\_to\_zero are inserted after the first part. The second part is naturally byte-aligned due to its design.

Table 2 shows the proposed profile\_tier\_level() syntax where the changes are highlighted in yellow. There is no semantics change.

Table : Proposed Profile, tier, level syntax on Solution 1

|  |  |
| --- | --- |
| profile\_tier\_level( ProfilePresentFlag, MaxNumSubLayersMinus1 ) { | **Descriptor** |
| if( ProfilePresentFlag ) { |  |
| **general\_profile\_space** | u(2) |
| **general\_tier\_flag** | u(1) |
| **general\_profile\_idc** | u(5) |
| for( i = 0; i < 32; i++ ) |  |
| **general\_profile\_compatibility\_flag**[ i ] | u(1) |
| **general\_reserved\_zero\_16bits** | u(16) |
| } |  |
| **general\_level\_idc** | u(8) |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **sub\_layer\_profile\_present\_flag[ i ]** | u(1) |
| **sub\_layer\_level\_present\_flag[ i ]** | u(1) |
| } |  |
| While( !byte\_aligned( ) ) |  |
| **bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **~~sub\_layer\_profile\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| **~~sub\_layer\_level\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| if( ProfilePresentFlag && sub\_layer\_profile\_present\_flag[ i ] ) { |  |
| **sub\_layer\_profile\_space**[ i ] | u(2) |
| **sub\_layer\_tier\_flag**[ i ] | u(1) |
| **sub\_layer\_profile\_idc**[ i ] | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **sub\_layer\_profile\_compatibility\_flag**[ i ][ j ] | u(1) |
| **sub\_layer\_reserved\_zero\_16bits**[ i ] | u(16) |
| } |  |
| if( sub\_layer\_level\_present\_flag[ i ] ) |  |
| **sub\_layer\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |

## Solution 2

The second solution is further to separate sub-layer profile present flags and level present flags besides that separation of those present flags and its information. In this option, sub-layer profile present flag is only signaled when ProfilePresentFlag is equal to 1. Whole part is byte aligned with inserting padding bits (bit\_equal\_to\_zero).

Table 3 shows proposed profile\_tier\_level() syntax where the changes are highlighted in yellow. Semantics of sub\_layer\_profile\_present\_flag[ i ] is changed as follows.

**sub\_layer\_profile\_present\_flag**[ i ] equal to 1, when ProfilePresentFlag is equal to 1, specifies that profile information is present in the profile\_tier\_level( ) syntax structure for the representation of the sub-layer with TemporalId equal to i. sub\_layer\_profile\_present\_flag[ i ] equal to 0 specifies that profile information is not present in the profile\_tier\_level( ) syntax structure for the representations of the sub-layer with TemporalId equal to i. When ProfilePresentFlag is equal to 0, sub\_layer\_profile\_present\_flag[ i ] is inferred to be equal to 0.

Table : Proposed Profile, tier, level syntax on Solution 2

|  |  |
| --- | --- |
| profile\_tier\_level( ProfilePresentFlag, MaxNumSubLayersMinus1 ) { | **Descriptor** |
| if( ProfilePresentFlag ) { |  |
| **general\_profile\_space** | u(2) |
| **general\_tier\_flag** | u(1) |
| **general\_profile\_idc** | u(5) |
| for( i = 0; i < 32; i++ ) |  |
| **general\_profile\_compatibility\_flag**[ i ] | u(1) |
| **general\_reserved\_zero\_16bits** | u(16) |
| } |  |
| **general\_level\_idc** | u(8) |
| if( ProfilePresentFlag ){ |  |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **sub\_layer\_profile\_present\_flag[ i ]** | u(1) |
| } |  |
| While( !byte\_aligned( ) ) |  |
| **bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| } |  |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **sub\_layer\_level\_present\_flag[ i ]** | u(1) |
| } |  |
| While( !byte\_aligned( ) ) |  |
| **bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| for( i = 0; i < MaxNumSubLayersMinus1; i++ ) { |  |
| **~~sub\_layer\_profile\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| **~~sub\_layer\_level\_present\_flag~~**~~[ i ]~~ | ~~u(1)~~ |
| if( ProfilePresentFlag && sub\_layer\_profile\_present\_flag[ i ] ) { |  |
| **sub\_layer\_profile\_space**[ i ] | u(2) |
| **sub\_layer\_tier\_flag**[ i ] | u(1) |
| **sub\_layer\_profile\_idc**[ i ] | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **sub\_layer\_profile\_compatibility\_flag**[ i ][ j ] | u(1) |
| **sub\_layer\_reserved\_zero\_16bits**[ i ] | u(16) |
| } |  |
| if( sub\_layer\_level\_present\_flag[ i ] ) |  |
| **sub\_layer\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |
|  |  |

## Solution 3

The third solution is to move resereved\_zero\_12bits syntax element before profile\_tier\_level() section in VPS and extend its bit-length to 16 bits for maintaining byte-alignment. Table 4 shows the proposed syntax arrangement of “next\_essential\_byte\_offset” on vps where the changes are highlighted in yellow.

The change has the advantage that when that syntax is later specified to signal the byte offset of the next set of fixed-length coded information in the video parameter set NAL unit, it is easier to parse the value signaled in those 16 bits without the need to parse the entire profile\_tier\_level() section. This would be especially useful for network entities in quickly accessing this byte offset to access only the information in the video parameter set that is essential for session negotiation and/or capability exchange.

Note that it is available to apply the combination of solution 1 and 3 or that of solution 2 and 3 at the same time.

Table : Proposed video parameter set RBSP syntax on Solution 3

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **video\_parameter\_set\_id** | u(4) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| **vps\_reserved\_zero\_2bits** | u(2) |
| **vps\_reserved\_zero\_6bits** | u(6) |
| **vps\_max\_sub\_layers\_minus1** | u(3) |
| **vps\_reserved\_zero\_16bits**  **// next\_essential\_byte\_offset** | u(16) |
| profile\_tier\_level( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| **~~vps\_reserved\_zero\_12bits~~** | ~~u(12)~~ |
| for( i = 0; 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\_num\_hrd\_parameters** | ue(v) |
| for( i = 0; i < vps\_num\_hrd\_parameters; i++ ) { |  |
| if( i > 0 ) |  |
| operation\_point( i ) |  |
| hrd\_parameters( i = = 0, 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( ) |  |
| } |  |

# Conclusion

This contribution presents the improved profile\_tier\_level() syntax structure and the syntax order on VPS, which is more decoder-friendly and development. We recommend one of the proposed solutions to be adopted as a part of the HEVC. Especially, we recommend the combination of solution 1 and 3 or that of solution 2 and 3 to be adopted as part of the HEVC.

# References

[1] B. Bross, W-J Han, J-R Ohm, G. J. Sullivan, and T. Wiegand, “High efficiency video coding (HEVC) text specification draft 8,” JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 10th Meeting: Stockholm, SE, July, 2012

[2] Y-K Wang, “Solutions considered for NAL unit header and video parameter set for HEVC extensions,” JCTVC-J1007, JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 10th Meeting: Stockholm, SE, July, 2012

# 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).**