|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  18th Meeting: Sapporo, JP, 30 June – 9 July 2014 | Document: JCTVC-R0164-r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Bug fix in SHVC draft specification** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Philippe Bordes, Pierre Andrivon, Fabien Racapé  975 avenue des champs blancs  CS 17616, 35576, Cesson-Sévigné Cedex, France  Yuwen He, Yan Ye  9710 Scranton Rd, #250  San Diego, CA 92121 | Tel: Email: | +33 299273242  [philippe.bordes@technicolor.com](mailto:philippe.bordes@technicolor.com),  [pierre.andrivon@technicolor.com](mailto:pierre.andrivon@technicolor.com),  [fabien.racape@technicolor.com](mailto:edouard.françois@technicolor.com)  +1-858-210-4819  [yuwen.he@interdigital.com](mailto:yuwen.he@interdigital.com) |
| *Source:* | Technicolor, InterDigital Communications Inc | | |

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

# Abstract

A bug-fix of the syntax of colour\_mapping\_octants() is proposed in this contribution, in order to arrange the explicitly-encoded vertices more uniformly in the colour mapping table. The performance and behaviour of the SHM encoder is not impacted by the bug fix.

# Introduction

In the current “Preliminary version of High efficiency video coding (HEVC) scalable extension Draft 6” (JCTVC\_Q1008\_v3), the coding of the color mapping table (**colour\_mapping\_enabled\_flag**=1 in PPS) uses a tree of depth up to 3 (**cm\_octant\_depth**). The LUT can be further partitioned in the Y direction with the *YPartNum* variable (**cm\_y\_part\_num\_log2**) to support asymmetric LUTs, as depicted in Figure 1.

The **split\_octant\_flag** syntax element allows to locally control the color space sampling resolution in each octant. This has the benefits of both reducing signaling overhead and reducing the number of syntax elements to parse.

When **split\_octant\_flag**=0 anddepth< **cm\_octant\_depth**, some vertices are not explicitly encoded but derived from neighboring (explicitly encoded) vertices as specified in F.7.4.6.2 of the Draft 6. In this case, the (**res\_y, res\_u, res\_v**) syntax elements are inferred to be equal to 0 and the colour mapping table reconstruction process (H.8.1.4.3.1) allows to recursively derive these vertices.



Figure 1: Example of color space partitioning with the CM\_octnat\_depth=1 and cm\_y\_part\_num\_log2=2 .

However, with the current syntax of *colour\_mapping\_octants()* (F.7.3.5.2) the explicitly-encoded-vertices are always arranged in the LUT in consecutive way when **cm\_y\_part\_num\_log2** > 0 in the Y dimension since the index location (yIdx) is incremented by 1 regardless of the value of *depth* variable (see red text in the spec below).

|  |  |
| --- | --- |
| colour\_mapping\_octants( depth, yIdx, uIdx, vIdx, length ) { | **Descriptor** |
| if ( depth < cm\_octant\_depth ) |  |
| **split\_octant\_flag** | u(1) |
| if ( split\_octant\_flag ) |  |
| for( k = 0; k < 2; k++ ) |  |
| for( m = 0; m < 2 ; m++ ) |  |
| for( n = 0; n < 2; n++ ) |  |
| colour\_mapping\_octants( depth + 1, yIdx + YPartNum \* k \* length / 2, uIdx + m \* length / 2, vIdx + n \* length / 2, length / 2) |  |
| else |  |
| for( i = 0; i < YPartNum; i++ ) |  |
| for( vertex = 0; vertex < 4; vertex++ ) { |  |
| **coded\_vertex\_flag**[ yIdx + **i** ][ uIdx ][ vIdx ][ vertex ] | u(1) |
| if( coded\_vertex\_flag[ yIdx + **i** ][ uIdx ][ vIdx ][ vertex ] ) { |  |
| **res\_y**[ yIdx + **i** ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **res\_u**[ yIdx + **i** ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **res\_v**[ yIdx + **i** ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **}** |  |
| } |  |
| } |  |

Figure 2 depicts an example of LUT (Y and U dimensions) obtained with **cm\_octant\_depth**=2, **YPartNum**=4, where the *octant(y=0,u=0,v=0, depth=0)* is split in 8 octants, in which the first octant *octant (y=0,u=0,v=0, depth=1)* is split again in 8 octants but the second octant *octant (y=1,u=0,v=0, depth=1)* (depicted in green) is not split. The explicitly-encoded vertices are represented with red circles and the non-explicitly-encoded vertices are represented with “?”.



Figure 2: Example of LUT coding tree with explicitly-decoded vertices in red and non-explicitly-decoded vertices marked with “?”.

In Figure 2 we also depicted how the non-explicitly-encoded vertices are derived from neighboring explicitly-encoded vertices (blue arrows). We can notice two issues in the non-split octant (green):

1. The explicitly-encoded vertices are not arranged uniformly in the LUT.
2. The relative distance of vertex derivation is up to R/4 in the first row and up to R/2 in the next row (with R = range of the component, for example R= 256 for 8-bit signal).

# Proposed bug-fix

The proposed modified syntax is depicted in Table 1 (modified part in red), where the variable “shift” is equal to “**cm\_octant\_depth** – depth”.

Table 1: proposed colour\_mapping\_octants() modified syntax.

|  |  |
| --- | --- |
| colour\_mapping\_octants( depth, yIdx, uIdx, vIdx, length ) { | **Descriptor** |
| if ( depth < cm\_octant\_depth ) |  |
| **split\_octant\_flag** | u(1) |
| if ( split\_octant\_flag ) |  |
| for( k = 0; k < 2; k++ ) |  |
| for( m = 0; m < 2 ; m++ ) |  |
| for( n = 0; n < 2; n++ ) |  |
| colour\_mapping\_octants( depth + 1, yIdx + YPartNum \* k \* length / 2, uIdx + m \* length / 2, vIdx + n \* length / 2, length / 2) |  |
| else { |  |
| shift = cm\_octant\_depth – depth |  |
| for( i = 0; i < YPartNum; i++ ) |  |
| for( vertex = 0; vertex < 4; vertex++ ) { |  |
| **coded\_vertex\_flag**[ yIdx + (i<<shift) ][ uIdx ][ vIdx ][ vertex ] | u(1) |
| if( coded\_vertex\_flag[ yIdx + (i<<shift) ][ uIdx ][ vIdx ][ vertex ] ) { |  |
| **res\_y**[ yIdx + (i<<shift) ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **res\_u**[ yIdx + (i<<shift) ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **res\_v**[ yIdx + (i<<shift) ][ uIdx ][ vIdx ][ vertex ] | se(v) |
| **}** |  |
| } |  |
| } |  |
| } |  |

The Figure 3 depicts the same configuration as in Figure 2 but with the proposed modification. The explicitly-encoded vertices are arranged now uniformly in the LUT and the relative distance of vertex derivation is reduced.



Figure 3: Example of LUT coding tree with explicitly-decoded vertices in red and non-explicitly-decoded vertices marked with “?”, with the proposed syntax modification.

# Conclusion

We propose a bug fix in the current draft spec to arrange the explicitly-encoded vertices nigh-uniformly in the color mapping table during the decoding process. The performance of the SHM encoder is not impacted by the bug fix. The SHM encoder and decoder should be modified as in the provided patch on top of SHM6.0.

# Patent rights declaration(s)

**Technicolor 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).**

**InterDigital Communications Inc 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).**