|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  9th Meeting: Geneva, CH, 27th April – 7th May, 2012 | Document: JCTVC-I0348\_r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **On reference picture list modification** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Adarsh K. Ramasubramonian Ying Chen Ye-Kui Wang  5775 Morehouse Drive, San Diego, CA 92121-1714 USA | Tel: Email: | 1-858-658-5804 [aramasub@qualcomm.com](mailto:aramasub@qualcomm.com)  1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

In this proposal, shortcomings of the reference picture list modification (RPLM) design in the latest HEVC draft spec (WD 6) are discussed. Based on the discussion, a changed RPLM design based on the one in HEVC WD 5 is proposed. It is reported by the proponents that, for test cases 2.8 and 3.5 in the common test conditions for reference picture marking and list construction proposals in JCTVC-H0725, 24% bit reduction of RPLM bits was achieved for the low-delay configuration compared to the RPLM method in HEVC WD 6, the performance is the same for the random access configuration. It is further that the proposed RPLM method, when applied to HEVC-based 3DV, outperforms the RPLM method in HEVC WD 6, when applied to 3DV, with 34% bit reduction on average for non-base views under the 3DV common test conditions.

# Shortcomings of the RPLM method in HEVC WD6

The current design in HEVC WD6 has the following problems compared with the original design in HEVC WD 5.

The reference picture list modification (RPLM) design in HEVC WD 5 follows the similar fashion as in AVC, so that the number of commands can be much less than the number of commands needed for the RPLM in the current draft, especially for the following examples:

1. Example 1: Five pictures in the initial list are to be put in the final list, and only the first two entries in the initial list need to be swapped to get the final list. For the AVC design, only one command is needed while for the design in HEVC WD 6, five commands are needed.
2. Example 2: Five pictures are to be put in the final list. A picture at the end of the initial list or not included in the initial list needs to be added at the first entry of the final list, the last entry is removed and the position of the other entries have their positions shifted by one. This is a typical case, e.g., in MVC.

Note that if in the WD 5 design, if list\_modification\_idc and ref\_pic\_set\_idx are both coded as u(v) instead of ue(v), it is more efficient than WD 6 in LD case (15% bit-saving).

# Proposal

The proposed syntax, semantics and decoding process changes are as follows.

## Slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **…** |  |
| if( lists\_modification\_present\_flag ) { |  |
| ~~ref\_pic\_list\_modification( )~~ |  |
| if( slice\_type != 2 ) |  |
| ref\_pic\_list\_modification( 0 ) |  |
| if( slice\_type = = 1 ) |  |
| ref\_pic\_list\_modification( 1 ) |  |
| ref\_pic\_list\_combination( ) |  |
| } |  |
| **…** |  |
| } |  |

## Reference picture list modification syntax

|  |  |
| --- | --- |
| ref\_pic\_list\_modification( X ) { |  |
| **ref\_pic\_list\_modification\_flag\_lX** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_lX ) { |  |
| **start\_pos\_lX** | u(v) |
| if( start\_pos\_lX < num\_ref\_idx\_lX\_active\_minus1 ) |  |
| **num\_inserted\_pics\_lX\_minus1** | ue(v) |
| for( i =0 ; i <= num\_inserted\_pics\_lX\_minus1; i++ ) { |  |
| if( NumberSourceRPS >1 ) |  |
| **source\_list\_idx\_lX[** i **]** | u(v) |
| if( NumPicsInSourceList[ source\_list\_lX[ i ] ] > 1 ) |  |
| **ref\_pic\_set\_idx\_lX[** i **]** | u(v) |
| } |  |
| } |  |
| } |  |

## Semantics

**ref\_pic\_list\_modification\_flag\_lX** (X being 0 or 1)equal to 1 specifies that the syntax element start\_pos\_lX is present in the syntax structure ref\_pic\_list\_modification( X ), and the initial RefPicListX are to be modified. ref\_pic\_list\_modification\_flag\_lX equal to 0 specifies that the syntax element start\_pos\_lX is not present in the syntax structure ref\_pic\_list\_modification( X ), and the initial RefPicListX are not to be modified hence the final RefPicListX is identical to the initial RefPicListX.

**start\_pos\_lX** specifies the starting position of the initial list for reference picture list modification. The entries with a position smaller than start\_pos\_lX will not be modified. It is in the range of 0 to num\_ref\_idx\_lX\_active\_minus1, inclusive. When not present, it is inferred to be equal to 0. The length of start\_pos\_lX is Ceil( log2( num\_ref\_idx\_lX\_active\_minus1 + 1 ) ) bits.

**num\_inserted\_pics\_lX\_minus1** plus 1specifies the number of entries signalled in the loop in this syntax structure ref\_pic\_list\_modification( X ). The value of num\_inserted\_picsX\_minus1 is in the range of 0 to num\_ref\_idx\_lX\_active\_minus1 – start\_pos\_lX, inclusive. When not present, the value of num\_inserted\_pics\_lX\_minus1 is inferred to be equal to 0.

The arrays RPSSubsetX[ ] and NumPicsInSourceList[ ], and the variable NumberSourceRPS are derived as follows:

RefPicSetStCurr0 = (X = =0) ? RefPicSetStCurrBefore: RefPicSetStCurrAfter  
 RefPicSetStCurr1 = (X == 0) ? RefPicSetStCurrAfter: RefPicSetStCurrBefore

NumPicsStCurr0 = (X = =0) ? NumPocStCurrBefore: NumPocSetStCurrAfter  
 NumPicsStCurr1 = (X == 0) ? NumPocStCurrAfter: NumPocStCurrBefore

* i is initialized to 0
* if RefPicSetStCurr0 is non-empty, then RPSSubsetX[ i ] = RefPicSetStCurr0, NumPicsInSourceList[ i ] = NumPocStCurr0, and i = i + 1
* if RefPicSetStCurr1 is non-empty, then RPSSubsetX[ i ] = RefPicSetStCurr1, NumPicsInSourceList[ i ] = NumPocStCurr1, and i = i + 1
* if RefPicListLtCurr is non-empty, then RPSSubsetX[ i ] = RefPicListLtCurr, NumPicsInSourceList[ i ] = NumPocLtCurr, and i = i + 1
* NumberSourceRPS = i

**source\_list\_idx\_lX[**i**]** specifies the index to the list of candidate RPS subsets of the i-th reference picture to be moved to the current index (at that moment) in RefPicListX. The value of source\_list\_idx\_lX[ i ] is in the range of 0 to NumberSourceRPS - 1, inclusive. The length is Ceil( log2( NumberSourceRPS ) ) bits. When not present, the value of source\_list\_idx\_lX[ i ] is inferred to be equal to 0.

**ref\_pic\_set\_idx\_lX[** i **]** specifies the index, to RPSSubsetX[ source\_list\_idx\_lX[ i ] ], of the i-th reference picture to be moved to the current index in RefPicListX. ref\_pic\_set\_idx\_lX[ i ] is in the range of 0 to RPSSubsetX[ source\_list\_idx\_lX[ i ] ] – 1, inclusive. The number of bits needed to signal ref\_pic\_set\_idx\_lX[ i ] is Ceil( log2( RPSSubsetX[ source\_list\_idx\_lX[ i ] ] ) ). When not present, ref\_pic\_set\_idx\_lX[ i ] is inferred to be equal to 0.

## Decoding process for reference picture list construction

**Initialisation process for reference picture lists**

This initialisation process is invoked when decoding a P or B slice header.

When decoding a P or B slice, there shall be at least one reference picture in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr.

The following procedure is conducted once for a P slice with X equal to 0 and twice for a B slice with X equal to 0 and 1 respectively to construct the initial RefPicListX (with X equal to 0 or 1):

cIdx = 0  
nEntries = num\_ref\_idx\_lX\_active\_minus1 + 1  
while( cIdx < nEntries )  
 for ( RPSSetIdx = 0; RPSSetIdx < NumberSourceRPS; RPSSetIdx++ )  
 for ( i = 0; i < NumPicsInSourceList[ RPSSetIdx ] && cIdx < nEntries; i++, cIdx++)  
 RefPicListX[ cIdx ] = RPSSubsetX[ RPSSetIdx ][ i ]

**Modification process for reference picture lists**

Input for this process is a list RefPicListX (with X being 0 or 1).

The variable NumPicsInSourceList is set equal to RPSSubsetX[ source\_list\_idx\_lX[ i ] ].The following procedure is conducted to derive the modified list

if( ref\_pic\_list\_modification\_lX\_flag )

for( cIdx = start\_pos\_lX, i = 0; i <= num\_inserted\_pics\_lX\_minus1; i++ ) {

for( nIdx = num\_ref\_idx\_lX\_active\_minus1 + 1; nIdx > cIdx; nIdx− − )  
 RefPicListX[ nIdx ] = RefPicListX[ nIdx − 1]

RefPicListX[ cIdx ] = RPSSubsetX[ source\_list\_idx\_lX[ i ] ][ ref\_pic\_set\_idx\_lX[ i ] ]

for( pIdx = cIdx+1, nIdx = cIdx+1; nIdx <= num\_ref\_idx\_lX\_active\_minus1+1; nIdx++ )

if( PicOrderCnt( RefPicListX[ cIdx ] ) != PicOrderCnt( RefPicListX[ nIdx ] ) )

RefPicListX[ pIdx++ ] = RefPicListX[ nIdx ];

}

# Signaling efficiency

We present below a bit-count comparison for parameters present in the syntax structures ref\_pic\_list\_modification( 0 ) and ref\_pic\_list\_modification( 1 ) for the test cases 2.8 and 3.5 in the common test conditions for reference picture marking and list construction proposals in [1]. The test results show that the proposed method provides overall bit-savings in the signalling compared to the reference picture list modification design in HEVC WD 6.

|  |  |  |
| --- | --- | --- |
|  | HEVC WD 6 | Proposed method |
| Test 1: Low-delay |  |  |
| ref\_pic\_list\_modfication related bit-count | 7530 | 5752 |
| % reduction |  | 24% |
| Test 2: Random access |  |  |
| ref\_pic\_list\_modfication related bit-count | 1172 | 1172 |
| % reduction |  | 0% |

It is reported in the MPEG document m24944 that the proposed reference picture list modification method, when applied to HEVC-based 3DV, outperforms the reference picture list modification method in HEVC WD 6 as extended to 3DV. The proposed modification method provides a 34% bit reduction, on an average, compared to the modification method in HEVC WD 6 for non-base views in the 3DV common test conditions.

# References

[1] Y-K. Wang, M. M. Hannuksela, T. K. Tan, R. Sjöberg, and Yan Ye, “Common conditions for reference picture marking and list construction proposals,” JCT-VC document JCTVC-H0725, San Jose, February 2012.

# Patent rights declaration(s)

**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).**