|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  9th Meeting: Geneva, CH, 27 April – 7 May 2012 | Document: JCTVC-I0347r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Simplification of inter-RPS prediction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Adarsh K. Ramasubramonian Ying Chen Ye-Kui Wang  5775 Morehouse Dr San Diego, CA 92121 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 the current working draft of HEVC, inter prediction between RPS candidates in SPS is enabled. A simplification of the current syntax design for inter-RPS prediction is proposed in this document.

Revision 1 of this document includes software, simulation results and a proposal of a further simplied syntax design for inter-RPS prediction.

The results show an overall average bit-count increases of 22% of the related syntax elements for the simplified synbtax, and 34% for the further simplified method. These percentages roughly correspond to increase of 31 bits (about 4 bytes) and 47 bits (about 6 bytes) for the SPS.

More detailed results are included in the attached file JCTVC-I0347.xlsx, and the software package is included in the attached file JCTVC-I0347\_sw.zip.

It is proposed to adopt the further simplied syntax for simplicity.

# Introduction

In the current HEVC WD, prediction among the short-term RPS candidates inside an SPS is enabled by the highlighted syntax design as shown in the short-term reference picture set syntax table copied below.

We received complaints from implementers about the complexity of the syntax and associated decoding process.

In some cases, when the RPS of a picture A is used to predict the RPS of a picture B, most likely the picture A will be added into the RPS of the picture B.

In most cases, used\_by\_curr\_pic\_flag can be predicted from the flags associated with the reference RPS. It is less efficient and not clean to still signal this flag for inter-RPS prediction.

## Current Inter-RPS prediciton 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) |
| for( j = 0; j <= NumDeltaPocs[ RIdx ]; j++ ) { |  |
| **used\_by\_curr\_pic\_flag**[ j ] | u(1) |
| if( !**used\_by\_curr\_pic\_flag**[ j ]) |  |
| **use\_delta\_flag**[ j ] | u(1) |
| } |  |
| } |  |
| else { |  |
| **num\_negative\_pics** | ue(v) |
| **num\_positive\_pics** | ue(v) |
| for( i = 0; i < num\_negative\_pics; i++ ) { |  |
| **delta\_poc\_s0\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s0\_flag**[ i ] | u(1) |
| } |  |
| for( i = 0; i < num\_positive\_pics; i++ ) { |  |
| **delta\_poc\_s1\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s1\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |

## Current Inter-RPS prediciton semantics

**inter\_ref\_pic\_set\_prediction\_flag** equal to 1 specifies that the reference picture set of the current picture shall be predicted using another reference picture set in the active sequence parameter set.

**delta\_idx\_minus1** plus 1 specifies the difference between the index of the reference picture set of the current picture and the index of the reference picture set used for inter reference picture set prediction.

The variable RIdx shall be derived as follows.

RIdx = idx − (delta\_idx\_minus1+1) (7‑45)

**delta\_rps\_sign** specifies the sign of DiffPicOrderCnt( picA, picB ), where picA is associated with DeltaPoc[ RIdx ] and picB is associated with DeltaPoc[ idx ]. A value of 0 indicates that the sign is positive and a value of 1 indicates that the sign is negative.

**abs\_delta\_rps\_minus1** plus 1specifies the absolute value of DiffPicOrderCnt( picA, picB ), where picA is associated with DeltaPoc[ RIdx ] and picB is associated with DeltaPoc[ idx ].

The variable DeltaRPS is derived as follows.

DeltaRPS = (1 – (delta\_rps\_sign << 1)) \* (abs\_delta\_rps\_minus1 + 1) (7‑46)

**used\_by\_curr\_pic\_flag**[ j ] indicates whether a picture is used for reference by the current picture.

**use\_delta\_flag[ j ]** specifies whether there is a corresponding i-th reference picture for determining the value of DeltaPoc[ Rdx ][ j ]. When use\_delta\_flag[ j ] is not present, it is inferred to be equal to 1.

Table 7‑9 – used\_by\_curr\_pic\_flag[ j ] and use\_delta\_flag[ j ] operations for the reference picture set

|  |  |  |
| --- | --- | --- |
| used\_by\_curr\_pic\_flag[ j ] | use\_delta\_flag[ j ] | **Properties of i-th reference picture** |
| 1 | 1 | Picture is used for reference by the current picture DeltaPoc[ idx ][ i ] = DeltaPoc[ RIdx ][ j ] + DeltaRPS |
| 0 | 1 | Picture is not used for reference by the current picture (but used by future pictures) DeltaPoc[ idx ][ i ] = DeltaPoc[ RIdx ][ j ] + DeltaRPS |
| 0 | 0 | There is no corresponding i-th reference picture for determining the value of DeltaPoc[ RIdx ][ j ]. |

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++ )  
 if( used\_by\_curr\_pic\_flag[ j ] | | use\_delta\_flag[ j ] ) {  
 DPoc = DeltaPoc[ RIdx ][ j ] + DeltaRPS  
 if( DPoc < 0 ) {  
 DeltaPocS0[ idx ][ i0 ] = DPoc (7‑47)  
 UsedByCurrPicS0[ idx ][ i0 ] = used\_by\_curr\_pic\_flag[ j ] (7‑48)  
 i0++  
 } else {  
 DeltaPocS1[ idx ][ i1 ] = DPoc (7‑49)  
 UsedByCurrPicS1[ idx ][ i1 ] = used\_by\_curr\_pic\_flag[ j ] (7‑50)  
 i1++  
 }  
 }  
NumNegativePics[ idx ] = i0 (7‑51)  
NumPositivePics[ idx ] = i1 (7‑52)

When DeltaPoc[ RIdx ][ j ] is unavailable, it is set to 0.

The elements of the lists DeltaPocS0[ idx ] and UsedByCurrPicS0[ idx ] are sorted together by the decoder, such that DeltaPocS0[ idx ][ i0 ] > DeltaPocS0[ idx ][ i0 + 1 ] for i0 in the range of 0 to NumNegativePics[ idx ] − 2.

The elements of the lists DeltaPocS1[ idx ] and UsedByCurrPicS1[ idx ] are sorted together by the decoder, such that DeltaPocS1[ idx ][ i1 ] < DeltaPocS1[ idx ][ i1 + 1 ] for i1 in the range of 0 to NumPositivePics[ idx ] − 2.

# Proposal

In this document, the following modifications are proposed:

* delta\_rps\_sign and abs\_delta\_rps\_minus1 are merged into one syntax element;
* used\_by\_curr\_pic\_flag for each reference picture in the inter-predicted RPS is not signalled and always derived to be the same value as that flag of the corresponding picture in the reference RPS;
* An inter-predicted RPS can include not only the pictures in a reference RPS, but also the current picture of a reference RPS.

## The simplied 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)~~ |
| **delta\_rps\_minus1** | se(v) |
| ~~for( j = 0; j <= NumDeltaPocs[ RIdx ]; j++ ) {~~ |  |
| **~~used\_by\_curr\_pic\_flag~~**~~[ j ]~~ | ~~u(1)~~ |
| ~~if( !~~**~~used\_by\_curr\_pic\_flag~~**~~[ j ]~~~~)~~ |  |
| **~~use\_delta\_flag~~**~~[ j ]~~ | ~~u(1)~~ |
| ~~}~~ |  |
| **add\_predictor\_rps\_pic\_flag** | u(1) |
| **num\_ref\_pics\_drops\_from\_begin** | ue(v) |
| **num\_ref\_pics\_drops\_from\_end** | ue(v) |
| } |  |
| else { |  |
| **num\_negative\_pics** | ue(v) |
| **num\_positive\_pics** | ue(v) |
| for( i = 0; i < num\_negative\_pics; i++ ) { |  |
| **delta\_poc\_s0\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s0\_flag**[ i ] | u(1) |
| } |  |
| for( i = 0; i < num\_positive\_pics; i++ ) { |  |
| **delta\_poc\_s1\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s1\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |

## Semantics of the simplied syntax

**delta\_rps\_minus1** specifies the POC difference between picA and picB, where picA is associated with DeltaPoc[ RIdx ] and picB is associated with DeltaPoc[ idx ]. Note picA is called the current picture of DeltaPoc[ RIdx ] and picB is called the current picture of DeltaPoc[ idx ].

The variable DeltaRPS is derived as follows.

DeltaRPS = sign (delta\_rps\_minus1) ? (delta\_rps\_minus1 + 1) : (delta\_rps\_minus1 - 1)

**add\_predictor\_rps\_pic\_flag** equal to 1 indicates that the previous picture in decoding order using the reference RPS (with index equal to RIdx = idx - delta\_idx\_minus1-1) is included in the RPS with index equal to idx. add\_predictor\_rps\_pic\_flag equal to 0 indicates the previous picture in decoding order of the reference RPS (with index equal to RIdx ) is not included in the RPS with index equal to idx.

**num\_ref\_pics\_drops\_from\_begin** specifies the number of reference pictures starting from the beginning of the reference RPS (with index equal to RIdx) that are not used to predict the current RPS (with index equal to idx). Let RNPP and RNNP be num\_positive\_pics and num\_negative\_pics of the reference RPS, respectively. The value of num\_ref\_pics\_drops\_from\_begin shall be in the range of 0 to RNPP + RNNP - 1, inclusive.

**num\_ref\_pics\_drops\_from\_end** specifies the number of reference pictures starting from the end of the reference RPS (with index equal to RIdx) that are not used to predict the current RPS (with index equal to idx). num\_ref\_pics\_drops\_from\_end is in the range of 0 to RNPP + RNNP -1- num\_ref\_pics\_drops\_from\_begin, inclusive.

Note that it is assumed that the reference RPS (short-term) has been sorted in a way that all the entries are ordered in the ascending order of POC values.

When inter\_ref\_pic\_set\_prediction\_flag is 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 = num\_ref\_pics\_drops\_from\_begin, i0 = 0, i1 = 0, insertFlag = add\_predictor\_rps\_pic\_flag; j <= NumDeltaPocs[ RIdx ]- num\_ref\_pics\_drops\_from\_end; j++ )  
{  
 if ((DeltaPocSorted[ RIdx ][ j ]>DeltaRPS | |   
 j= = NumDeltaPocs[ RIdx ]- num\_ref\_pics\_drops\_from\_end) && insert\_flag) {  
 DPoc = DeltaRPS  
 currFlag = 1   
 insertFlag = 0  
 if (j< NumDeltaPocs[RIdx]) j--   
 }  
 else {  
 DPoc = DeltaPocSorted[ RIdx ][ j ] + DeltaRPS  
 currFlag = UsedByCurrPic[ Ridx ][ j ]  
 }  
 if( DPoc < 0 ) {  
 DeltaPocS0[ idx ][ i0 ] = DPoc  
 UsedByCurrPicS0[ idx ][ i0 ] = currFlag ~~used\_by\_curr\_pic\_flag[ j ]~~   
 i0++  
 } else {  
 DeltaPocS1[ idx ][ i1 ] = DPoc  
 UsedByCurrPicS1[ idx ][ i1 ] = currFlag ~~used\_by\_curr\_pic\_flag[ j ]~~   
 i1++  
 }   
}  
NumNegativePics[ idx ] = i0

NumPositivePics[ idx ] = i1

Regardless the value of inter\_ref\_pic\_set\_prediction\_flag, the following process is used to derive a sorted list of delta POC values and the associated used\_by\_curr\_pic\_sx\_flag values of the entries in the RPS. The arrays DeltaPocSorted[ idx ] and UsedByCurrPic[ idx ][ ] are derived as follows.

for(j = 0, i = NumNegativePics[ idx ] - 1; i >= 0; i--, j++) {

DeltaPocSorted[ idx ][ j ] = DeltaPocS0[ idx ][ i ]  
UsedByCurrPic[ idx ][ j ] = UsedByCurrPicS0[ idx ][ i ]

}

for(j = NumNegativePics[ idx ], i = NumNegativePics[ idx ] ; i < NumPositivePics[ idx ] + NumNegativePics[ idx ]; i++, j++) {

DeltaPocSorted[ idx ][ j ] = DeltaPocS1[ idx ][ i ]  
UsedByCurrPic[ idx ][ j ] = UsedByCurrPicS1[ idx ][ i ]

}

## The further simplied 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)~~ |
| **delta\_rps\_plus1** | se(v) |
| ~~for( j = 0; j <= NumDeltaPocs[ RIdx ]; j++ ) {~~ |  |
| **~~used\_by\_curr\_pic\_flag~~**~~[ j ]~~ | ~~u(1)~~ |
| ~~if( !~~**~~used\_by\_curr\_pic\_flag~~**~~[ j ]~~~~)~~ |  |
| **~~use\_delta\_flag~~**~~[ j ]~~ | ~~u(1)~~ |
| ~~}~~ |  |
| **add\_predictor\_rps\_pic\_flag** | u(1) |
| } |  |
| else { |  |
| **num\_negative\_pics** | ue(v) |
| **num\_positive\_pics** | ue(v) |
| for( i = 0; i < num\_negative\_pics; i++ ) { |  |
| **delta\_poc\_s0\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s0\_flag**[ i ] | u(1) |
| } |  |
| for( i = 0; i < num\_positive\_pics; i++ ) { |  |
| **delta\_poc\_s1\_minus1**[ i ] | ue(v) |
| **used\_by\_curr\_pic\_s1\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |

## Semantics for the furhter simplified syntax

**delta\_rps\_plus1** minus 1 specifies the POC difference between picA and picB, where picA is associated with DeltaPoc[ RIdx ] and picB is associated with DeltaPoc[ idx ]. Note picA is called the current picture of DeltaPoc[ RIdx ] and picB is called the current picture of DeltaPoc[ idx ].

The variable DeltaRPS is derived as follows.

DeltaRPS = sign (delta\_rps\_plus1) ? (delta\_rps\_plus1 + 1) : (delta\_rps\_plus1 - 1)

**add\_predictor\_rps\_pic\_flag** equal to 1 indicates that the previous picture in decoding order using the reference RPS (with index equal to RIdx = idx - delta\_idx\_minus1-1) is included in the RPS with index equal to idx. add\_predictor\_rps\_pic\_flag equal to 0 indicates the previous picture in decoding order of the reference RPS (with index equal to RIdx ) is not included in the RPS with index equal to idx.

When inter\_ref\_pic\_set\_prediction\_flag is 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, insertFlag = add\_predictor\_rps\_pic\_flag; j <= NumDeltaPocs[ RIdx ]; j++ )  
{

if( j < NumNegativePics[ RIdx ] ) {

k = NumNegativePics[ RIdx ] – 1 – j

currFlag = UsedByCurrPicS0[ RIdx ][ k ]

}

else {

k = j

currFlag = UsedByCurrPicS1[ RIdx ][ k – NumNegativePics[ RIdx ] ]

}  
 if((DeltaPoc [ RIdx ][ k ]>DeltaRPS | |   
 j = = NumDeltaPocs[ RIdx ]) && insert\_flag) {  
 DPoc = DeltaRPS  
 currFlag = 1  
 insertFlag = 0  
 if ( j < NumDeltaPocs[ RIdx ]) j--   
 }  
 else   
 DPoc = DeltaPoc [ RIdx ][  k ] + DeltaRPS  
 if( DPoc < 0 ) {  
 DeltaPocS0[ idx ][ i0 ] = DPoc  
 UsedByCurrPicS0[ idx ][  i0 ] = currFlag   
 i0++

} else {  
 DeltaPocS1[ idx ][ i1 ] = DPoc  
 UsedByCurrPicS1[ idx ][  i1 ] = currFlag  
 i1++  
 }   
}  
NumNegativePics[ idx ] = i0

NumPositivePics[ idx ] = i1

# Results

The performances of both simplified inter-RPS prediction syntaxes were compared with the algorithm in WD 6 of HEVC. The bit count results provided in Table 1 present the number of bits used to signal the short-term RPSs in the SPS for the random-access and the low-delay cases; these results includes those sets that are signalled explicitly in the SPS and those that are predicted.

The results show an overall average bit-count increases of 22% of the related syntax elements for the simplified synbtax, and 34% for the further simplified method. These percentages roughly correspond to increase of 31 bits (about 4 bytes) and 47 bits (about 6 bytes) for the SPS.

Table 1: Bit-count comparison of short-term RPS signalled in the SPS for the CD and the proposed method

|  |  |  |  |
| --- | --- | --- | --- |
|  | WD 6 | Simplied | Further simplied |
| Random-access |  |  |  |
| Total ST RPS bits in SPS | 142 | 141 | 155 |
| % increase |  | -0.70% | 9.15 % |
|  |  |  |  |
| Low-delay |  |  |  |
| Total ST RPS bits in SPS | 135 | 195 | 214 |
| % increase |  | 44.44% | 58.52% |

More detailed results are included in the attached file JCTVC-I0347.xlsx.

# 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).**