|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  19th Meeting: Strasbourg, FR, 17–24 Oct. 2014 | Document: JCTVC-S0258 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-CE6: Unification of coding of escape indices and other palette indices** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Xiaoyu Xiu, Yuwen He, Yan Ye 9710 Scranton Rd, #250 San Diego, CA 92121, USA  Vadim Seregin, Rajan Joshi, Marta Karczewicz, Wei Pu, Joel Sole  5775 Morehouse Drive San Diego, CA 92121, USA | Email: | [Xiaoyu.Xiu@InterDigital.com](mailto:Xiaoyu.Xiu@InterDigital.com)  [vseregin@qti.qualcomm.com](mailto:vseregin@qti.qualcomm.com) |
| *Source:* | InterDigital Communications, LLC  Qualcomm Incorporated | | |

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

# Abstract

In SCM2.0, escape coded samples are assigned the highest palette index (after expanding the palette index by 1). But in coding of palette indices, the escape index is treated differently from other palette indices. This contribution proposes a method that is asserted to unify the coding of escape index and other palette indices. The proposed method uses elements from JCTVC-S0157 (solution 3) and JCTVC-S0181. BD-rate results for the proposed method with and without encoder modification are presented. Without encoder modification, for the category of text and graphics with motion 1080p, the BD-rate saving for the Y/G component is 0.1% for the All-Intra lossy configuration, and the bit-rate savings for the Y/G component 0.6% for the All-intra lossless configuration. With encoder modification, the corresponding BD-rate savings and bit-rate savings are 0.4% and 0.8% for lossy and lossless configurations, respectively.

# Proposal

In SCM2.0, escape coded samples are assigned the highest palette index (after expanding the palette index by 1). But in coding of palette indices, the escape index is treated differently from other palette indices. For example, although escape index is signalled in the 'INDEX' mode, no run is signalled. Similarly, 'COPY\_ABOVE' mode may not start with an escape index and escape samples may not be included in the 'COPY\_ABOVE' runs. JCTVC-S0157 and JCTVC-S0181 proposed some alternatives to address these inconsistencies. In this proposal, a method unifying the coding of escape index and other palette indices is proposed. The proposed method uses elements from JCTVC-S0157 (solution 3) and JCTVC-S0181. The main aspects of the proposed method are:

1. When escape index is signalled as 'INDEX' mode, a run value is coded to specify how many subsequent indices are also escapes. This is followed by coding of escape values.
2. It is possible to start a 'COPY\_ABOVE' mode with escape index. Furthermore, escape indices may be copied as a part of 'COPY\_ABOVE' run. After signalling the 'COPY\_ABOVE' run, escape values are coded depending on the number of escape indices copied from above.

The asserted advantage of the proposed method is that escape index and other indices are treated identically during the coding of palette indices, which simplifies the draft specification text. Additionally, as pointed out in JCTVC-S0181, the proposed method can also remove the parsing dependency of truncated binary coding (TBC) maximum level on escape color indication. It should be noted that in the current HEVC SCC specification JCTVC-R1005-v3, 'COPY\_ABOVE' mode runs may include escape indices. But the corresponding escape values are not signalled. And, this specification bug is fixed after applying the proposed method.

# Simulation Results

The proposed methods are implemented on top SCM 2.0 and simulated under common test conditions JCTVC-R1015. The simulation platform is a homogenous LINUX cluster consisting of Intel(R) XEON CPUs. Two sets of results are presented, one without any changes to the encoder, the other one with encoder changes similar to those proposed in JCTVC-S0156.

Table 1 shows BD-rate results for the proposed method without (left) and with encoder modification (right) for All-Intra lossy configuration. Table 2 shows the corresponding results for lossless configuration. The anchor is SCM2.0. The full results including other configurations are available in the accompanying spreadsheets.



Table 1: BD-rate performance for the proposed method without encoder modification (left) and with encoder modification (right) for All-Intra lossy configuration (anchor SCM2.0)



Table 2: BD-rate performance for the proposed method without encoder modification (left) and with encoder modification (right) for All-Intra lossless configuration (anchor SCM2.0)

# Conclusions

This contribution proposes a method to unify the coding of escape index and other palette indices. The proposed method uses elements from JCTVC-S0157 (solution 3) and JCTVC-S0181. BD-rate results for the proposed method with and without encoder modification are presented. Without encoder modification, for the category of text and graphics with motion 1080p, the BD-rate saving for the Y/G component is 0.1% for the All-Intra lossy configuration, and the bit-rate savings for the Y/G component 0.6% for the All-intra lossless configuration. With encoder modification, the corresponding BD-rate savings and bit-rate savings are 0.4% and 0.8% for lossy and lossless configurations, respectively.

It is proposed to adopt this method into the next version of the screen content working draft and software model.

# Proposed specification change

|  |  |
| --- | --- |
| palette\_coding( x0, y0, nCbS ) { | Descriptor |
| **palette\_transpose\_flag** | ae(v) |
| **palette\_share\_flag**[ x0 ][ y0 ] | ae(v) |
| if( palette\_share\_flag[ x0 ][ y0 ] ) { |  |
| palette\_size = previousPaletteSize |  |
| for( n = 0; n < palette\_size; n++ ) |  |
| for( cIdx = 0; cIdx < 3; cIdx++ ) |  |
| palette\_entries[ cIdx ][ n ] = previousPaletteEntries[ cIdx ][ n ] |  |
| } else { |  |
| numPredPreviousPalette = 0 |  |
| for( i = 0; i < previousPaletteStuffingSize; i++ ) |  |
| previous\_palette\_entry\_flag[ i ] = 0 |  |
| palette\_last\_group = 0 |  |
| for( i = 0; i < previousPaletteStuffingSize && !palette\_last\_group &&  numPredPreviousPalette < max\_palette\_size; i++ ) { |  |
| lastPossibleGroupFlag = ( i + 4 >= previousPaletteStuffingSize ) |  |
| lastIdx = min( i + 4, previousPaletteStuffingSize ) – 1 |  |
| if( i > 3 && !lastPossibleGroupFlag ) |  |
| **palette**\_**all\_zeros\_in\_group** | ae(v) |
| Else |  |
| palette\_all\_zeros\_in\_group = 0 |  |
| if( palette\_all\_zeros\_in\_group ) |  |
| i += 4 |  |
| else { |  |
| numOnesInGroup = 0 |  |
| for( idx = i; idx <= lastIdx && numPredPreviousPalette < max\_palette\_size;  idx++ ) { |  |
| if ( idx = = lastIdx && numOnesInGroup = = 0 ) |  |
| previous\_palette\_entry\_flag[ idx ] = 1 |  |
| Else |  |
| **previous\_palette\_entry\_flag**[ idx ] | ae(v) |
| if ( previous\_palette\_entry\_flag[ idx ] ) { |  |
| for ( cIdx = 0; cIdx < 3; cIdx++ ) |  |
| palette\_entries[ cIdx ][ numPredPreviousPalette ] =   previousPaletteEntries[ cIdx ][ idx ] |  |
| numPredPreviousPalette++ |  |
| numOnesInGroup++ |  |
| } |  |
| } |  |
| if( !palette\_all\_zeros\_in\_group &&   !lastPossibleGroupFlag && numPredPreviousPalette < max\_palette\_size ) |  |
| **palette**\_**last\_group** | ae(v) |
| } |  |
| } |  |
| if( numPredPreviousPalette < max\_palette\_size) |  |
| **num\_signalled\_palette\_entries** | ae(v) |
| for( cIdx = 0; cIdx < 3; cIdx++ ) |  |
| for( i = 0; i < num\_signalled\_palette\_entries; i++ ) |  |
| **palette\_entries**[ cIdx ][ numPredPreviousPalette + i ] | ae(v) |
| palette\_size = numPredPreviousPalette + num\_signalled\_palette\_entries |  |
| } |  |
| **palette\_escape\_val\_present\_flag** | ae(v) |
| if( palette\_escape\_val\_present\_flag ) |  |
| indexMax = palette\_size |  |
| Else |  |
| indexMax = palette\_size – 1 |  |
| scanPos = 0 |  |
| while( scanPos < nCbS \* nCbS ) { |  |
| xC = x0 + travScan[ scanPos ][ 0 ] |  |
| yC = y0 + travScan[ scanPos ][ 1 ] |  |
| if( scanPos > 0) { |  |
| xC\_prev = x0 + travScan[ scanPos − 1 ][ 0 ] |  |
| yC\_prev = y0 + travScan[ scanPos − 1 ][ 1 ] |  |
| } |  |
| if( scanPos > = nCbS && palette\_mode[xC\_prev][yC\_prev] ! = COPY\_ABOVE ) |  |
| **palette\_mode**[ xC ][ yC ] | ae(v) |
| if( palette\_mode[ xC ][ yC ] ! = COPY\_ABOVE ) { |  |
| adjustedIndexMax = indexMax |  |
| adjustedRefIndex = indexMax + 1 |  |
| } |  |
| if( scanPos > 0 ~~&& palette\_mode[xC\_prev][yC\_prev] ! = ESCAPE~~) { |  |
| if( palette\_mode[xC\_prev][yC\_prev] = = INDEX ) { |  |
| ~~adjustedIndexMax − = 1~~ |  |
| adjustedRefIndex = paletteMap[ xC\_prev ][ yC\_prev ] |  |
| } |  |
| else~~if( scanPos > = nCbS && palette\_mode[ xC\_prev ][ yC\_prev ] = = COPY\_ABOVE~~   ~~&& palette\_mode[ xC ][ yC − 1 ] ! = ESCAPE~~) { |  |
| ~~adjustedIndexMax − = 1~~ |  |
| adjustedRefIndex = paletteMap[ xC ][ yC − 1 ] |  |
| } |  |
| adjustedIndexMax −= 1 |  |
| } |  |
| if(palette\_mode[ xC ][ yC ] ! = COPY\_ABOVE ) { |  |
| if( adjustedIndexMax > 0 ) |  |
| **palette\_index** | ae(v) |
| if( palette\_index > = adjustedRefIndex ) |  |
| palette\_index++ |  |
| ~~if( palette\_index = = palette\_size ) {~~ |  |
| ~~for( cIdx = 0; cIdx < 3; cIdx++ ) {~~ |  |
| **~~palette\_escape\_val~~** | ae(v) |
| ~~paletteEscapeVal[ cIdx ][ xC ][ yC ] = palette\_escape\_val~~ |  |
| ~~}~~ |  |
| ~~palette\_mode[ xC ][ yC ] = ESCAPE~~ |  |
| ~~scanPos++~~ |  |
| ~~}~~ |  |
| } |  |
| ~~if(palette\_mode[xC][yC] ! = ESCAPE) {~~ |  |
| **palette\_run** | ae(v) |
| runPos = 0 |  |
| runMode = palette\_mode[ xC ][ yC ] |  |
| while ( runPos < = palette\_run ) { |  |
| xC = x0 + travScan[ scanPos ][ 0 ] |  |
| yC = y0 + travScan[ scanPos ][ 1 ] |  |
| if( ~~palette\_mode[ xC ][ yC ]~~ runMode = = INDEX ) { |  |
| palette\_mode[ xC ][ yC ] = INDEX |  |
| paletteMap[ xC ][ yC ] = palette\_index |  |
| } else { |  |
| palette\_mode[ xC ][ yC ] = COPY\_ABOVE |  |
| paletteMap[ xC ][ yC ] = paletteMap[ xC ][ y − 1 ] |  |
| } |  |
| if( paletteMap[ xC ][ yC ] = = palette\_size ) { |  |
| for( cIdx = 0; cIdx < 3; cIdx++ ) { |  |
| **palette\_escape\_val** | ae(v) |
| paletteEscapeVal[ cIdx ][ xC ][ yC ] = palette\_escape\_val |  |
| } |  |
| } |  |
| runPos++ |  |
| scanPos++ |  |
| } |  |
| ~~}~~ |  |
| } |  |
| tempPaletteEntries[ cIdx ][ i ] = palette\_entries[ cIdx ][ i ] |  |
| for( i = 0; i < previousPaletteStuffingSize && current\_size < max\_palette\_predictor\_size;   i++ ) |  |
| if( previous\_palette\_entry\_flag[ i ] = = 0 ) { |  |
| for ( cIdx = 0; cIdx < 3; cIdx++ ) |  |
| tempPaletteEntries[ cIdx ][ current\_size ] = previousPaletteEntries[ cIdx ][ i ] |  |
| current\_size++ |  |
| } |  |
| previousPaletteStuffingSize = current\_size |  |
| previousPaletteEntries = tempPaletteEntries |  |
| } |  |

# References

1. R. Joshi, J. Xu, HEVC Screen Content Coding Draft Text 1, JCTVC-R1005, July 2014, Sapporo, Japan.
2. H. Yu, R. Cohen, K. Rapaka, J. Xu, Common Test Conditions for Screen Content Coding, JCTVC-R1015, August 2014, Sapporo, Japan.
3. W. Pu, F. Zou, V. Seregin, R. Joshi, M. Karczewicz, and J. Sole, Non-CE6: Palette parsing dependency and palette encoder improvement, JCTVC-S0157, October 2014, Strasbourg, France.
4. X. Xiu, Y. He, Y. Ye, Non-CE6: Removal of parsing dependency in palette-based coding, JCTVC-S0181, October 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).**

**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).**