|  |  |
| --- | --- |
| **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-H0501r1 |

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

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

# Abstract

In this proposal, an adaptive POC signaling is proposed. Unlike the current HEVC design which signals for each picture a fixed number of Least Significant Bits (LSB) of the POC, with this proposal, pictures having different temporal\_id values can be signalled with different number of bits for the POC LSB.

# Problem

In the current HEVC design, POC values of pictures are signalled using POC LSB. The length of the POC LSB is fixed for the whole coded video sequence.

When the length of the POC LSB is not large, there is a problem on temporal scalability.

If the length of the least significant bits (LSBs) of POC is e.g. 4 bits, and if the bitstream is temporally scalable, then in an extracted bitstream subset containing one or more of the lower temporal layers, the gap of POC values may result in incorrect POC values of pictures.

Assume a temporal scalable bitstream has at least two temporal layers, with the lowest temporal layer having temporal\_id equal to 0, and other temporal layers having temporal\_id greater than 0. Pictures of the lowest temporal layer have POC values of 10\*n, i.e., 0, 10, 20, 30, 40, …, and so on, and only 4 bits are used to signal the POC LSB, hence the LSB values are in the range of 0 to 15, inclusive. Thus, when the picture with POC equal to 20, for which the POC LSB value is 4, is lost, the POC value of the picture with POC equal to 30, for which the POC LSB value is 14, would be derived as equal to 14, and consequently for the following pictures the derived POC values would also be incorrect.

To solve the above problem, the encoder can choose to increase the number of bits used for the POC LSB signaling, thus more bits are to be used for every slice.

# Proposal

It is proposed pictures with a lower temporal\_id may be signalled with a larger length POC LSB value.

## Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **profile\_idc** | u(8) |
| **reserved\_zero\_8bits** /\* equal to 0 \***/** | u(8) |
| **level\_idc** | u(8) |
| **seq\_parameter\_set\_id** | ue(v) |
| **max\_temporal\_layers\_minus1** | u(3) |
| **pic\_width\_in\_luma\_samples** | u(16) |
| **pic\_height\_in\_luma\_samples** | u(16) |
| **bit\_depth\_luma\_minus8** | ue(v) |
| **bit\_depth\_chroma\_minus8** | ue(v) |
| **pcm\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_bit\_depth\_chroma\_minus1** | u(4) |
| **log2\_max\_frame\_num\_minus4** | ue(v) |
| **min\_poc\_lsb\_len\_minus4** | ue(v) |
| **num\_add\_poc\_lsb\_lens** | ue(v) |
| for ( i=0; i < num\_add\_poc\_lsb\_lens; i++ ) { |  |
| **max\_temporal\_id\_poc\_len**[ i ] | u(3) |
| **poc\_len\_delta\_minus1**[ i ] | ue(v) |
| } |  |
| **max\_num\_ref\_frames** | ue(v) |
| **gaps\_in\_frame\_num\_value\_allowed\_flag** | u(1) |
| **log2\_min\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| **log2\_min\_pcm\_coding\_block\_size\_minus3** | ue(v) |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **chroma\_pred\_from\_luma\_enabled\_flag** | u(1) |
| **loop\_filter\_across\_slice\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **cu\_qp\_delta\_enabled\_flag** | u(1) |
| **temporal\_id\_nesting\_flag** | u(1) |
| **inter\_4x4\_enabled\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

## Sequence parameter set RBSP Semantics

**num\_add\_poc\_lsb\_lens** specifies the number of the following syntax element pairs max\_temporal\_id\_poc\_len[ i ] and poc\_len\_delta\_minus1[ i ] present in the sequence parameter set. The value of num\_add\_poc\_lsb\_lens shall be in the range of 0 to max\_temporal\_layers\_minus1.

**max\_temporal\_id\_poc\_len**[ i ] and **poc\_len\_delta\_minus1**[ i ], together with min\_poc\_lsb\_len\_minus4, specify the different lengths, in bits, used to represent the pic\_order\_cnt\_lsb syntax element for different coded pictures. The value of max\_temporal\_id\_poc\_len[ i ] shall be in the range of 0 to max\_temporal\_layers\_minus1, inclusive. The value of poc\_len\_delta\_minus1[ i ] shall be in the range of 0 to 27, inclusive. If not present, the value of max\_temporal\_id\_poc\_len[ i ] shall be inferred to be equal to 0 . If not present, the value of poc\_len\_delta\_minus1[ i ] shall be inferred to be equal to –1.

## Derivation of the POC LSB length

The variables PocLsbLen[ i ], for i values in the range of 0 to max\_temporal\_layers\_minus1, inclusive, are derived as follows:

len = min\_poc\_lsb\_len\_minus4 + 4  
for( i = 0; i <= max\_temporal\_layers\_minus1; i++ )  
 PocLsbLen[ i ] = len  
if( num\_add\_poc\_lsb\_lens > 0 ) {  
 for( i = 0; i < num\_add\_poc\_lsb\_lens – 1; i++ )  
 {  
 len += poc\_len\_delta\_minus1[ i ] + 1  
 for( tid = max\_temporal\_id\_poc\_len[ i ]; tid > max\_temporal\_id\_poc\_len[ i +1 ]; tid--)  
 PocLsbLen[ tid ] = len   
 }  
 len+= poc\_len\_delta\_minus1[ i ] + 1  
 while( tid >=0 )  
 PocLsbLen[ tid-- ] = len (8-x)  
}

## Slice header semantics

Let temporal\_id of the coded slice NAL unit be tId. The variable PocLSBLength is derived as equal to PocLsbLen[ tId ].

The variable MaxPicOrderCntLsb is derived as equal to 2PocLSBLength.

**pic\_order\_cnt\_lsb** specifies the value of the variable PicOrderCntLsb used in the derivation of the picture order count of the coded picture . The length of the pic\_order\_cnt\_lsb syntax element is PocLSBLength bits. The value of the pic\_order\_cnt\_lsb syntax element shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive.

The variable PicOrderCntLsb is derived as equal to pic\_order\_cnt\_lsb.

## Decoding process for picture order count

Output of this process is PicOrderCnt, the picture order count of the current picture.

Each coded picture is associated with one picture order count, called PicOrderCnt. PicOrderCnt indicates the picture order of the corresponding picture relative to the previous IDR picture in decoding order.

The variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows.

* If the current picture is an IDR picture, both prevPicOrderCntLsb and prevPicOrderCntMsb are set equal to 0.

– Otherwise (the current picture is not an IDR picture), the following applies.

– Let prevRefPic be the previous reference picture in decoding order ~~that has temporal\_id equal to 0~~ and prevPicOrderCnt be equal to PicOrderCnt of prevRefPic. The variable prevPicOrderCntLsb is set equal to Abs( pic\_order\_cnt\_lsb) % MaxPicOrderCntLsb ~~of prevRefPic~~, and the variable prevPicOrderCntMsb is set equal to prevPicOrderCnt – prevPicOrderCntLsb ~~PicOrderCntMsb  of prevRefPic~~.

PicOrderCntMsb of the current picture is derived as specified by the following pseudo-code:

if( ( pic\_order\_cnt\_lsb < prevPicOrderCntLsb ) &&  
 ( ( prevPicOrderCntLsb − pic\_order\_cnt\_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb (8-x)  
else if( (pic\_order\_cnt\_lsb > prevPicOrderCntLsb ) &&  
 ( (pic\_order\_cnt\_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb  
else  
 PicOrderCntMsb = prevPicOrderCntMsb

PicOrderCnt is derived as

PicOrderCnt = PicOrderCntMsb + pic\_order\_cnt\_lsb (8-x)

# 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).**