|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  8th Meeting: San José, CA, USA, 1–10 February, 2012 | Document: JCTVC-H0138\_r2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG21: Unification of reference picture list modification processes** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Yong He Yan Ye  9710 Scranton Rd, Suite 250 San Diego, CA 92121, USA | Tel: Email: | +1-858-210-4803 [yong.he@interdigital.com](mailto:yong.he@interdigital.com)  [yan.ye@interdigital.com](mailto:yan.ye@interdigital.com) |
| *Source:* | InterDigital Communications, LLC | | |

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

# Abstract

Currently HEVC WD5 specifies two different methods to modify the reference picture lists List0/List1 and the combined list. In this contribution, unification of reference picture list modification process for List0/List1 and combined list is proposed to simplify the syntax and modification process of reference picture lists List 0/List 1. Additionally, an encoder-only L0/L1 reordering method was implemented in the HM5.0 software. Simulation results using HM5.0 RA testing conditions show that the proposed syntax consistently achieves 47% reduction in signalling overhead for ref\_pic\_list\_modification().

# Introduction

The reference picture list modification syntax for list 0 and list 1 in Table 1 [3] was adopted at the 7th JCTVC meeting and was incorporated into WD5\_d6 [1]. Table 2 shows the list\_modification\_idc operations for modification of reference picture lists.

1. Reference picture list modification syntax for List 0 and List 1

|  |  |
| --- | --- |
| ref\_pic\_list\_modification( ) { | Descriptor |
| if( slice\_type != 2 ) { // P slice or B slice |  |
| **ref\_pic\_list\_modification\_flag\_l0** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l0 ) |  |
| do { |  |
| **list\_modification\_idc** | ue(v) |
| if( list\_modification\_idc != 3 ) |  |
| **ref\_pic\_set\_idx** | ue(v) |
| } while( list\_modification\_idc != 3 ) |  |
| } |  |
| if( slice\_type = = 1 ) { // B slice |  |
| **ref\_pic\_list\_modification\_flag\_l1** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l1 ) |  |
| do { |  |
| **list\_modification\_idc** | ue(v) |
| if( list\_modification\_idc != 3 ) |  |
| **ref\_pic\_set\_idx** | ue(v) |
| } while( list\_modification\_idc != 3 ) |  |
| } |  |
| } |  |

1. list\_modification\_idc operations for modification of reference picture lists

|  |  |
| --- | --- |
| **list\_modification\_idc** | **modification specified** |
| 0 | For list 0: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr0; For list 1: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr1 |
| 1 | For list 0: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr1; For list 1: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr0 |
| 2 | ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetLtCurr |
| 3 | End loop for modification of the initial reference picture list |

When ref\_pic\_list\_modification\_flag\_l0 is equal to 1 or, when decoding a B slice, ref\_pic\_list\_modification\_flag\_l1 is equal to 1, the initial reference picture list RefPicList0 and, for B slices, RefPicList1 are modified. The procedure is conducted by placing the reference picture into the index position, and shift the position of any other remaining pictures in the list to later in the list. The modification loop ends when list\_modification\_idc is equal to 3. The reference picture is chosen from reference picture subsets, RefPicSetCurr0, RefPicSetCurr1 or RefPicSetLtCurr depending on list\_modification\_idc (in Table 2) and ref\_pic\_set\_idx.

shows an exemplar reference picture list modification process using the current list modification syntax in Table 1. For simplicity of explanation, only short term reference pictures are used in this example. The example considers only List 0 modification, but List 1 modification adopts the same method. As shown, 5 steps are used to perform the desired modification process, at each step except the last, two syntax elements are sent, and at the last step, one syntax element is sent.



**Figure 1. An example of ref\_pic\_list\_modification process for L0**

# Proposed modifications

Referring to documents JCTVC-D421 [4] and JCTVC-H0137 [5] on the modification process of the combined list LC, it is noted that the modification processes for combined list LC and for L0/L1 in Table 1 are not unified. Modification of combined list LC is more straightforward as each entry in the modified list is signalled explicitly. We propose to adopt the LC modification process for L0 and L1 as well. Using the same example in Figure 1, and adopting the principle of explicitly sending each entry in the modified list, the modification process for L0 is illustrated in Figure 2 using the proposed method (syntax in Table 3). As shown, 4 steps are used to perform the desired modification. At each step, only one syntax element, ref\_pic\_set\_idx, is sent, and list\_modification\_idc is no longer necessary.

As shown in the following sections 2.1 to 2.3, with the proposed method, the WD text on reference picture list construction is significantly simplified. WD text reduction is summarized as follows:

* syntax table: reduced by 4 lines
* semantics: reduced by ~10 lines, addionally 1 table removed
* decoding process: reduced by ~35 lines



**Figure 2. Example of ref\_pic\_list\_modification process for L0 using proposed syntax**

## Proposed syntax and semantics hange

The proposed reference picture list modification syntax is given in Table 3. Changes are highlighted in yellow.

The derivation of reference picture temporal lists (RefPicSetCurrTempList0 and RefPicSetCurrTempList1) and NumRpsCurrTempList0/NumRpsCurrTempList1 are specified in 2.2.

1. Proposed Reference picture list modification syntax for List 0 and List 1

|  |  |
| --- | --- |
| ref\_pic\_list\_modification( ) { | Descriptor |
| if( slice\_type != 2 ) { // P slice or B slice |  |
| **ref\_pic\_list\_modification\_flag\_l0** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l0 ) |  |
| ~~do {~~ |  |
| **~~ref\_pic\_list\_modification\_idc~~** | ~~ue(v)~~ |
| ~~if( ref\_pic\_list\_modification\_idc != 3 )~~ |  |
| **~~ref\_pic\_set\_idx~~** | ~~ue(v)~~ |
| ~~} while( ref\_pic\_list\_modification\_idc != 3 )~~ |  |
| for ( i =0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) { |  |
| if ( NumRpsCurrTempList0 > 1 ) |  |
| **ref\_pic\_set\_idx** | u(v) |
| } |  |
| } |  |
| if( slice\_type = = 1 ) { // B slice |  |
| **ref\_pic\_list\_modification\_flag\_l1** | u(1) |
| if( ref\_pic\_list\_modification\_flag\_l1 ) |  |
| ~~do {~~ |  |
| **~~ref\_pic\_list\_modification\_idc~~** | ~~ue(v)~~ |
| ~~if( ref\_pic\_list\_modification\_idc != 3 )~~ |  |
| **~~ref\_pic\_set\_idx~~** | ~~ue(v)~~ |
| ~~} while( ref\_pic\_list\_modification\_idc != 3 )~~ |  |
| for ( i =0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) { |  |
| if (NumRpsCurrTempList1 > 1 ) |  |
| **ref\_pic\_set\_idx** | u(v) |
| } |  |
| } |  |
| } |  |

The syntax element~~s ref\_pic\_list\_modification\_idc and~~ ref\_pic\_set\_idx specif~~y~~ies the change from the initial reference picture lists to the reference picture lists to be used for decoding the slice.

**ref\_pic\_list\_modification\_flag\_l0** equal to 1 specifies that the syntax element ~~ref\_pic\_list\_modification\_idc~~ ref\_pic\_set\_idx is present for specifying reference picture list 0. ref\_pic\_list\_modification\_flag\_l0 equal to 0 specifies that this syntax element is not present.

~~When ref\_pic\_list\_modification\_flag\_l0 is equal to 1, the number of times that ref\_pic\_list\_modification\_idc is not equal to 3 following ref\_pic\_list\_modification\_flag\_l0 shall not exceed num\_ref\_idx\_l0\_active\_minus1 + 1.~~

**ref\_pic\_list\_modification\_flag\_l1** equal to 1 specifies that the syntax element ~~ref\_pic\_list\_modification\_idc~~ ref\_pic\_set\_idx is present for specifying reference picture list 1. ref\_pic\_list\_modification\_flag\_l1 equal to 0 specifies that this syntax element is not present.

~~When ref\_pic\_list\_modification\_flag\_l1 is equal to 1, the number of times that ref\_pic\_list\_modification\_idc is not equal to 3 following ref\_pic\_list\_modification\_flag\_l1 shall not exceed num\_ref\_idx\_l1\_active\_minus1 + 1.~~

**~~ref\_pic\_list\_modification\_idc~~** ~~together with ref\_pic\_set\_idx specifies which of the reference pictures are re-mapped. The values of ref\_pic\_list\_modification\_idc are specified in . The value of the first ref\_pic\_list\_modification\_idc that follows immediately after ref\_pic\_list\_modification\_flag\_l0 or ref\_pic\_list\_modification\_flag\_l1 shall not be equal to 3.~~

~~Table 7‑6 – ref\_pic\_list\_modification\_idc operations for modification of reference picture lists~~

|  |  |
| --- | --- |
| **~~ref\_pic\_list\_modification\_idc~~** | **~~modification specified~~** |
| ~~0~~ | ~~For list 0: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr0; For list 1: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr1~~ |
| ~~1~~ | ~~For list 0: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr1; For list 1: ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetStCurr0~~ |
| ~~2~~ | ~~ref\_pic\_set\_idx is present and corresponds to an index to RefPicSetLtCurr~~ |
| ~~3~~ | ~~End loop for modification of the initial reference picture list~~ |

**ref\_pic\_set\_idx** specifies the index~~, to RefPicSetStCurr0, RefPicSetStCurr1 or RefPicSetLtCurr,~~ of the reference picture ~~being moved to the current index in the reference picture list~~ in RefPicSetCurrTempListX to be placed at the current position of reference picture list LX. The value of ref\_pic\_set\_idx shall be in the range of 0 to max\_num\_ref\_frames, inclusive. If the syntax element ref\_pic\_set\_idx is not present, it is set to 0.

## Proposed initialization process for reference picture lists

The initialization 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 RefPicSetStCurr0, RefPicSetStCurr1 or RefPicSetLtCurr.

The following procedure is conducted to construct the ~~initial RefPicList0~~RefPicSetCurrTempList0:

cIdx = 0  
if (ref\_pic\_list\_modification\_flag\_l0 == 0)

NumRpsCurrTempList0 = num\_ref\_idx\_l0\_active\_minus1+1

else {

NumRpsCurrTempList0 = NumRpsStCurr0 + NumRpsStCurr1 + NumRpsLtCurr

if (NumRpsCurrTempList0 <= num\_ref\_idx\_l0\_active\_minus1)

NumRpsCurrTempList0 = num\_ref\_idx\_l0\_active\_minus1+1

}

while( cIdx <~~= num\_ref\_idx\_l0\_active\_minus1~~ NumRpsCurrTempList0 )  
{  
 for( i=0; i < NumPocStCurr0 && cIdx <~~= num\_ref\_idx\_l0\_active\_minus1~~ NumRpsCurrTempList0; cIdx++, i++ )  
 ~~RefPicList0~~ RefPicSetCurrTempList0 [ cIdx ] = RefPicSetStCurr0[ i ]  
 for( i=0; i < NumPocStCurr1 && cIdx <~~= num\_ref\_idx\_l0\_active\_minus1~~ NumRpsCurrTempList0; cIdx++, i++ )   
 ~~RefPicList0~~ RefPicSetCurrTempList0 [ cIdx ] = RefPicSetStCurr1[ i ]  
 for( i=0; i < NumPocLtCurr && cIdx <~~= num\_ref\_idx\_l0\_active\_minus1~~ NumRpsCurrTempList0; cIdx++, i++ )  
 ~~RefPicList0~~ RefPicSetCurrTempList0 [ cIdx ] = RefPicSetLtCurr[ i ]  
}

If ref\_pic\_list\_modification\_flag\_l0 is 0, the initial RefPicList0 is equivalent to RefPicSetCurrTempList0. If ref\_pic\_list\_modification\_flag\_l0 is 1, the process in is invoked with RefPicSetCurrTempList0 and num\_ref\_idx\_l0\_active\_minus1 as input, and RefPicList0 as output.

The following procedure is conducted to construct the ~~initial RefPicList1~~ RefPicSetCurrTempList1:

cIdx = 0  
if (ref\_pic\_list\_modification\_flag\_l1 == 0)

NumRpsCurrTempList1 = num\_ref\_idx\_l1\_active\_minus1+1

else {

NumRpsCurrTempList1 = NumRpsStCurr0 + NumRpsStCurr1 + NumRpsLtCurr

if (NumRpsCurrTempList1 <= num\_ref\_idx\_l1\_active\_minus1)

NumRpsCurrTempList1 = num\_ref\_idx\_l1\_active\_minus1+1

}

while( cIdx <~~= num\_ref\_idx\_l0\_active\_minus1~~ NumRpsCurrTempList1 )  
{  
 for( i=0; i < NumPocStCurr1 && cIdx <~~= num\_ref\_idx\_l1\_active\_minus1~~ NumRpsCurrTempList1; cIdx++, i++ )  
 ~~RefPicList1~~ RefPicSetCurrTempList1 [ cIdx ] = RefPicSetStCurr1[ i ]   
 for( i=0; i < NumPocStCurr0 && cIdx <~~= num\_ref\_idx\_l1\_active\_minus1~~ NumRpsCurrTempList1; cIdx++, i++ )   
 ~~RefPicList1~~ RefPicSetCurrTempList1 [ cIdx ] = RefPicSetStCurr0[ i ]  
 for( i=0; i < NumPocLtCurr && cIdx <~~= num\_ref\_idx\_l1\_active\_minus1~~ NumRpsCurrTempList1; cIdx++, i++ )  
 ~~RefPicList1~~ RefPicSetCurrTempList1 [ cIdx ] = RefPicSetLtCurr[ i ]  
}

If ref\_pic\_list\_modification\_flag\_l1 is 0, the initial RefPicList1 is equivalent to RefPicSetCurrTempList1. If ref\_pic\_list\_modification\_flag\_l1 is 1, the process in is invoked with RefPicSetCurrTempList1 and num\_ref\_idx\_l1\_active\_minus1 as input, and RefPicList1 as output.

#### ~~Modification process for reference picture lists~~

~~After the invocation of this process, there shall be no reference pictures with greater temporal\_id than the current slice included in the output RefPicList0 or RefPicList1.~~

~~When ref\_pic\_list\_modification\_flag\_l0 is equal to 1, the following applies:~~

1. ~~Let refIdxL0 be an index into the reference picture list RefPicList0. It is initially set equal to 0.~~
2. ~~The corresponding syntax elements modification\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.~~

~~– If modification\_of\_pic\_nums\_idc is equal to 0 or equal to 1, the process specified in subclause  is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.~~

~~– Otherwise, if modification\_of\_pic\_nums\_idc is equal to 2, the process specified in subclause  is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.~~

~~– Otherwise (modification\_of\_pic\_nums\_idc is equal to 3), the modification process for reference picture list RefPicList0 is finished.~~

~~When the current slice is a B slice and ref\_pic\_list\_modification\_flag\_l1 is equal to 1, the following applies:~~

1. ~~Let refIdxL1 be an index into the reference picture list RefPicList1. It is initially set equal to 0.~~
2. ~~The corresponding syntax elements modification\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.~~

~~– If modification\_of\_pic\_nums\_idc is equal to 0 or equal to 1, the process specified in subclause  is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.~~

~~– Otherwise, if modification\_of\_pic\_nums\_idc is equal to 2, the process specified in subclause  is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.~~

~~– Otherwise (modification\_of\_pic\_nums\_idc is equal to 3), the modification process for reference picture list RefPicList1 is finished.~~

##### ~~Modification process of reference picture lists for short-term reference pictures~~

~~Input to this process is an index refIdxLX (with X being 0 or 1).~~

~~Output of this process is an incremented index refIdxLX.~~

~~The variable picNumLXNoWrap is derived as follows.~~

~~If ref\_pic\_list\_modification\_idc is equal to 0, the following applies.~~

~~– If the current reference picture list is RefPicList0, curRefPicSet is set to RefPicSetStCurr0.  
– Otherwise (the current reference picture list is RefPicList1), curRefPicSet is set to RefPicSetStCurr1.~~

~~– Otherwise, if ref\_pic\_list\_modification\_idc is equal to 1, the following applies.~~

~~– If the current reference picture list is RefPicList0, curRefPicSet is set to RefPicSetStCurr1.  
– Otherwise (the current reference picture list is RefPicList1), curRefPicSet is set to RefPicSetStCurr0.~~

~~– Otherwise, if ref\_pic\_list\_modification\_idc is equal to 2, curRefPicSet is set to RefPicSetLtCurr.~~

~~The variable pocLX is derived as follows.~~

~~pocLX = curRefPicSet[ ref\_pic\_set\_idx ] (8‑9)~~

~~The following procedure is conducted to place the picture picR with PicOrderCnt( picR ) equal to pocLX into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX.~~

~~for( cIdx = num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx > refIdxLX; cIdx− − )  
 RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]  
RefPicListX[ refIdxLX++ ] = pocLX  
nIdx = refIdxLX (8‑9)  
for( cIdx = refIdxLX; cIdx <= num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx++ )   
 if( PicOrderCnt( RefPicListX[ cIdx ] ) != pocLX )  
 RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]~~

~~NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num\_ref\_idx\_lX\_active\_minus1 of the list need to be retained.~~

## Proposed modification process for reference picture lists

Input to this process is an array of reference picture RefPicSetCurrTempLX, and the size of the reference picture list num\_ref\_idx\_lX\_active\_minus1 (with X being 0 or 1).

Output of this process is an array containing the modified reference picture list RefPicListX.

Let refIdxLX be an index into the reference picture list RefPicListLX. It is initially set equal to 0.

The following process is repeated until refIdxLX is greater than num\_ref\_idx\_lX\_active\_minus1+1.

– RefPicListX [ refIdxLX++ ] = RefPicSetCurrTempLX [ ref\_pic\_set\_idx ]

# Encoder implementation

The proposed changes are implemented in HM5.0. In order to test the reference picture list modification process, an encoder-only change is additionally implemented in HM5.0. The encoder-only change modifies the list 0 and list 1 by maximizing the number of unique entries in L0 and L1 (in other words, minimizing the chance that the two lists contain the same entries). Entries in RefPicSetCurr0, RefPicSetCurr1, the initial L0 and L1, as well as modified L0 and L1 using this encoder-only change, are shown in Table 4 for the GOP structure (HM5.0 common testing conditions) in Figure 3.

1. **Reference picture reordering example**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **POC** | **RPS** | | **Initial lists** | | **Modified lists** | |
| **StCurr0** | **StCurr1** | **L0** | **L1** | **L0** | **L1** |
| 16 | {8, 6, 4, 0} | {NULL} | {8, 6, 4, 0} | {8, 6, 4, 0} | {8, 6, 4, 0} | {8, 6, 4, 0} |
| 12 | {8, 6} | {16} | {8, 6} | {16, 8} | {8, 6} | {16, 8} |
| 10 | {8, 6} | {12, 16} | {8, 6} | {12, 16} | {8, 6} | {12, 16} |
| 9 | {8} | {10, 12, 16} | {8, 10} | {10, 12} | {8, 12} | {10, 16} |
| 11 | {10, 8} | {12, 16} | {10, 8} | {12, 16} | {10, 8} | {12, 16} |
| 14 | {12, 10, 8} | {16} | {12, 10} | {16, 12} | {12, 10} | {16, 8} |
| 13 | {12, 8} | {14, 16} | {12, 8} | {14, 16} | {12, 8} | {14, 16} |
| 15 | {14, 12, 8} | {16} | {14, 12} | {16, 14} | {14, 12} | {16, 8} |



**Figure 3. GOP structure for Random Access**

# Simulation results

Simulations were carried out using with Random Access common testing conditions [2]. Table 5 shows the rate distortion performance using the reordered L0 and L1 in Table 4 for all 3 random access settings in [2]. No significant RD performance difference is observed. Detailed RD data can be found in the attached spreadsheet.

The reordered LC is signalled using two methods, the existing ref\_pic\_list\_modification() method in WD5 and the proposed method. Table 6 shows the number of bits used for each picture in the GOP in Figure 3. Table 7, and show the bit count for ref\_pic\_list\_modification() aggregated over a random access period (between two random access points) for each sequence with the common testing conditions RA-HE, RA-LC and RA-10, as defined in [2] and section 2.1 in [6].

An alternative coding method of ref\_pic\_set\_idx was suggested at the BoG meeting, which is to use u(v) instead of te(v). Table 10 shows the number of bits for each picture in the GOP in Figure 3. Table 11, Table 12, and Table 13 show the bit count for ref\_pic\_list\_modification() aggregated over a random access period for each sequence with the common testing conditions RA-HE, RA-LC and RA-10. As indicated by the bit counting data, u(v) represents a more efficient way of coding ref\_pic\_set\_idx. Therefore, the syntax in this document has been updated to use u(v) instead of te(v) in the syntax table in Table 3.

1. RD performance using the reordered L0/L1 vs using the default L0/L1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Random Access HE** | | | **Random Access LC** | | | **Random Access HE-10** | | |
|  | Y | U | V | Y | U | V | Y | U | V |
| Class A (8bit) | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 0.0% | -0.3% | -0.3% |
| Class B | -0.2% | -0.1% | -0.1% | -0.1% | 0.0% | -0.1% | -0.1% | -0.1% | -0.1% |
| Class C | -0.1% | 0.0% | 0.0% | 0.0% | -0.1% | 0.0% |  |  |  |
| Class D | -0.1% | -0.2% | -0.1% | -0.1% | 0.0% | -0.1% |  |  |  |
| Class E |  |  |  |  |  |  |  |  |  |
| **Overall** | -0.1% | -0.1% | -0.1% | -0.1% | 0.0% | 0.0% | -0.1% | -0.2% | -0.2% |
|  | -0.1% | -0.1% | -0.1% | -0.1% | 0.0% | 0.0% | -0.1% | -0.2% | -0.1% |
| Class F | -0.1% | -0.1% | 0.0% | -0.1% | -0.1% | -0.1% |  |  |  |

1. Bit counting (per picture) for ref\_pic\_list\_modification()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POC** | **Modified lists** | | **Existing ref\_pic\_list\_modification()** | **Proposed ref\_pic\_list\_modification()** |
| **L0** | **L1** |
| 16 | {8, 6, 4, 0} | {8, 6, 4, 0} | 2 | 2 |
| 12 | {8, 6} | {16, 8} | 2 | 2 |
| 10 | {8, 6} | {12, 16} | 2 | 2 |
| 9 | {8, 12} | {10, 16} | 26 | 10 |
| 11 | {10, 8} | {12, 16} | 2 | 2 |
| 14 | {12, 10} | {16, 8} | 15 | 8 |
| 13 | {12, 8} | {14, 16} | 2 | 2 |
| 15 | {14, 12} | {16, 8} | 15 | 8 |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-HE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 3 | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% |
| PeopleOnStreet | 275 | 146 | 46.91% |
| Nebuta |  |  |  |
| SteamLocomotive |  |  |  |
| Class B | Kimono | 209 | 110 | 47.37% |
| ParkScene | 209 | 110 | 47.37% |
| Cactus | 407 | 218 | 46.44% |
| BasketballDrive | 407 | 218 | 46.44% |
| BQTerrace | 539 | 290 | 46.20% |
| Class C | BasketballDrill | 407 | 218 | 46.44% |
| BQMall | 539 | 290 | 46.20% |
| PartyScene | 407 | 218 | 46.44% |
| RaceHorsesC | 275 | 146 | 46.91% |
| Class D | BasketballPass | 407 | 218 | 46.44% |
| BQSquare | 539 | 290 | 46.20% |
| BlowingBubbles | 407 | 218 | 46.44% |
| RaceHorses | 275 | 146 | 46.91% |
| Class E | Vidyo1 |  |  |  |
| Vidyo3 |  |  |  |
| Vidyo4 |  |  |  |
| Class F | BasketballDrillText | 407 | 218 | 46.44% |
| ChinaSpeed | 275 | 146 | 46.91% |
| SlideEditing | 275 | 146 | 46.91% |
| SlideShow | 143 | 74 | 48.25% |
| **Total bits per GOP** | | **6677** | **3566** | **46.74%** |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-LC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 2 | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% |
| PeopleOnStreet | 275 | 146 | 46.91% |
| Nebuta |  |  |  |
| SteamLocomotive |  |  |  |
| Class B | Kimono | 209 | 110 | 47.37% |
| ParkScene | 209 | 110 | 47.37% |
| Cactus | 407 | 218 | 46.44% |
| BasketballDrive | 407 | 218 | 46.44% |
| BQTerrace | 539 | 290 | 46.20% |
| Class C | BasketballDrill | 407 | 218 | 46.44% |
| BQMall | 539 | 290 | 46.20% |
| PartyScene | 407 | 218 | 46.44% |
| RaceHorses | 275 | 146 | 46.91% |
| Class D | BasketballPass | 407 | 218 | 46.44% |
| BQSquare | 539 | 290 | 46.20% |
| BlowingBubbles | 407 | 218 | 46.44% |
| RaceHorses | 275 | 146 | 46.91% |
| Class E | Vidyo1 |  |  |  |
| Vidyo3 |  |  |  |
| Vidyo4 |  |  |  |
| Class F | BasketballDrillText | 407 | 218 | 46.44% |
| ChinaSpeed | 275 | 146 | 46.91% |
| SlideEditing | 275 | 146 | 46.91% |
| SlideShow | 143 | 74 | 48.25% |
| **Total bits per GOP** | | **6677** | **3566** | **46.70%** |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 3 | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% |
| PeopleOnStreet | 275 | 146 | 46.91% |
| Nebuta | 539 | 290 | 46.20% |
| SteamLocomotive | 539 | 290 | 46.20% |
| Class B | Kimono | 209 | 110 | 47.37% |
| ParkScene | 209 | 110 | 47.37% |
| Cactus | 407 | 218 | 46.44% |
| BasketballDrive | 407 | 218 | 46.44% |
| BQTerrace | 539 | 290 | 46.20% |
| Class C | BasketballDrill |  |  |  |
| BQMall |  |  |  |
| PartyScene |  |  |  |
| RaceHorses |  |  |  |
| Class D | BasketballPass |  |  |  |
| BQSquare |  |  |  |
| BlowingBubbles |  |  |  |
| RaceHorses |  |  |  |
| Class E | Vidyo1 |  |  |  |
| Vidyo3 |  |  |  |
| Vidyo4 |  |  |  |
| Class F | BasketballDrillText |  |  |  |
| ChinaSpeed |  |  |  |
| SlideEditing |  |  |  |
| SlideShow |  |  |  |
| **Total bits per GOP** | | **3399** | **1818** | **46.51%** |

1. Bit counting (per picture) for ref\_pic\_list\_modification()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **POC** | **Modified lists** | | **Existing ref\_pic\_list\_modification()** | **Proposed ref\_pic\_list\_modification()** | **Proposed ref\_pic\_list\_modification() with u(v) for ref\_pic\_set\_idx** |
| **L0** | **L1** |
| 16 | {8, 6, 4, 0} | {8, 6, 4, 0} | 2 | 2 | 2 |
| 12 | {8, 6} | {16, 8} | 2 | 2 | 2 |
| 10 | {8, 6} | {12, 16} | 2 | 2 | 2 |
| 9 | {8, 12} | {10, 16} | 26 | 10 | 10 |
| 11 | {10, 8} | {12, 16} | 2 | 2 | 2 |
| 14 | {12, 10} | {16, 8} | 15 | 8 | 6 |
| 13 | {12, 8} | {14, 16} | 2 | 2 | 2 |
| 15 | {14, 12} | {16, 8} | 15 | 8 | 6 |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-HE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 3 | Bits saved (%) | GOP bit count with u(v) | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% | 130 | 52.73% |
| PeopleOnStreet | 275 | 146 | 46.91% | 130 | 52.73% |
| Nebuta |  |  |  |  |  |
| SteamLocomotive |  |  |  |  |  |
| Class B | Kimono | 209 | 110 | 47.37% | 98 | 53.11% |
| ParkScene | 209 | 110 | 47.37% | 98 | 53.11% |
| Cactus | 407 | 218 | 46.44% | 194 | 52.33% |
| BasketballDrive | 407 | 218 | 46.44% | 194 | 52.33% |
| BQTerrace | 539 | 290 | 46.20% | 258 | 52.13% |
| Class C | BasketballDrill | 407 | 218 | 46.44% | 194 | 52.33% |
| BQMall | 539 | 290 | 46.20% | 258 | 52.13% |
| PartyScene | 407 | 218 | 46.44% | 194 | 52.33% |
| RaceHorsesC | 275 | 146 | 46.91% | 130 | 52.73% |
| Class D | BasketballPass | 407 | 218 | 46.44% | 194 | 52.33% |
| BQSquare | 539 | 290 | 46.20% | 258 | 52.13% |
| BlowingBubbles | 407 | 218 | 46.44% | 194 | 52.33% |
| RaceHorses | 275 | 146 | 46.91% | 130 | 52.73% |
| Class E | Vidyo1 |  |  |  |  |  |
| Vidyo3 |  |  |  |  |  |
| Vidyo4 |  |  |  |  |  |
| Class F | BasketballDrillText | 407 | 218 | 46.44% | 194 | 52.33% |
| ChinaSpeed | 275 | 146 | 46.91% | 130 | 52.73% |
| SlideEditing | 275 | 146 | 46.91% | 130 | 52.73% |
| SlideShow | 143 | 74 | 48.25% | 66 | 53.85% |
| **Total bits per GOP** | | **6677** | **3566** | **46.7%** | **3958** | **52.46%** |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-LC

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 3 | Bits saved (%) | GOP bit count using Table 3 with u(v) | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% | 130 | 52.73% |
| PeopleOnStreet | 275 | 146 | 46.91% | 130 | 52.73% |
| Nebuta |  |  |  |  |  |
| SteamLocomotive |  |  |  |  |  |
| Class B | Kimono | 209 | 110 | 47.37% | 98 | 53.11% |
| ParkScene | 209 | 110 | 47.37% | 98 | 53.11% |
| Cactus | 407 | 218 | 46.44% | 194 | 52.33% |
| BasketballDrive | 407 | 218 | 46.44% | 194 | 52.33% |
| BQTerrace | 539 | 290 | 46.20% | 258 | 52.13% |
| Class C | BasketballDrill | 407 | 218 | 46.44% | 194 | 52.33% |
| BQMall | 539 | 290 | 46.20% | 258 | 52.13% |
| PartyScene | 407 | 218 | 46.44% | 194 | 52.33% |
| RaceHorses | 275 | 146 | 46.91% | 130 | 52.73% |
| Class D | BasketballPass | 407 | 218 | 46.44% | 194 | 52.33% |
| BQSquare | 539 | 290 | 46.20% | 258 | 52.13% |
| BlowingBubbles | 407 | 218 | 46.44% | 194 | 52.33% |
| RaceHorses | 275 | 146 | 46.91% | 130 | 52.73% |
| Class E | Vidyo1 |  |  |  |  |  |
| Vidyo3 |  |  |  |  |  |
| Vidyo4 |  |  |  |  |  |
| Class F | BasketballDrillText | 407 | 218 | 46.44% | 194 | 52.33% |
| ChinaSpeed | 275 | 146 | 46.91% | 130 | 52.73% |
| SlideEditing | 275 | 146 | 46.91% | 130 | 52.73% |
| SlideShow | 143 | 74 | 48.25% | 66 | 53.85% |
| **Total bits per GOP** | | **6677** | **3566** | **46.7%** | **3958** | **52.4%** |

1. Bit counting (per GOP) for ref\_pic\_list\_modification(), per sequence, RA-10

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Sequence | GOP bit count using Table 1 | GOP bit count using Table 3 | Bits saved (%) | GOP bit count using Table 3 with u(v) | Bits saved (%) |
| Class A | Traffic | 275 | 146 | 46.91% | 130 | 52.73% |
| PeopleOnStreet | 275 | 146 | 46.91% | 130 | 52.73% |
| Nebuta | 539 | 290 | 46.20% | 258 | 52.13% |
| SteamLocomotive | 539 | 290 | 46.20% | 258 | 52.13% |
| Class B | Kimono | 209 | 110 | 47.37% | 98 | 53.11% |
| ParkScene | 209 | 110 | 47.37% | 98 | 53.11% |
| Cactus | 407 | 218 | 46.44% | 194 | 52.33% |
| BasketballDrive | 407 | 218 | 46.44% | 194 | 52.33% |
| BQTerrace | 539 | 290 | 46.20% | 258 | 52.13% |
| Class C | BasketballDrill |  |  |  |  |  |
| BQMall |  |  |  |  |  |
| PartyScene |  |  |  |  |  |
| RaceHorses |  |  |  |  |  |
| Class D | BasketballPass |  |  |  |  |  |
| BQSquare |  |  |  |  |  |
| BlowingBubbles |  |  |  |  |  |
| RaceHorses |  |  |  |  |  |
| Class E | Vidyo1 |  |  |  |  |  |
| Vidyo3 |  |  |  |  |  |
| Vidyo4 |  |  |  |  |  |
| Class F | BasketballDrillText |  |  |  |  |  |
| ChinaSpeed |  |  |  |  |  |
| SlideEditing |  |  |  |  |  |
| SlideShow |  |  |  |  |  |
| **Total bits per GOP** | | **3399** | **1818** | **46.51%** | **1618** | **52.40%** |

# References

1. B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, T. Wiegand. WD5: Working Draft 5 of High-Efficiency Video Coding. Document no JCTVC-G1103, November 2011.
2. F. Bossen, Common HM test conditions and software reference configurations. Document no JCTVC-G1200, November 2011.
3. J. Boyce, R. Sjoberg, Y. K. Wang, BoG report: Reference picture buffering and list construction. Document no JCTVC-G1002, November 2011.
4. Y. Suzuki, et al, Extension of uni-prediction simplification in B slices. Document no JCTVC-D421, January 2011.
5. Y. He, Y, Ye, AhG21: Reference picture lists combination syntax. Document no JCTVC-H0137, February 2012.
6. Y.-K. Wang, M. M. Hannuksela, T. K. Tan, R. Sjöberg, BoG report: Common conditions for reference picture marking and list construction proposals. Document no JCTVC-G1036, November 2011.

# 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).**