|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  20th Meeting: Geneva, CH, 10–18 Feb. 2015 | Document: JCTVC-T0206 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-CE1: On palette sharing mode** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Yuwen He, Xiaoyu Xiu, Yan Ye  9710 Scranton Rd, #250 San Diego, CA 92121, USA | Tel: Email: | + 1-858-210-4830 Yuwen.He@InterDigital.com [Xiaoyu.Xiu@InterDigital.com](mailto:Xiaoyu.Xiu@InterDigital.com)  [Yan.Ye@InterDigital.com](mailto:Yan.Ye@InterDigital.com) |
| *Source:* | InterDigital Communications, LLC | | |

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

# Abstract

This contribution generalizes the palette sharing mode. The palette\_share\_flag semantics is modified to indicate the presence of new palette colour entries. The syntax element palette\_num\_signalled\_entries is modified to be palette\_num\_signalled\_entries\_minus1, and is present only when palette\_share\_flag is equal to 0. Experimental results show that, for 4:4:4 lossy coding, the proposed method reportedly provides the average {Y, Cb, Cr} BD-rate savings for AI, RA and LB of {xx%, xx%, xx%}, {xx%, xx%, xx%} and {xx%, xx%, xx%}, respectively; for 4:4:4 lossless coding, the corresponding bit-rate saving of the proposed method for AI, RA and LB are xx%, xx% and xx%, respectively.

# Introduction

In the current palette mode design, the palette sharing mode reuses the first N palette table entries from the palette predictor, where N is set equal to the number of palette table entries of the last palette coded CU. This contribution proposes to generalize the palette sharing mode in the following way:

* The palette\_share\_flag semantics is modified to indicate whether any new palette colors are signaled for the current CU. If palette\_share\_flag is equal to 1, all palette table entries of the current CU are (partially) inherited from the palette predictor, that is, no new colors are signaled. Otherwise, if palette\_share\_flag is equal to 0, new palette colors are signaled.
* The palette\_num\_signalled\_entries is signaled only when palette\_share\_flag is equal to 0. Further, the value of palette\_num\_signalled\_entries is prohibited from being equal to 0. Instead of palette\_num\_signalled\_entries, a modified syntax element palette\_num\_signalled\_entries\_minus1 is signaled.

It is further asserted that this removes the need to keep track of the palette table size of the last palette coded CU, PreviousPaletteSize, and simplifies the palette table derivation process.

# Simulation results

The proposed methods are implemented based on SCM-3.0 and tested under the SCC common test conditions [2].

For 4:4:4 lossy coding, the proposed methods provide the average {G/Y, B/Cb, R/Cr} BD-rate savings for AI, RA and LB of …, respectively. For lossless coding, the corresponding average bit-rate savings for AI, RA and LB are … respectively.

Table 1 BD-rate performance for 4:4:4 lossy coding (To be updated)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **All Intra** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | -0.1% | 0.0% | 0.0% |
| RGB, mixed content, 1440p & 1080p | -0.1% | -0.1% | -0.1% |
| RGB, Animation, 720p | 0.0% | 0.0% | 0.0% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | 0.0% | 0.0% | 0.0% |
| YUV, mixed content, 1440p & 1080p | -0.1% | -0.2% | -0.1% |
| YUV, Animation, 720p | 0.0% | 0.0% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 103% | | |
| Dec Time[%] | 100% | | |
|  |  |  |  |
|  | **Random Access** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | #VALUE! | #VALUE! | #VALUE! |
| RGB, mixed content, 1440p & 1080p | -0.1% | -0.1% | -0.2% |
| RGB, Animation, 720p | 0.0% | 0.0% | -0.1% |
| RGB, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| YUV, text & graphics with motion, 1080p & 720p | #VALUE! | #VALUE! | #VALUE! |
| YUV, mixed content, 1440p & 1080p | -0.2% | -0.4% | -0.3% |
| YUV, Animation, 720p | -0.1% | -0.2% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | #VALUE! | | |
| Dec Time[%] | #VALUE! | | |
|  |  |  |  |
|  | **Low delay B** | | |
|  | G/Y | B/U | R/V |
| RGB, text & graphics with motion, 1080p & 720p | #VALUE! | #VALUE! | #VALUE! |
| RGB, mixed content, 1440p & 1080p | -0.1% | 0.2% | 0.0% |
| RGB, Animation, 720p | 0.0% | 0.0% | -0.1% |
| RGB, camera captured, 1080p | #VALUE! | #VALUE! | #VALUE! |
| YUV, text & graphics with motion, 1080p & 720p | #VALUE! | #VALUE! | #VALUE! |
| YUV, mixed content, 1440p & 1080p | -0.2% | -0.6% | -0.2% |
| YUV, Animation, 720p | 0.0% | 0.1% | 0.0% |
| YUV, camera captured, 1080p | 0.0% | -0.1% | 0.0% |
| Enc Time[%] | #VALUE! | | |
| Dec Time[%] | #VALUE! | | |

Table 2 Bit-rate performance for 4:4:4 lossless coding

# Proposed specification changes

#### 7.3.8.8 Palette syntax

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | Descriptor |
| **palette\_share\_flag**[ x0 ][ y0 ] | ae(v) |
| ~~if( !palette\_share\_flag[ x0 ][ y0 ] ) {~~ |  |
| palettePredictionFinished = 0 |  |
| paletteNumPredictedEntries = 0 |  |
| for( i = 0; i < PredictorPaletteSize  && !palettePredictionFinished &&  paletteNumPredictedEntries < palette\_max\_size; i++ ) { |  |
| **palette\_predictor\_run** | ae(v) |
| if( palette\_predictor\_run != 1 ) |  |
| if( palette\_predictor\_run > 1 ) |  |
| i += palette\_predictor\_run − 1 |  |
| PalettePredictorEntryReuseFlag[ i ] = 1 |  |
| paletteNumPredictedEntries ++ |  |
| } else |  |
| palettePredictionFinished = 1 |  |
| } |  |
| if( !palette\_share\_flag[ x0 ][ y0 ] ) { |  |
| if( paletteNumPredictedEntries < palette\_max\_size ~~&& !palette\_share\_flag~~ ) |  |
| **palette\_num\_signalled\_entries\_minus1** | ae(v) |
| for( cIdx = 0; cIdx < 3; cIdx++ ) |  |
| for( i = 0; i <= palette\_num\_signalled\_ entries\_minus1; i++ ) |  |
| **palette\_entry** | ae(v) |
| } |  |
| if( currentPaletteSize != 0 ) |  |
| **palette\_escape\_val\_present\_flag** | ae(v) |
| if( palette\_escape\_val\_present\_flag ) { |  |
| if( cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) { |  |
| **cu\_qp\_delta\_palette\_abs** | ae(v) |
| if( cu\_qp\_delta\_palette\_abs ) |  |
| **cu\_qp\_delta\_palette\_sign\_flag** | ae(v) |
| } |  |
| if( cu\_chroma\_qp\_offset\_enabled\_flag && !IsCuChromaQpOffsetCoded ) { |  |
| **cu\_chroma\_qp\_palette\_offset\_flag** | ae(v) |
| if( cu\_chroma\_qp\_offset\_flag && chroma\_qp\_offset\_list\_len\_minus1 > 0 ) |  |
| **cu\_chroma\_qp\_palette\_offset\_idx** | ae(v) |
| } |  |
| } |  |
| if( indexMax > 0) |  |
| **palette\_transpose\_flag** | ae(v) |
| … |  |
| } |  |

**palette\_share\_flag**[ x0 ][ y0 ] equal to 1 specifies that no new palette colour entries are signalled for the current coding unit. ~~that the palette for the current coding unit is derived by copying the first PreviousPaletteSize entries from the predictor palette. The variable PreviousPaletteSize is derived as specified in subclause 8.4.5.2.8.~~ palette\_share\_flag [ x0 ][ y0 ] equal to 0 specifies ~~the palette for the current coding unit is specified as a combination of palette entries from previous coding units and~~ new palette entries ~~which~~ are explicitly signalled for the current coding unit.

The variable PalettePredictorEntryReuseFlag[ i ]equal to 1specifies that the i-th entry in the predictor palette is reused in the current palette. PalettePredictorEntryReuseFlag[ i ] equal to 0specifies that the i-th entry in the predictor palette is an entry in the current palette. All elements of the array PalettePredictorEntryReuseFlag[ i ] are initialized to be equal to zero.

~~If palette\_share\_flag [ x0 ][ y0 ] is equal to 1, the array PalettePredictorEntryReuseFlag[ i ] is further modified as follows:~~

~~for( i = 0; i < PreviousPaletteSize; i++ ) (7‑70)  
 PalettePredictorEntryReuseFlag[ i ] = 1~~

**palette\_num\_signalled\_entries\_minus1** plus 1 specifies the number of entries in the current palette that are explicitly signalled.

When palette\_num\_signalled\_ entries is not present, it is inferred to be equal to 0.

The variable CurrentPaletteSize specifies the size of the current palette and is derived as follows:

* ~~If palette\_share\_flag [ x0 ][ y0 ] is equal to 1,~~

~~CurrentPaletteSize = PreviousPaletteSize (7‑71)~~

* ~~Otherwise (palette\_share\_flag [ x0 ][ y0 ] is equal to 0)~~

CurrentPaletteSize = paletteNumPredictedEntries + palette\_share\_flag ? 0: palette\_num\_signalled\_entries\_minus1 + 1 (7‑72)

The value of CurrentPaletteSize shall be in the range of 0 to palette\_max\_size, inclusive.

##### 8.4.5.2.8. Decoding process for palette mode

Inputs to this process are:

– a location ( xCb, yCb ) specifying the top-left sample of the current block relative to the top-left sample of the current picture,

– a variable cIdx specifying the colour component of the current block,

– a variable nCbS specifying the size of the current block,

– An array paletteSampleMode[ xCb + x ][ yCb + y ], with x = 0..nCbS − 1 and  y = 0..nCbS − 1, specifying the method of palette index derivation for each sample in the current block

– The array paletteIndexMap[ xCb + x ][ yCb + y ], with x = 0..nCbS − 1,  y = 0..nCbS − 1, specifying the palette index for each sample in the current block, and [Ed. (GJS): Check "is not" versus "is not equal to" (generally).] [Ed. (GJS): Check/fix use of "the palette mode".]

– The array paletteEscapeVal[cIdx][ xCb + x ][ yCb + y ], with x = 0..nCbS − 1 and  y = 0..nCbS − 1, specifying the escape value (possibly quantized) for the cIdx-th component of each sample in the current block for which the paletteSampleMode[ xCb + x ][ yCb + y ] is equal to ESCAPE\_MODE.

Output of this process is an array recSamples[ x ][ y ], with x = 0..nCbS − 1,  y = 0..nCbS − 1, specifying reconstructed sample values for the block.

Depending on the value of the colour component cIdx, the following assignments are made:

– If cIdx is equal to 0, the function clipCidx1 corresponds to Clip1Y.

– Otherwise, the function clipCidx1 corresponds to Clip1C.

The variables PredictorPaletteSize, ~~PreviousPaletteSize~~ and PredictorPaletteEntries are derived as follows:

– The initialization process as specified in subclause 9.3.2 is invoked when starting the parsing of one or more of the following:

– the slice segment data syntax specified in subclause 7.3.8.1

– the coding tree unit syntax specified in subclause 7.3.8.2 and the coding tree unit is the first coding tree unit in a tile

– the coding tree unit syntax specified in subclause 7.3.8.2, entropy\_coding\_sync\_enabled\_flag is equal to 1, and the associated luma coding tree block is the first luma coding tree block in a coding tree unit row

– Otherwise, PredictorPaletteSize, ~~PreviousPaletteSize,~~ and PredictorPaletteEntries updated after decoding the previous coding unit with palette\_mode\_flag[ x0 ][ y0 ] equal to 1 are used.

The (nCbS x nCbS) block of the reconstructed sample array recSamples at location ( xCb, yCb ) is derived as follows:

…

# References

1. R. Joshi, J. Xu, HEVC Screen Content Coding Draft Text 2, JCTVC-S1005, Oct. 2014, Strasbourg, France.
2. H. Yu, R. Cohen, K. Rapaka, J. Xu, Common Test Conditions for Screen Content Coding, JCTVC-S1015, Oct. 2014, Strasbourg, France.

# Patent rights declaration(s)

**InterDigital Communications LLC 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).**