|  |  |
| --- | --- |
| **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-I0344 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | On reference picture set definition and signaling | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Rajan L. Joshi Adarsh K. Ramasubramonian Ye-Kui Wang Ying Chen  5775 Morehouse Drive, San Diego, CA 92121-1714 USA | Tel: Email: | +1-858-658-4511 [rajanj@qualcomm.com](mailto:rajanj@qualcomm.com)  1-858-658-5804 [aramasub@qualcomm.com](mailto:aramasub@qualcomm.com)  1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com)  1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

In the HEVC committee draft a reference picture set may contain pictures with higher temporal\_id values than the current picture. This has the effect that if a bitstream corresponding to a lower temporal layer is extracted, the RPS of a picture in the extracted sub-bitstream may contain a picture belonging to a higher temporal layer thus not present in the sub-bitstream. A re-definition of the reference picture set (RPS) is proposed to exclude pictures that belong to a temporal layer higher than that of the current picture. In addition, a simplified method to signal the long-term and the short-term reference picture set is proposed. The method provides bit saving in most cases, particularly when all the pictures in the RPS are used for reference in the current picture.

# Introduction

Consider a temporally scalable bitstream corresponding to a GOP size of 8. Such a bitstream typically has 4 temporal layers. If only temporal layer with temporal\_id equal to 0 is extracted, RPS for a picture with temporal\_id 0 may contain pictures with higher temporal\_ids that are no longer a part of the extracted sub-bitstream. This situation is not ideal since there are pictures specified in the RPS of a picture in the extracted bitstream are not present. We propose a redefinition of RPS to exclude pictures with temporal\_id values higher than the current picture.

We also propose that used\_by\_curr\_\*\_flag used for reference marking be removed, and reference marking be instead provided by explicitly signalling the number of pictures that are used for reference by the current picture, and the number of pictures that are marked as not used for reference by the current picture.

# Proposal

## Re-definition of RPS

The definition of RPS is modified as follows:

**3.93 reference picture set**: A set of *reference pictures* associated with a *picture*, consisting of all *reference pictures* that are prior to the associated *picture* in decoding order, that may be used for *inter prediction* of the associated *picture* or any *picture* following the associated *picture* in *decoding order,* and that have temporal\_id less than or equal to that of the associated picture.

This has implications for the DPB buffer management in the sense that a picture that is not in the RPS of current picture and has a temporal\_id higher than the temporal\_id of the current picture should not be removed from DPB.

In section 8.3.2

1. All reference pictures in the decoded picture buffer that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetStFoll, and have temporal\_id less than or equal to the temporal\_id of the current picture, are marked as "unused for reference".

We provide an example of how modified RPS would work in practice. Consider the GOP structure used in common test conditions (encoder\_randomaccess\_main.cfg, GOP size 8). Table 2 shows the pictures present in the DPB and pictures which are output for the current definition of RPS. Table 3 shows the same information for the modified RPS. We have chosen to show the DPB state after decoding of first slice header in picture and after all the slices in the picture have been decoded. The reason for this is that after decoding the first slice header, the RPS for the current picture is known. Hence, pictures that are no longer needed for reference can be dropped from the DPB. The second column in Table 2 indicates whether the DPB status is after slice header decoding or picture data decoding.

Table 1: DPB buffer status for current definition of RPS and GOP size 8 (encoder\_randomaccess\_main.cfg)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Decoding order | Timing | RPS | DPB | Output |  |
| 0 | Slice header | - |  |  |  |
| 0 | Picture data |  | 0 |  |  |
| 8 | Slice header | 0 | 0 |  |  |
| 8 | Picture data |  | 0, 8 |  |  |
| 4 | Slice header | 0, 8 | 0, 8 |  |  |
| 4 | Picture data |  | 0, 4, 8 |  |  |
| 2 | Slice header | 0, 4, 8 | 0, 4, 8 |  |  |
| 2 | Picture data |  | 0, 2, 4, 8 | 0 |  |
| 1 | Slice header | 0, 2, 4, 8 | 0, 2, 4, 8 |  |  |
| 1 | Picture data |  | 0, 1, 2, 4, 8 | 1 |  |
| 3 | Slice header | 0, 2, 4, 8 | 0, 2, 4, 8 |  |  |
| 3 | Picture data |  | 0, 2, 3, 4, 8 | 2 |  |
| 6 | Slice header | 0, 2, 4, 8 | 0, 2, 3, 4, 8 |  |  |
| 6 | Picture data |  | 0, 2, 3, 4, 6, 8 | 3 |  |
| 5 | Slice header | 0, 4, 6, 8 | 0, 4, 6, 8 |  |  |
| 5 | Picture data |  | 0, 4, 5, 6, 8 | 4 |  |
| 7 | Slice header | 0, 4, 6, 8 | 0, 4, 5, 6, 8 |  |  |
| 7 | Picture data |  | 0, 4, 5, 6, 7, 8 | 5 |  |
| 16 | Slice header | 0, 4, 6, 8 | 0, 4, 6, 7, 8 |  |  |
| 16 | Picture data |  | 0, 4, 6, 7, 8, 16 | 6 |  |
| 12 | Slice header | 6, 8, 16 | 6, 7, 8, 16 |  |  |
| 12 | Picture data |  | 6, 7, 8, 12, 16 | 7 |  |
| 10 | Slice header | 6, 8, 12, 16 | 6, 8, 12, 16 |  |  |
| 10 | Picture data |  | 6, 8, 10, 12, 16 | 8 |  |
| 9 | Slice header | 8, 10, 12, 16 | 8, 10, 12, 16 |  |  |
| 9 | Picture data |  | 8, 9, 10, 12, 16 | 9 |  |
| 11 | Slice header | 8, 10, 12, 16 | 8, 10, 12, 16 |  |  |
| 11 | Picture data |  | 8, 10, 11, 12, 16 | 10 |  |
| 14 | Slice header | 8, 10, 12, 16 | 8, 10, 11, 12, 16 |  |  |
| 14 | Picture data |  | 8, 10, 11, 12, 14, 16 | 11 |  |
| 13 | Slice header | 8, 12, 14, 16 | 8, 12, 14, 16 |  |  |
| 13 | Picture data |  | 8, 12, 13, 14, 16 | 12 |  |
| 15 | Slice header | 8, 12, 14, 16 | 8, 12, 13, 14, 16 |  |  |
| 15 | Picture data |  | 8, 12, 13, 14, 15, 16 | 13 |  |

Table 2: DPB buffer status for proposed definition of RPS and GOP size 8 (encoder\_randomaccess\_main.cfg)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Decoding order | Timing | RPS | DPB | Output |  |
| 16 | Slice header | 0, 8 | 0, 4, 6, 7, 8 |  |  |
| 16 | Picture data |  | 0, 4, 6, 7, 8, 16 | 6 |  |
| 12 | Slice header | 8, 16 | 6, 7, 8, 16 |  |  |
| 12 | Picture data |  | 6, 7, 8, 12, 16 | 7 |  |
| 10 | Slice header | 6, 8, 12, 16 | 6, 8, 12, 16 |  |  |
| 10 | Picture data |  | 6, 8, 10, 12, 16 | 8 |  |
| 9 | Slice header | 8, 10, 12, 16 | 6, 8, 10, 12, 16 |  |  |
| 9 | Picture data |  | 8, 9, 10, 12, 16 | 9 |  |
| 11 | Slice header | 8, 10, 12, 16 | 8, 10, 12, 16 |  |  |
| 11 | Picture data |  | 8, 10, 11, 12, 16 | 10 |  |
| 14 | Slice header | 8, 10, 12, 16 | 8, 10, 11, 12, 16 |  |  |
| 14 | Picture data |  | 8, 10, 11, 12, 14, 16 | 11 |  |
| 13 | Slice header | 8, 12, 14, 16 | 8, 12, 14, 16 |  |  |
| 13 | Picture data |  | 8, 12, 13, 14, 16 | 12 |  |
| 15 | Slice header | 8, 12, 14, 16 | 8, 12, 13, 14, 16 |  |  |
| 15 | Picture data |  | 8, 12, 13, 14, 15, 16 | 13 |  |

## Explicit signaling of picture used for reference and used for future reference

### Slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| **…** | u(1) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| **num\_long\_term\_curr** | ue(v) |
| **num\_long\_term\_foll** | ue(v) |
| NumLongTerm = num\_long\_term\_curr + num\_long\_term\_foll |  |
| for( i = 0; i < NumLongTerm; i++ ) { |  |
| **delta\_poc\_lsb\_lt**[ i ] | ue(v) |
| **delta\_poc\_msb\_present\_flag**[ i ] | u(1) |
| if( delta\_poc\_msb\_present\_flag[ i ] ) |  |
| **delta\_poc\_msb\_cycle\_lt\_minus1**[ i ] | ue(v) |
| } |  |
| } |  |
| … |  |
| } |  |

### Slice header semantics

**num\_long\_term\_curr** specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture and are used by reference by the current picture . The value of num\_long\_term\_curr shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_curr is inferred to be equal to 0.

**num\_long\_term\_foll** specifies the number of the long-term reference pictures that are to be included in the long-term reference picture set of the current picture and are not used by reference by the current picture . The value of num\_long\_term\_foll shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – NumNegativePics[ StRpsIdx ] – NumPositivePics[ StRpsIdx ] , inclusive. When not present, the value of num\_long\_term\_foll is inferred to be equal to 0.

### Short-term reference picture set syntax

|  |  |
| --- | --- |
| short\_term\_ref\_pic\_set( idx ) { | Descriptor |
| **inter\_ref\_pic\_set\_prediction\_flag** | u(1) |
| if( inter\_ref\_pic\_set\_prediction\_flag) { |  |
| **delta\_idx\_minus1** | ue(v) |
| **delta\_rps\_sign** | u(1) |
| **abs\_delta\_rps\_minus1** | ue(v) |
| } |  |
| else { |  |
| **num\_short\_term\_curr0** | ue(v) |
| **num\_short\_term\_curr1** | ue(v) |
| **num\_short\_term\_foll0** | ue(v) |
| **num\_short\_term\_foll1** | ue(v) |
| NumShortTerm = num\_short\_term\_curr0 + num\_short\_term\_curr1 +  num\_short\_term\_foll0 + num\_short\_term\_foll1 |  |
| for( i = 0; i < NumShortTerm; i++ ) |  |
| **short\_term\_delta\_poc\_minus1**[ i ] | ue(v) |
| } |  |
| } |  |

### Short term reference picture set semantics

**num\_short\_term\_curr0** specifies the number of the short-term reference pictures that are used as reference by the current picture and that have a POC value less than the POC value of the current picture. The value of num\_short\_term\_curr0 shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ], inclusive.

**num\_short\_term\_curr1** specifies the number of the short-term reference pictures that are used as reference by the current picture and that have a POC value more than the POC value of the current picture. The value of num\_short\_term\_curr1 shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – num\_short\_term\_curr0, inclusive.

**num\_short\_term\_foll0** specifies the number of the short-term reference pictures that are not used as reference by the current picture and that have a POC value less than the POC value of the current picture. The value of num\_short\_term\_foll0 shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – num\_short\_term\_curr0 – num\_short\_term\_curr1, inclusive.

**num\_short\_term\_foll1** specifies the number of the short-term reference pictures that are not used as reference by the current picture and that have a POC value more than the POC value of the current picture. The value of num\_short\_term\_foll1 shall be in the range of 0 to max\_dec\_pic\_buffering[ max\_temporal\_layers\_minus1 ] – num\_short\_term\_curr0 – num\_short\_term\_curr1 – num\_short\_term\_foll0, inclusive.

The variables NumNegativePics[ idx ] and NumPositivePics[ idx ] are derived as follows.

NumNegativePics[ idx ] = num\_short\_term\_curr0 + num\_short\_term\_foll0  
NumPositivePics[ idx ] = num\_short\_term\_curr1 + num\_short\_term\_foll1

The variables UsedByCurrPicS0[ idx ][ i ] and UsedByCurrPicS1[ idx ][ i ] are derived as follows.

NumPositivePics[ idx ] = num\_short\_term\_curr1 + num\_short\_term\_foll1

**short\_term\_delta\_poc\_minus1**[ i ]specifies an absolute difference between two picture order count values. The value of short\_term\_delta\_poc\_minus1[ i ] shall be in the range of 0 to 215 – 1, inclusive.

The arrays DeltaPocLtS0[ idx ][ ], DeltaPocLtS1[ idx ][ ], UsedByCurrPicS0[ idx ][ ], and UsedByCurrPicS1[ idx ][ ] are derived as follows.

for(j = 0, i0 = 0; j < num\_short\_term\_curr0; j++, i0++) {  
 if(i0 = = 0)  
 DeltaPocS0Curr[i0] = - ( short\_term\_delta\_poc\_minus1[ j ] + 1 )  
 else  
 DeltaPocS0Curr[i0] = DeltaPocS0Curr[i0] - ( short\_term\_delta\_poc\_minus1[ j ] + 1 )  
 }

for( j = num\_short\_term\_curr0, i0 = 0; j < num\_short\_term\_curr0 + num\_shor\_term\_foll0; j++, i0++) {  
 if(i0 = = 0)  
 DeltaPocS0Foll[i0] = - ( short\_term\_delta\_poc\_minus1[ j ] + 1 )  
 else  
 DeltaPocS0Foll[i0] = DeltaPocS0Curr[i0] - ( short\_term\_delta\_poc\_minus1[ j ] + 1 )

}

for( j = 0, i0 = 0, i1 = 0; j < num\_short\_term\_curr0 + num\_shor\_term\_foll0; j++ ) {

if( DeltaPocS0Curr[ i0 ] < DeltaPocS0Curr[ i1 ] ) {

DeltaPocS0[ idx ][ j ] = DeltaPocS0Curr[ i0 ]

UsedByCurrPicS0[ idx ][ j ] = 1

i0++

}  
 else {

DeltaPocS0[ idx ][ [ j ] = DeltaPocS0Foll[ i1 ]

UsedByCurrPicS0[ idx ][ [ j ] = 0

i1++

}

}

for(j = 0, i0 = 0; j < num\_short\_term\_curr1; j++, i0++) {  
 if(i0 == 0)  
 DeltaPocS1Curr[i0] = + ( short\_term\_delta\_poc\_minus1[ j ] + 1 )  
 else  
 DeltaPocS1Curr[i0] = DeltaPocS1Curr[i0] + ( short\_term\_delta\_poc\_minus1[ j ] + 1 )

}

for( j = num\_short\_term\_curr1, i0 = 0; j < num\_short\_term\_curr1 + num\_shor\_term\_foll1; j++, i0++) {  
 if(i0 == 0)  
 DeltaPocS1Foll[i0] = + ( short\_term\_delta\_poc\_minus1[ j ] + 1 )  
 else  
 DeltaPocS1Foll[i0] = DeltaPocS1Curr[i0] + ( short\_term\_delta\_poc\_minus1[ j ] + 1 )

}

for( j = 0, i0 = 0, i1 = 0; j < num\_short\_term\_curr1 + num\_shor\_term\_foll1; j++ ) {

if( DeltaPocS1Curr[ i0 ] < DeltaPocS1Curr[ i1 ] ) {

DeltaPocS1[ idx ][ [ j ] = DeltaPocS1Curr[ i0 ]

UsedByCurrPicS1[ idx ][ [ j ] = 1

i0++

}  
 else {

DeltaPocS1[ idx ][ [ j ] = DeltaPocS1Foll[ i1 ]

UsedByCurrPicS1[ idx ][ [ j ] = 0

i1++

}

}

When inter\_ref\_pic\_set\_prediction\_flagis equal to 1, the variables DeltaPocS0[ idx ][ i0 ], UsedByCurrPicS0[ idx ][ i0 ], DeltaPocS1[ idx ][ i1 ], UsedByCurrPicS1[ idx ][ i1], NumNegativePics[ idx ] and NumPositivePics[ idx ] are derived as follows.

for( j = 0, i0 = 0, i1 = 0; j < NumDeltaPocs[ RIdx ]; j++ )  
 DPoc = DeltaPoc[ RIdx ][ j ] + DeltaRPS  
 if( DPoc < 0 ) {  
 DeltaPocS0[ idx ][ i0 ] = DPoc  
 UsedByCurrPicS0[ idx ][ i0 ] = UsedByCurrPicS0[ RIdx ][ j ]   
 i0++  
 } else {  
 DeltaPocS1[ idx ][ i1 ] = DPoc  
 UsedByCurrPicS1[ idx ][ i1 ] = UsedByCurrPicS1[ RIdx ][ j ]   
 i1++  
 }

if( DeltaRPS < 0 ) {

DeltaPocS0[ idx ][ i0 ] = DeltaRPS

UsedByCurrPicS0[ idx ][ i0 ] = 1

} else {

DeltaPocS1[ idx ][ i0 ] = DeltaRPS

UsedByCurrPicS1[ idx ][ i0 ] = 1

}  
 NumNegativePics[ idx ] = i0  
 NumPositivePics[ idx ] = i1

# 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).**