|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, IT, 14-22 July, 2011 | Document: JCTVC-F460 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Getting rid of “non-existing” pictures** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | **Qiu Shen**  Huawei Technologies Co., Ltd. Bantian, Longgang District  Shenzhen 518129, China  **Ye-Kui Wang**  Huawei Technologies Co., Ltd. 400 Crossing Blvd, 2nd Floor Bridgewater, NJ 08807, USA | Tel: Email: | +86 755 2878 8206 [shenqiu@huawei.com](mailto:shenqiu@huawei.com)  +1 908 541 3518 [yekui.wang@huawei.com](mailto:yekui.wang@huawei.com) |
| *Source:* | Huawei Technologies Co., Ltd. | | |

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

# Abstract

This document proposes to get rid of the processes for generating and handling of “non-existing” pictures in the HEVC decoding process. The proposed changes include a change to the frame\_num semantics, some alternative changes to the reference picture list construction process, and some changes to the reference picture marking process. Assuming that in HEVC the hypothetical reference decoder (HRD) process is specified basically the same as AVC; then those HRD texts mentioned “non-existing” pictures also need to be changed. An example of such changes to the HRD process is also provided, for information.

It is proposed to adopt

* the change to frame\_num semantics provided in Section 3
* one of the two alternative sets of changes to the reference picture list construction process and the reference picture marking process provided in Section 6
* the minor change needed to reference picture marking process

# Introduction

In HEVC WD3d8, the field temporal\_id is present in NAL unit header of the VCL NAL units (with NAL unit type 1, 4, or 5). The reference picture list construction process is specified such that, for any slice, any reference picture with greater temproal\_id value would never appear in the reference picture list of a slice during its reference picture list construction process, simply because those reference pictures would never be used for inter prediction by the current slice.

HEVC WD3d8 also includes the processes for generating and handling of “non-existing” pictures (though some text pieces are missing but basically the same process as in AVC is assumed).

Based on some analyses, it is concluded that generating and handling of “non-existing” pictures as part of the decoding process are not necessary; thanks to the inclusion of temporal\_id in VCL NAL unit headers and the above-mentioned reference picture list construction process.

Benefits of getting rid of “non-existing” pictures include 1) the memory consumed by “non-existing” pictures may be saved; 2) the decoding complexity needed for generating and handling of “non-existing” pictures can be avoided; 3) the processes for generating and handling of “non-existing” pictures can be waived in decoder implementations; and 4) the specification text becomes simpler.

The rest of this document is organized as follows. Section 2 provides some analyses from which it is concluded that generating and handling of “non-existing” pictures as part of the decoding process are not necessary. Section 3 provides the proposed change to the frame\_num semantics. Section 4 provides a ‘naive’ solution of the required changes to the reference picture list construction process and the reference picture marking process. Section 5 provides some analyses from which it is concluded that the ‘naive’ solution is not sufficient. Section 6 provides the two sets of alternative proposed changes to the reference picture list construction process and the reference picture marking process. Section 7 provides an example of the required changes to the hypothetical reference decoder (HRD) process, for information.

# Why “non-existing” pictures are not needed

“Non-existing” pictures are generated when there is a gap in frame\_num values (i.e., there is one or more frame\_num values missing) and when gaps\_in\_frame\_num\_value\_allowed\_flag in the active sequence parameter set is equal to 1. This occurs when some high temporal layers containing reference pictures are discarded from an entire temporally scalable bitstream, which results into a lower-frame-rate representation of the same video content. The “non-existing” pictures generated would have the missing frame\_num values, and the original coded pictures having these missing frame\_num values belong to those discarded high temporal layers.

To make temporal scalability work, a coded picture A with a particularly temporal\_id value cannot refer to another coded picture B with a greater temporal\_id value (hence belonging to a higher temporal layer), otherwise the coded picture B cannot be discarded and temporal scalability simply does not work. Therefore, “non-existing” pictures shall not be used for inter prediction reference. Then why to generate them at the first place? The main reason in our understanding is to help maintaining the same DPB status for decoding any particular inter-coded picture regardless of whether the entire stream or a bitstream subset is being decoded. Herein, the DPB status is defined as which reference pictures that are marked as “used for reference” are present in the DPB and other markings (i.e., “used for short-term reference”, “used for long-term reference”, “non-existing”) of these reference pictures. As long as the DPB status remains the same for a coded picture, the reference picture list(s) would remain the same, and hence the decoding result would also remain the same (assuming that the bitstream or bitstream subset is conforming, of course).

However, though DPB status being the same is a sufficient condition for the reference picture list(s) of an inter-coded picture to be the same, it is more than necessary. The sufficient and necessary condition for the reference picture list(s) of an inter-coded picture to be the same should be: 1) the reference pictures to be included in the reference picture list(s) are present in the DPB, and 2) their order in the initial reference picture list(s) remain the same.

If “non-existing” pictures are not generated, when decoding any particular coded picture, the first part of the sufficient and necessary condition would always be fulfilled. Then what is left is whether there is a way to ensure that the second half of the condition can be fulfilled. We know that for any inter-coded picture for which each slice contains reference picture list modification commands, whenever the first half of the condition is fulfilled, the second part would also be fulfilled. ***Therefore, we can conclude that sufficient and necessary condition can be fulfilled without generating “non-existing” pictures. Consequently, the processes for generating and handling of “non-existing” pictures are not necessary and can be removed from the decoding process.***

Interestingly, as shown in the following example, the decoding process with “non-existing” pictures can have some problem with “legal” use of reference picture list construction and marking settings. ***This might indicate a potential bug in AVC/SVC/MVC.***

Fig. 1 presents an example video sequence coded using hierarchical B coding structure with five temporal layers. Pictures are labeled with their frame\_num values which correspond to the decoding order. The pictures reside in the highest temporal layer and printed in italics are non-reference pictures, and others are reference pictures. In the figure, TL means temporal layer (and the value is equal to temporal\_id), when present, and POC means picture order count which correspond to the output order.



**Fig. 1 Hierarchical structure of five temporal scalable layers with frame\_num values shown**

In the following example:

* max\_num\_ref\_frames is set to 5 (this is the minimum possible value for such a coding structure, see [1])
* no reference picture list modification commands are used
* num\_ref\_idx\_l0\_active and num\_ref\_idx\_l1\_active are set to 1
* MMCO type 1 (i.e., memory\_management\_control\_operation is equal to 1) is included in some coded pictures of temporal level greater than 0 to mark the previous reference picture in decoding with the same temporal level as “unused for reference”

Table 1 shows the DPB status and reference picture lists for each picture when decoding all the five temporal levels for this example.

**Table 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Decoding order / frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference pictures lists when decoding the picture** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0 | 16 |
| 3 / 3 | 4(2) | yes | 0,16,8,4 | 0 | 8 |
| 4 / 4 | 2(3) | yes | 0,16,8,4,2 | 0 | 4 |
| 5 / 5 | 1(4) | no | 0,16,8,4,2 | 0 | 2 |
| 6 / 5 | 3(4) | no | 0,16,8,4,2 | 2 | 4 |
| 7 / 5 | 6(3) | yes(MMCO1) | 0,16,8,4,6 | 4 | 8 |
| 8 / 6 | 5(4) | no | 0,16,8,4,6 | 4 | 6 |
| 9 / 6 | 7(4) | no | 0,16,8,4,6 | 6 | 8 |
| 10 / 6 | 12(2) | yes(MMCO1) | 0,16,8, 6,12 | 8 | 16 |
| 11 / 7 | 10(3) | yes(MMCO1) | 0,16,8,12,10 | 8 | 12 |
| 12 / 8 | 9(4) | no | 0,16,8,12,10 | 8 | 10 |
| 13 / 8 | 11(4) | no | 0,16,8,12,10 | 10 | 12 |
| 14 / 8 | 14(3) | yes(MMCO1) | 0,16,8,12,14 | 12 | 16 |
| 15 / 9 | 13(4) | no | 0,16,8,12,14 | 12 | 14 |
| 16 / 9 | 15(4) | no | 0,16,8,12,14 | 14 | 16 |
| 17 / 9 | 32(0) | yes | 16,8,12,14,32 | 16 | NA |
| 18 / 10 | 24(1) | yes(MMCO1) | 16,12,14,32,24 | 16 | 32 |
| … | … | … | … | … | … |

However, when only the two lowest temporal levels are decoded, problem arises when decoding the picture with POC 24. When decoding the picture with POC 32, “non-existing” pictures with the missing frame\_num values 3 to 8, inclusive, corresponding to the pictures with POC values 4, 2, 6, 12, 10, 14 in the entire bitstream, are generated, and the sliding window operation marks picture with POC values 0, 16, 4, 2 are marked as “unused for reference”. For convenience, in Table 2, we still denote “non-existing” pictures with POC values, but with italic font. As can be seen, now when decoding the picture with POC 24, the only reference picture in RefPicList0 is the picture with POC 32, while when decoding the entire bitstream, it is the picture with POC 16. This will results the picture with POC 24 and all following pictures using it for inter prediction reference, directly or indirectly, to be most likely incorrectly decoded (i.e., the decoding result is different compared to when the entire bitstream is decoded).

**Table 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Decoding order/ frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference pictures lists when decoding the picture** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0 | 16 |
| 17 / 9 | 32(0) | yes | *6,12,10,14*,32 | 16 | NA |
| 18 / 10 | 24(1) | yes(MMCO1) | *6,12,10,14*,32,24 | 32 | 32 |
| … | … | … | … | … | … |

# Proposed change to frame\_num semantics

The proposed change to the semantics of frame\_num is as follows:

|  |
| --- |
| **frame\_num** is used as an identifier for pictures and shall be represented by log2\_max\_frame\_num\_minus4 + 4 bits in the bitstream. frame\_num is constrained as follows:  The variable PrevRefFrameNum is derived as follows.  – If the current picture is an IDR picture, PrevRefFrameNum is set equal to 0.  – Otherwise (the current picture is not an IDR picture), PrevRefFrameNum is set ~~as follows.~~  ~~– If the decoding process for gaps in frame\_num was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, PrevRefFrameNum is set equal to the value of frame\_num for the last of the "non-existing" reference frames inferred by the decoding process for gaps in frame\_num.~~  ~~– Otherwise, PrevRefFrameNum is set~~ equal to the value of frame\_num for the previous access unit in decoding order that contained a reference picture.  The value of frame\_num is constrained as follows.  – If the current picture is an IDR picture, frame\_num shall be equal to 0.  – Otherwise (the current picture is not an IDR picture), ~~referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture,~~ the value of frame\_num for the current picture shall not be equal to PrevRefFrameNum~~, unless both of the following conditions are true~~.  ~~– the current picture and the preceding reference picture belong to consecutive access units in decoding order~~  ~~– one or more of the following conditions is true~~  ~~– the preceding reference picture is an IDR picture~~  ~~– the preceding reference picture includes a memory\_management\_control\_operation syntax element equal to 5~~  ~~NOTE 1 – When the preceding reference picture includes a memory\_management\_control\_operation syntax element equal to 5, PrevRefFrameNum is equal to 0.~~  ~~– there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have frame\_num equal to PrevRefFrameNum~~  ~~– there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture~~  When the value of frame\_num is not equal to PrevRefFrameNum, and the following applies.  – There shall not be any previous frame in decoding order that is currently marked as "used for short-term reference" that has a value of frame\_num equal to any value taken on by the variable UnusedShortTermFrameNum in the following:  UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum while( UnusedShortTermFrameNum != frame\_num )   UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum  – The value of frame\_num is constrained as follows.  – If gaps\_in\_frame\_num\_value\_allowed\_flag is equal to 0, the value of frame\_num for the current picture shall be equal to ( PrevRefFrameNum + 1 ) % MaxFrameNum.  – Otherwise (gaps\_in\_frame\_num\_value\_allowed\_flag is equal to 1), the following applies.  – If frame\_num is greater than PrevRefFrameNum, there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which either of the following conditions is true.  – The value of frame\_num for the non-reference picture is less than PrevRefFrameNum.  – The value of frame\_num for the non-reference picture is greater than the value of frame\_num for the current picture.  – Otherwise (frame\_num is less than PrevRefFrameNum), there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which both of the following conditions are true.  – The value of frame\_num for the non-reference picture is less than PrevRefFrameNum.  – The value of frame\_num for the non-reference picture is greater than the value of frame\_num for the current picture.  A picture including a memory\_management\_control\_operation equal to 5 shall have frame\_num constraints as described above and, after the decoding of the current picture and the processing of the memory management control operations, the picture shall be inferred to have had frame\_num equal to 0 for all subsequent use in the decoding process. |

Note that the second part of the changes (under “The value of frame\_num is constrained as follows”) is a bug from dropping the field related text from the semantics, and should be fixed similarly even the “non-existing” pictures related processes are kept.

# A ‘naive’ solution

In a ‘naive’ solution, the required changes to the reference picture list construction process and the reference picture marking process are provided in this Section.

The change to subclause 8.2.2.2.1 (Initialisation process for the reference picture list for P slices) is as follows:

|  |
| --- |
| This initialisation process is invoked when decoding a P slice in a coded picture.  When this process is invoked, there shall be at least one reference picture that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") ~~and is not marked as "non-existing"~~. Pictures with higher values of temporal\_id than the current picture cannot be used for reference, and are not included in the reference picture list. |

The change to subclause 8.2.2.2.2 (Initialisation process for reference picture lists for B slices) is as follows:

|  |
| --- |
| This initialisation process is invoked when decoding a B slice in a coded picture.  For purposes of the formation of the reference picture lists RefPicList0 and RefPicList1 the term reference entry refers in the following to decoded reference pictures.  When this process is invoked, there shall be at least one reference entry that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") ~~and is not marked as "non-existing"~~. Pictures with higher values of temporal\_id than the current picture cannot be used for reference, and are not included in either RefPicList0 or RefPicList1.  For B slices, the order of short-term reference entries in the reference picture lists RefPicList0 and RefPicList1 depends on output order, as given by PicOrderCnt( ). ~~When pic\_order\_cnt\_type is equal to 0, reference pictures that are marked as "non-existing" as specified in subclause  are not included in either RefPicList0 or RefPicList1.~~  ~~NOTE 1 – When gaps\_in\_frame\_num\_value\_allowed\_flag is equal to 1, encoders should use reference picture list modification to ensure proper operation of the decoding process (particularly when pic\_order\_cnt\_type is equal to 0, in which case PicOrderCnt( ) is not inferred for "non-existing" pictures).~~ |

The derivation of the variable picNumLX in the reference picture list modification process for short-term reference pictures is changed as follows:

|  |
| --- |
| The variable picNumLX is derived as specified by the following pseudo-code:  if( picNumLXNoWrap > CurrPicNum )  picNumLX = picNumLXNoWrap − MaxPicNum (8‑6) else  picNumLX = picNumLXNoWrap  picNumLX shall be equal to the PicNum of a reference picture that is marked as "used for short-term reference" ~~and shall not be equal to the PicNum of a short-term reference picture that is marked as "non-existing"~~*.* The short-term reference picture with PicNum equal to picNumLX shall not have greater temporal\_id than the current slice. |

The decoding process for gaps in frame\_num is removed.

The marking process of s short-term reference picture as “unused for reference” is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 1.  Let picNumX be specified by  picNumX = CurrPicNum − ( difference\_of\_pic\_nums\_minus1 + 1 ). (8‑13)  The value of picNumX is used to mark the corresponding short-term reference picture as "unused for reference". The short-term reference picture identified by picNumX may or may not be present in the DPB. |

The assignment process of a LongTermFrameIdx to a short-term reference picture is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 3.  Given the syntax element difference\_of\_pic\_nums\_minus1, the variable picNumX is obtained as specified in subclause . picNumX shall refer to a picture marked as "used for short-term reference" ~~and not marked as "non-existing"~~.  When LongTermFrameIdx equal to long\_term\_frame\_idx is already assigned to a long-term reference picture, that picture is marked as "unused for reference".  The value of LongTermFrameIdx is used to mark the corresponding picture from "used for short-term reference" to "used for long-term reference". |

# Analysis of the ‘naive’ solution

Since no “non-existing” pictures are generated in decoder, the DPB (Decode Picture Buffer) status would be changed when some high temporal levels are discarded. In this Section we compare the DPB status for the cases of some typical coding configurations. The example video sequence shown in Fig.1 is still used for the analysis.

In the following example:

* max\_num\_ref\_frames is to 12 (to enable using sliding window only)
* no reference picture list modification commands are used
* num\_ref\_idx\_l0\_active and num\_ref\_idx\_l1\_active are set to 1
* no MMCO commands are used

Table 3 shows the DPB status and reference picture lists for each picture when decoding all the five temporal layers for this example.

**Table 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Decoding order/ frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference pictures lists when decoding the picture** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0 | 16 |
| 3 / 3 | 4(2) | yes | 0,16,8,4 | 0 | 8 |
| 4 / 4 | 2(3) | yes | 0,16,8,4,2 | 0 | 4 |
| 5 / 5 | 1(4) | no | 0,16,8,4,2 | 0 | 2 |
| 6 / 5 | 3(4) | no | 0,16,8,4,2 | 2 | 4 |
| 7 / 5 | 6(3) | yes | 0,16,8,4,2,6 | 4 | 8 |
| 8 / 6 | 5(4) | no | 0,16,8,4,2,6 | 4 | 6 |
| 9 / 6 | 7(4) | no | 0,16,8,4,2,6 | 6 | 8 |
| 10 / 6 | 12(2) | yes | 0,16,8,4,2,6,12 | 8 | 16 |
| 11 / 7 | 10(3) | yes | 0,16,8,4,2,6,12,10 | 8 | 12 |
| 12 / 8 | 9(4) | no | 0,16,8,4,2,6,12,10 | 8 | 10 |
| 13 / 8 | 11(4) | no | 0,16,8,4,2,6,12,10 | 10 | 12 |
| 14 / 8 | 14(3) | yes | 0,16,8,4,2,6,12,10,14 | 12 | 16 |
| 15 / 9 | 13(4) | no | 0,16,8,4,2,6,12,10,14 | 12 | 14 |
| 16 / 9 | 15(4) | no | 0,16,8,4,2,6,12,10,14 | 14 | 16 |
| 17 / 9 | 32(0) | yes | 0,16,8,4,2,6,12,10,14,32 | 16 | NA |
| 18 / 10 | 24(1) | yes | 0,16,8,4,2,6,12,10,14,32,24 | 16 | 32 |
| 19 / 11 | 20(2) | yes | 0,16,8,4,2,6,12,10,14,32,24,20 | 16 | 24 |
| 20 / 12 | 18(3) | yes | 16,8,4,2,6,12,10,14,32,24,20,18 | 16 | 20 |
| 21 / 13 | 17(4) | no | 16,8,4,2,6,12,10,14,32,24,20,18 | 16 | 18 |
| 22 / 13 | 19(4) | no | 16,8,4,2,6,12,10,14,32,24,20,18 | 18 | 20 |
| 23 / 13 | 22(3) | yes | 8,4,2,6,12,10,14,32,24,20,18,22 | 20 | 24 |
| 24 / 14 | 21(4) | no | 8,4,2,6,12,10,14,32,24,20,18,22 | 20 | 22 |
| 25 / 14 | 23(4) | no | 8,4,2,6,12,10,14,32,24,20,18,22 | 22 | 24 |
| 26 / 14 | 28(2) | yes | 4,2,6,12,10,14,32,24,20,18,22,28 | 24 | 32 |
| 27 / 15 | 26(3) | yes | 2,6,12,10,14,32,24,20,18,22,28,26 | 24 | 28 |
| 28 / 16 | 25(4) | no | 2,6,12,10,14,32,24,20,18,22,28,26 | 24 | 26 |
| 29 / 16 | 27(4) | no | 2,6,12,10,14,32,24,20,18,22,28,26 | 26 | 28 |
| 30 / 16 | 30(3) | yes | 6,12,10,14,32,24,20,18,22,28,26,30 | 28 | 32 |
| 31 / 17 | 29(4) | no | 6,12,10,14,32,24,20,18,22,28,26,30 | 28 | 30 |
| 32 / 17 | 31(4) | no | 6,12,10,14,32,24,20,18,22,28,26,30 | 30 | 32 |
| 33 / 17 | 48(0) | yes | 12,10,14,32,24,20,18,22,28,26,30,48 | 32 | NA |
| 34 / 18 | 40(1) | yes | 10,14,32,24,20,18,22,28,26,30,48,40 | 32 | 48 |
| … | … | … | … | … | … |

When some high temporal layers are discarded, gaps in frame\_num are present in the bitstream. In this case, the decoding process generates short-term “non-existing” pictures having the missing frame\_num values. Such “non-existing” pictures are handled similarly as normal short-term reference pictures in the sliding window reference picture marking process. Table 4 shows the DPB status and reference picture lists for each picture when decoding only the temporal layers with temporal\_id less than 2. After picture 8 (the picture with POC 8) is decoded, pictures 4, 2, 6, 12, 10, 14 are marked as “used for short-term reference” and “non-existing” and stored in the DPB. Similarly as in the previous example, for convenience, herein we still denote “non-existing” pictures with POC values, but with italic font.

**Table 4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Decding number / frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference picture lists when decoding the picture** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0 | 16 |
| 3 / 9 | 32(0) | yes | 0,16,8*,4,2,6,12,10,14*,32 | 0 | NA |
| 4 / 10 | 24(1) | yes | 0,16,8*,4,2,6,12,10,14*,32,24 | 16 | 32 |
| 5 / 17 | 48(0) | yes | *12,10,14*,32,24,*20,18,22,28,26,30*,48 | 32 | NA |
| 6 / 18 | 40(1) | yes | *10,14*,32,24,*20,18,22,28,26,30*,48,40 | 32 | 48 |
| … | … | … | … | … | … |

If “non-existing” pictures are not generated, the DPB the status will be different as shown in Table 5. However, the change of DPB status is that more reference pictures that are not marked as “non-existing” are present in the DPB for some pictures, but all the reference pictures need for decoding remain present in the DPB. Therefore, after reference picture list initialization reference pictures included in the reference picture lists remain the same, which means the decoding result would have no mismatch compared to when the entire bitstream is decoded.

**Table 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deoding number/ frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference picture lists when decoding the picture** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0 | 16 |
| 3 / 9 | 32(0) | Yes | 0,16,8,32 | 0 | NA |
| 4 / 10 | 24(1) | yes | 0,16,8,32,24 | 16 | 32 |
| 5 / 17 | 48(0) | yes | 0,16,8,32,24,48 | 32 | NA |
| 6 / 18 | 40(1) | yes | 0,16,8,32,24,48,40 | 32 | 48 |
| … | … | … | … | … | … |

The examples above can cover most cases of temporal scalable coding, but in some very special cases, the order of some reference pictures in the reference picture lists may change. In other words, the second part of the sufficient and necessary condition as described in Section 2 is not fulfilled. This is shown by the following example, wherein:

* max\_num\_ref\_frames is to 12 (to enable using sliding window only)
* no reference picture list modification commands are used
* num\_ref\_idx\_l0\_active and num\_ref\_idx\_l1\_active are set to 3 (this is the only difference compared to the previous example)
* no MMCO commands are used

Table 6 shows the DPB status and reference picture lists for each picture when decoding all the five temporal layers, while Table 7 shows the DPB status and reference picture lists for each picture when decoding only the temporal layers with temporal\_id less than 2.

**Table 6**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Decoding order/ frame\_num** | **POC(TL)** | **POC** | **Marked as “used for reference”** | **Reference pictures in the DPB after decoding the picture** | **Reference picture lists when decoding the picture** | |
|  | **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | 0 | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | 16 | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | 8 | yes | 0,16,8 | 0,16 | 16,0 |
| 3 / 3 | 4(2) | 4 | yes | 0,16,8,4 | 0,8,16 | 8,16,0 |
| 4 / 4 | 2(3) | 2 | yes | 0,16,8,4,2 | 0,4,8 | 4,8,16 |
| 5 / 5 | 1(4) | 1 | no | 0,16,8,4,2 | 0,2,4 | 2,4,8 |
| 6 / 5 | 3(4) | 3 | no | 0,16,8,4,2 | 2,0,4 | 4,8,16 |
| 7 / 5 | 6(3) | 6 | yes | 0,16,8,4,2,6 | 4,2,0 | 8,16,4 |
| 8 / 6 | 5(4) | 5 | no | 0,16,8,4,2,6 | 4,2,0 | 6,8,16 |
| 9 / 6 | 7(4) | 7 | no | 0,16,8,4,2,6 | 6,4,2 | 8,16,6 |
| 10 / 6 | 12(2) | 12 | yes | 0,16,8,4,2,6,12 | 8,4,0 | 16,8,4 |
| 11 / 7 | 10(3) | 10 | yes | 0,16,8,4,2,6,12,10 | 8,6,4 | 12,16,8 |
| 12 / 8 | 9(4) | 9 | no | 0,16,8,4,2,6,12,10 | 8,6,4 | 10,12,16 |
| 13 / 8 | 11(4) | 11 | no | 0,16,8,4,2,6,12,10 | 10,8,6 | 12,16,10 |
| 14 / 8 | 14(3) | 14 | yes | 0,16,8,4,2,6,12,10,14 | 12,10,8 | 16,12,10 |
| 15 / 9 | 13(4) | 13 | no | 0,16,8,4,2,6,12,10,14 | 12,10,8 | 14,16,12 |
| 16 / 9 | 15(4) | 15 | no | 0,16,8,4,2,6,12,10,14 | 14,12,10 | 16,14,12 |
| 17 / 9 | 32(0) | 32 | yes | 0,16,8,4,2,6,12,10,14,32 | 16,0 | NA |
| 18 / 10 | 24(1) | 24 | yes | 0,16,8,4,2,6,12,10,14,32,24 | 16,8,0 | 32,16,8 |
| 19 / 11 | 20(2) | 20 | yes | 0,16,8,4,2,6,12,10,14,32,24,20 | 16,12,8 | 24,32,16 |
| 20 / 12 | 18(3) | 18 | yes | 16,8,4,2,6,12,10,14,32,24,20,18 | 16,14,12 | 20,24,32 |
| 21 / 13 | 17(4) | 17 | no | 16,8,4,2,6,12,10,14,32,24,20,18 | 16,14,12 | 18,20,24 |
| 22 / 13 | 19(4) | 19 | no | 16,8,4,2,6,12,10,14,32,24,20,18 | 18,16,14 | 20,24,32 |
| 23 / 13 | 22(3) | 22 | yes | 8,4,2,6,12,10,14,32,24,20,18,22 | 20,18,16 | 24,32,20 |
| 24 / 14 | 21(4) | 21 | no | 8,4,2,6,12,10,14,32,24,20,18,22 | 20,18,16 | 22,24,32 |
| 25 / 14 | 23(4) | 23 | no | 8,4,2,6,12,10,14,32,24,20,18,22 | 22,20,18 | 24,32,22 |
| 26 / 14 | 28(2) | 28 | yes | 4,2,6,12,10,14,32,24,20,18,22,28 | 24,20,16 | 32,24,20 |
| 27 / 15 | 26(3) | 26 | no | 2,6,12,10,14,32,24,20,18,22,28,26 | 24,22,20 | 28,32,24 |
| 28 / 16 | 25(4) | 25 | no | 2,6,12,10,14,32,24,20,18,22,28,26 | 24,22,20 | 26,28,32 |
| 29 / 16 | 27(4) | 27 | yes | 2,6,12,10,14,32,24,20,18,22,28,26 | 26,24,22 | 28,32,26 |
| 30 / 16 | 30(3) | 30 | yes | 6,12,10,14,32,24,20,18,22,28,26,30 | 28,26,24 | 32,28,26 |
| 31 / 17 | 29(4) | 29 | no | 6,12,10,14,32,24,20,18,22,28,26,30 | 28,26,24 | 30,32,28 |
| 32 / 17 | 31(4) | 31 | no | 6,12,10,14,32,24,20,18,22,28,26,30 | 30,28,26 | 32,30,28 |
| 33 / 17 | 48(0) | 48 | yes | 12,10,14,32,24,20,18,22,28,26,30,48 | 32 | NA |
| 34 / 18 | 40(1) | 40 | yes | 10,14,32,24,20,18,22,28,26,30,48,40 | 32,24,48 | 48,32,24 |
| … | … | … | … | … | … | … |

**Table 7**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Coding number**  **/ frame\_num** | **POC(TL)** | **Marked as “used for reference”** | **pictures buffered in the DPB after coding the picture** | **Reference pictures in lists after ref pic list construction** | |
| **LIST0** | **LIST1** |
| 0 / 0 | 0(0) | yes | 0 | NA | NA |
| 1 / 1 | 16(0) | yes | 0,16 | 0 | NA |
| 2 / 2 | 8(1) | yes | 0,16,8 | 0,16 | 16,0 |
| 3 / 9 | 32(0) | Yes | 0,16,8,32 | 0,16 | NA |
| 4 / 10 | 24(1) | yes | 0,16,8,32,24 | 16,8,0 | 32,16,8 |
| 5 / 17 | 48(0) | yes | 0,16,8,32,24,48 | 32,16,0 | NA |
| 6 / 18 | 40(1) | yes | 0,16,8,32,24,48,40 | 32,24,16 | 48,32,24 |
| … | … | … | … | … | … |

In Table 6 when decoding picture 40 picture 16 does not appear in the DPB, therefore in the reference picture initialization process, after adding reference pictures 32 and 24 into RefPicList0, reference picture 48 is also added into RefPicList0. However, in Table 7, when only the two lowest temporal layers are decoded without “non-existing” pictures being generated, the reference pictures in the DPB are 0, 16, 8, 32, 24, 48, and 40 after decoding picture 48, therefore reference pictures 32, 24, 16 are added into RefPicList0. As seen, the third reference picture in the list changed which may affect the decoding result once the third reference picture is used for inter prediction reference.

Therefore, the ‘naive’ solution is not sufficient.

# Enhanced solutions

This Section provides two sets of alternative proposed changes to the reference picture list construction process and the reference picture marking process, referred to as two alternative enhanced solutions.

## Enhanced solution 1

In this solution, the reference picture list initialisation process is changed to avoid the problem as described in Section 5.

The decoding process for picture numbers in subclause 8.2.2.1 is changed as follows:

|  |
| --- |
| [Ed.Note(YKW): This subclause should be in parallel with subclauses 8.2.2 (reference picture list construction) and 8.2.3 (reference picture marking).]  This process is invoked when the decoding process for reference picture lists construction specified in subclause , or the decoded reference picture marking process specified in subclause ~~, or the decoding process for gaps in frame\_num specified in subclause  is invoked~~.  The variables FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx, and LongTermPicNum are used for the initialisation process for reference picture lists in subclause , the modification process for reference picture lists in subclause , and the decoded reference picture marking process in subclause ~~, and the decoding process for gaps in frame\_num in subclause~~.  To each short-term or long-term reference picture the variables FrameNum and FrameNumWrap are assigned as follows. First, FrameNum is set equal to the syntax element frame\_num that has been decoded in the slice header(s) of the corresponding short-term or long-term reference picture. Then the variable FrameNumWrap is derived as  if( FrameNum > frame\_num )  FrameNumWrap = FrameNum − MaxFrameNum (8‑1) else  FrameNumWrap = FrameNum  where the value of frame\_num used in Equation  is the frame\_num in the slice header(s) for the current picture.  Each long-term reference picture has an associated value of LongTermFrameIdx (that was assigned to it as specified in subclause ).  To each short-term or long-term reference picture a variable PicNum is assigned, and to each long-term reference picture a variable LongTermPicNum is assigned.  PicNum = FrameNumWrap (8‑2)  LongTermPicNum = LongTermFrameIdx (8‑3)  Let numLongTerm be the total number of reference pictures present in the decoded picture buffer that are marked as "used for long-term reference". For each short-term reference picture in the decoded picture buffer, the PicNum of which is denoted as PicNumX, the corresponding variables num1, num2, and MinPicNum are derived. Let num1 be the total number of latest reference pictures in decoding order that have PicNum in the range of picNumX + 1 to frame\_num − 1, inclusive, and that have been marked as “unused for reference”. Let num2 be the total number of latest reference pictures in decoding order that have PicNum in the range of picNumX + 1 to frame\_num − 1, inclusive, and that are marked as “used for long-term reference”, and that are present in the decoded picture buffer. The variable MinPicNum is derived as follows.  MinPicNum = frame\_num − (max\_num\_ref\_frames − numLongTerm + num1 + num2) (8‑x)  The value of frame\_num used in the above paragraph and equation is the frame\_num in the slice header(s) for the current picture. |

The change to subclause 8.2.2.2.1 (Initialisation process for the reference picture list for P slices) is as follows:

|  |
| --- |
| This initialisation process is invoked when decoding a P slice in a coded picture.  When this process is invoked, there shall be at least one reference picture that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") ~~and is not marked as "non-existing"~~. Pictures with higher values of temporal\_id than the current picture cannot be used for reference, and are not included in the reference picture list.  The reference picture list RefPicList0 is ordered so that short-term reference pictures have lower indices than long-term reference pictures.  The short-term reference pictures are ordered starting with the picture with the highest PicNum value and proceeding through in descending order to the picture with the lowest PicNum value, excluding any frame with a temporal\_id value higher than that of current picture or with PicNum value less than the corresponding MinPicNum.  The long-term reference pictures are ordered starting with the picture with the lowest LongTermPicNum value and proceeding through in ascending order to the picture with the highest LongTermPicNum value, excluding any frame with a temporal\_id value higher than that of the current picture.  ~~For example, when three reference pictures are marked as "used for short-term reference" with PicNum equal to 300, 302, and 303 and two reference pictures are marked as "used for long-term reference" with LongTermPicNum equal to 0 and 3, the initial index order is:~~  ~~– RefPicList0[0] is set equal to the short-term reference picture with PicNum = 303,~~  ~~– RefPicList0[1] is set equal to the short-term reference picture with PicNum = 302,~~  ~~– RefPicList0[2] is set equal to the short-term reference picture with PicNum = 300,~~  ~~– RefPicList0[3] is set equal to the long-term reference picture with LongTermPicNum = 0,~~  ~~– RefPicList0[4] is set equal to the long-term reference picture with LongTermPicNum = 3.~~ |

The change to subclause 8.2.2.2.2 (Initialisation process for reference picture lists for B slices) is as follows:

|  |
| --- |
| This initialisation process is invoked when decoding a B slice in a coded picture.  For purposes of the formation of the reference picture lists RefPicList0 and RefPicList1 the term reference entry refers in the following to decoded reference pictures.  When this process is invoked, there shall be at least one reference entry that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") ~~and is not marked as "non-existing"~~. Pictures with higher values of temporal\_id than the current picture cannot be used for reference, and are not included in either RefPicList0 or RefPicList1.  For B slices, the order of short-term reference entries in the reference picture lists RefPicList0 and RefPicList1 depends on output order, as given by PicOrderCnt( ). ~~When pic\_order\_cnt\_type is equal to 0, reference pictures that are marked as "non-existing" as specified in subclause  are not included in either RefPicList0 or RefPicList1.~~  ~~NOTE 1 – When gaps\_in\_frame\_num\_value\_allowed\_flag is equal to 1, encoders should use reference picture list modification to ensure proper operation of the decoding process (particularly when pic\_order\_cnt\_type is equal to 0, in which case PicOrderCnt( ) is not inferred for "non-existing" pictures).~~  The reference picture list RefPicList0 is ordered such that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows:   1. Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short‑term reference" and have PicNum value not less than the corresponding MinPicNum, and which have a value of temporal\_id equal to or lower than the temporal\_id of the current picture. When some values of entryShortTerm are present having PicOrderCnt( entryShortTerm ) less than PicOrderCnt( CurrPic ), these values of entryShortTerm are placed at the beginning of refPicList0 in descending order of PicOrderCnt( entryShortTerm ). All of the remaining values of entryShortTerm (when present) are then appended to refPicList0 in ascending order of PicOrderCnt( entryShortTerm ). 2. The long-term reference entries which have a value of temporal\_id equal to or lower than the temporal\_id of the current picture are ordered starting with the long-term reference entry that has the lowest LongTermPicNum value and proceeding through in ascending order to the long-term reference entry that has the highest LongTermPicNum value.   The reference picture list RefPicList1 is ordered so that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows:   1. Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short‑term reference" and have with PicNum value not less than the corresponding MinPicNum, and which have a value of temporal\_id equal to or lower than the temporal\_id of the current picture. When some values of entryShortTerm are present having PicOrderCnt( entryShortTerm ) greater than PicOrderCnt( CurrPic ), these values of entryShortTerm are placed at the beginning of refPicList1 in ascending order of PicOrderCnt( entryShortTerm ). All of the remaining values of entryShortTerm (when present) are then appended to refPicList1 in descending order of PicOrderCnt( entryShortTerm ). 2. Long-term reference entries which have a value of temporal\_id equal to or lower than the temporal\_id of the current picture are ordered starting with the long-term reference picture that has the lowest LongTermPicNum value and proceeding through in ascending order to the long‑term reference entry that has the highest LongTermPicNum value. 3. When the reference picture list RefPicList1 has more than one entry and RefPicList1 is identical to the reference picture list RefPicList0, the first two entries RefPicList1[ 0 ] and RefPicList1[ 1 ] are switched. |

The derivation of the variable picNumLX in the reference picture list modification process for short-term reference pictures is changed as follows:

|  |
| --- |
| The variable picNumLX is derived as specified by the following pseudo-code:  if( picNumLXNoWrap > CurrPicNum )  picNumLX = picNumLXNoWrap − MaxPicNum (8‑6) else  picNumLX = picNumLXNoWrap  picNumLX shall be equal to the PicNum of a reference picture that is marked as "used for short-term reference" ~~and shall not be equal to the PicNum of a short-term reference picture that is marked as "non-existing"~~*.* The short-term reference picture with PicNum equal to picNumLX shall not have greater temporal\_id than the current slice. |

The decoding process for gaps in frame\_num is removed.

The marking process of s short-term reference picture as “unused for reference” is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 1.  Let picNumX be specified by  picNumX = CurrPicNum − ( difference\_of\_pic\_nums\_minus1 + 1 ). (8‑13)  The value of picNumX is used to mark the corresponding short-term reference picture as "unused for reference". The short-term reference picture identified by picNumX may or may not be present in the decoded picture buffer. |

The assignment process of a LongTermFrameIdx to a short-term reference picture is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 3.  Given the syntax element difference\_of\_pic\_nums\_minus1, the variable picNumX is obtained as specified in subclause . picNumX shall refer to a picture marked as "used for short-term reference" ~~and not marked as "non-existing"~~.  When LongTermFrameIdx equal to long\_term\_frame\_idx is already assigned to a long-term reference picture, that picture is marked as "unused for reference".  The value of LongTermFrameIdx is used to mark the corresponding picture from "used for short-term reference" to "used for long-term reference". |

With the above changes, problems similar to the one as described in Section 5 can be avoided. Still using the example in Section 5, now when decoding picture 40, after adding picture 32, 24 into RefPicList0, picture 16 is not added as its PicNum (equal to 1) is smaller than the corresponding MinPicNum (equal to 6). Therefore picture 48 would be added into RefPicList0, and the result is then the same as when the entire bitstream is decoded.

## Enhanced solution 2

In this solution, reference picture list modification commands are mandated to be present at any inter-coded slice, which can also avoid the problem as described in Section 5.

In this case, then the reference picture list Initialisation process is not needed at all, and can be removed. The changes for this solution are as follows.

#### 7.3.3.1 Reference picture list construction syntax

|  |  |
| --- | --- |
| ref\_pic\_list\_construction( ) { | Descriptor |
| if( slice\_type % 5 != 2 && slice\_type % 5 != 4 ) { |  |
| do { |  |
| **rplc\_of\_pic\_nums\_idc** | ue(v) |
| if( rplc\_of\_pic\_nums\_idc = = 0 | |  rplc\_of\_pic\_nums\_idc = = 1 ) |  |
| **abs\_diff\_pic\_num\_minus1** | ue(v) |
| else if( rplc\_of\_pic\_nums\_idc = = 2 ) |  |
| **long\_term\_pic\_num** | ue(v) |
| } while( rplc\_of\_pic\_nums\_idc != 3 ) |  |
| } |  |
| if( slice\_type % 5 = = 1 ) { |  |
| do { |  |
| **rplc\_of\_pic\_nums\_idc** | ue(v) |
| if( rplc\_of\_pic\_nums\_idc = = 0 | |  rplc\_of\_pic\_nums\_idc = = 1 ) |  |
| **abs\_diff\_pic\_num\_minus1** | ue(v) |
| else if( rplc\_of\_pic\_nums\_idc = = 2 ) |  |
| **long\_term\_pic\_num** | ue(v) |
| } while( modification\_of\_pic\_nums\_idc != 3 ) |  |
| } |  |
| } |  |

#### 7.4.3.1 Reference picture list construction semantics

The syntax elements construction\_of\_pic\_nums\_idc, abs\_diff\_pic\_num\_minus1, and long\_term\_pic\_num specify the change from the initial reference picture lists to the reference picture lists to be used for decoding the slice.

The number of times that rplc\_of\_pic\_nums\_idc is not equal to 3 in the first loop shall be equal to num\_ref\_idx\_l0\_active\_minus1 + 1.

The number of times that rplc\_of\_pic\_nums\_idc is not equal to 3 in the second loop shall be equal to num\_ref\_idx\_l1\_active\_minus1 + 1.

**rplc\_of\_pic\_nums\_idc** together with abs\_diff\_pic\_num\_minus1 or long\_term\_pic\_num specifies which of the reference pictures are included into the reference picture lists. The values of rplc\_of\_pic\_nums\_idc are specified in . The value of the first rplc\_of\_pic\_nums\_idc in each loop shall not be equal to 3.

Table ‑ – rplc\_of\_pic\_nums\_idc operations for construction of reference picture lists

|  |  |
| --- | --- |
| **rplc\_of\_pic\_nums\_idc** | **construction specified** |
| 0 | abs\_diff\_pic\_num\_minus1 is present and corresponds to a difference to subtract from a picture number prediction value |
| 1 | abs\_diff\_pic\_num\_minus1 is present and corresponds to a difference to add to a picture number prediction value |
| 2 | long\_term\_pic\_num is present and specifies the long-term picture number for a reference picture |
| 3 | End loop for modification of the initial reference picture list |

**abs\_diff\_pic\_num\_minus1** plus 1 specifies the absolute difference between the picture number of the picture being moved to the current index in the list and the picture number prediction value. abs\_diff\_pic\_num\_minus1 shall be in the range of 0 to MaxPicNum − 1. The allowed values of abs\_diff\_pic\_num\_minus1 are further restricted as specified in subclause .

**long\_term\_pic\_num** specifies the long-term picture number of the picture being moved to the current index in the list. When decoding a coded picture, long\_term\_pic\_num shall be equal to a LongTermPicNum assigned to one of the reference pictures marked as "used for long-term reference".

|  |
| --- |
| 8.2.2 Decoding process for reference picture lists construction [Ed. (TW): clarify the following paragraphs]  This process is invoked at the beginning of the decoding process for each P or B slice.  Decoded reference pictures are marked as "used for short-term reference" or "used for long-term reference" as specified by the bitstream and specified in subclause . Short-term reference pictures are identified by the value of frame\_num. Long-term reference pictures are assigned a long-term picture index as specified by the bitstream and specified in subclause .  Subclause  is invoked to specify  – the assignment of variables FrameNum, FrameNumWrap, and PicNum to each of the short-term reference pictures, and  – the assignment of variable LongTermPicNum to each of the long-term reference pictures.  Reference pictures are addressed through reference indices as specified in subclause **Error! Reference source not found.**. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.  At the beginning of the decoding process for each slice, reference picture list RefPicList0, and for B slices RefPicList1, are derived as specified in subclause 8.2.2.2.  The number of entries in the reference picture list RefPicList0 is num\_ref\_idx\_l0\_active\_minus1 + 1, and for B slices the number of entries in the reference picture list RefPicList1 is num\_ref\_idx\_l1\_active\_minus1 + 1. A reference picture may appear at more than one index in the reference picture lists RefPicList0 or RefPicList1. 8.2.2.1 Decoding process for picture numbers [Ed.Note(YKW): This subclause should be in parallel with subclauses 8.2.2 (reference picture list construction) and 8.2.3 (reference picture marking).]  This process is invoked when the decoding process for reference picture lists construction specified in subclause  or the decoded reference picture marking process specified in subclause .  The variables FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx, and LongTermPicNum are used for the the construction process for reference picture lists in subclause 8.2.2.2 and the decoded reference picture marking process in subclause .  To each short-term reference picture the variables FrameNum and FrameNumWrap are assigned as follows. First, FrameNum is set equal to the syntax element frame\_num that has been decoded in the slice header(s) of the corresponding short-term reference picture. Then the variable FrameNumWrap is derived as  if( FrameNum > frame\_num )  FrameNumWrap = FrameNum − MaxFrameNum (8‑1) else  FrameNumWrap = FrameNum  where the value of frame\_num used in Equation  is the frame\_num in the slice header(s) for the current picture.  Each long-term reference picture has an associated value of LongTermFrameIdx (that was assigned to it as specified in subclause ).  To each short-term reference picture a variable PicNum is assigned, and to each long-term reference picture a variable LongTermPicNum is assigned.  PicNum = FrameNumWrap (8‑2)  LongTermPicNum = LongTermFrameIdx (8‑3) 8.2.2.2 Construction 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.  The following applies for construction of RefPicList0:   1. Let refIdxL0 be an index into the reference picture list RefPicList0. It is initially set equal to 0. 2. The corresponding syntax elements rplc\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.   – If rplc\_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 rplc\_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 (rplc\_of\_pic\_nums\_idc is equal to 3), the construction process for reference picture list RefPicList0 is finished.  The following applies for construction of RefPicList1:   1. Let refIdxL1 be an index into the reference picture list RefPicList1. It is initially set equal to 0. 2. The corresponding syntax elements rplc\_of\_pic\_nums\_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.   – If rplc\_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 rplc\_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 (rplc\_of\_pic\_nums\_idc is equal to 3), the construction process for reference picture list RefPicList1 is finished. 8.2.2.2.1 Construction 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 rplc\_of\_pic\_nums\_idc is equal to 0,  if( picNumLXPred − ( abs\_diff\_pic\_num\_minus1 + 1 ) < 0 )  picNumLXNoWrap = picNumLXPred − ( abs\_diff\_pic\_num\_minus1 + 1 ) + MaxPicNum (8‑4) else  picNumLXNoWrap = picNumLXPred − ( abs\_diff\_pic\_num\_minus1 + 1 )  – Otherwise (rplc\_of\_pic\_nums\_idc is equal to 1),  if( picNumLXPred + ( abs\_diff\_pic\_num\_minus1 + 1 ) >= MaxPicNum )  picNumLXNoWrap = picNumLXPred + ( abs\_diff\_pic\_num\_minus1 + 1 ) − MaxPicNum (8‑5) else  picNumLXNoWrap = picNumLXPred + ( abs\_diff\_pic\_num\_minus1 + 1 )  picNumLXPred is the prediction value for the variable picNumLXNoWrap. When the process specified in this subclause is invoked the first time for a slice (that is, for the first occurrence of rplc\_of\_pic\_nums\_idc equal to 0 or 1 in the ref\_pic\_list\_construction( ) syntax), picNumL0Pred and picNumL1Pred are initially set equal to CurrPicNum. After each assignment of picNumLXNoWrap, the value of picNumLXNoWrap is assigned to picNumLXPred.  The variable picNumLX is derived as specified by the following pseudo-code:  if( picNumLXNoWrap > CurrPicNum )  picNumLX = picNumLXNoWrap − MaxPicNum (8‑6) else  picNumLX = picNumLXNoWrap  picNumLX shall be equal to the PicNum of a reference picture that is marked as "used for short-term reference"*.* The short-term reference picture with PicNum equal to picNumLX shall not have greater temporal\_id than the current slice.  The following procedure is conducted to place the picture with short-term picture number picNumLX 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++ ] = short-term reference picture with PicNum equal to picNumLX nIdx = refIdxLX for( cIdx = refIdxLX; cIdx <= num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx++ ) (8‑7)  if( PicNumF( RefPicListX[ cIdx ] ) != picNumLX )  RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]  where the function PicNumF( RefPicListX[ cIdx ] ) is derived as follows.  – If the picture RefPicListX[ cIdx ] is marked as "used for short-term reference", PicNumF( RefPicListX[ cIdx ] ) is the PicNum of the picture RefPicListX[ cIdx ].  – Otherwise (the picture RefPicListX[ cIdx ] is not marked as "used for short-term reference"), PicNumF( RefPicListX[ cIdx ] ) is equal to MaxPicNum.  NOTE 1 – A value of MaxPicNum can never be equal to picNumLX.  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. 8.2.2.2.2 Construction process of reference picture lists for long-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 following procedure is conducted to place the picture with long-term picture number long\_term\_pic\_num 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++ ] = long-term reference picture with LongTermPicNum equal to long\_term\_pic\_num nIdx = refIdxLX for( cIdx = refIdxLX; cIdx <= num\_ref\_idx\_lX\_active\_minus1 + 1; cIdx++ ) (8‑8)  if( LongTermPicNumF( RefPicListX[ cIdx ] ) != long\_term\_pic\_num )  RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]  where the long-term reference picture with LongTermPicNum equal to long\_term\_pic\_num shall not have greater temporal\_id than the current slice, and the function LongTermPicNumF( RefPicListX[ cIdx ] ) is derived as follows.  – If the picture RefPicListX[ cIdx ] is marked as "used for long-term reference", LongTermPicNumF( RefPicListX[ cIdx ] ) is the LongTermPicNum of the picture RefPicListX[ cIdx ].  – Otherwise (the picture RefPicListX[ cIdx ] is not marked as "used for long-term reference"), LongTermPicNumF( RefPicListX[ cIdx ] ) is equal to 2 \* ( MaxLongTermFrameIdx + 1 ).  NOTE 1 – A value of 2 \* ( MaxLongTermFrameIdx + 1 ) can never be equal to long\_term\_pic\_num.  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. |

The decoding process for gaps in frame\_num is removed.

The marking process of s short-term reference picture as “unused for reference” is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 1.  Let picNumX be specified by  picNumX = CurrPicNum − ( difference\_of\_pic\_nums\_minus1 + 1 ). (8‑13)  The value of picNumX is used to mark the corresponding short-term reference picture as "unused for reference". The short-term reference picture identified by picNumX may or may not be present in the decoded picture buffer. |

The assignment process of a LongTermFrameIdx to a short-term reference picture is changed as follows:

|  |
| --- |
| This process is invoked when memory\_management\_control\_operation is equal to 3.  Given the syntax element difference\_of\_pic\_nums\_minus1, the variable picNumX is obtained as specified in subclause . picNumX shall refer to a picture marked as "used for short-term reference" ~~and not marked as "non-existing"~~.  When LongTermFrameIdx equal to long\_term\_frame\_idx is already assigned to a long-term reference picture, that picture is marked as "unused for reference".  The value of LongTermFrameIdx is used to mark the corresponding picture from "used for short-term reference" to "used for long-term reference". |

# HRD changes

It is assumed that the HRD process as specified in Annex C of the AVC standard is taken as the HRD process for HEVC. The required changes for removing of the generation and handling of “non-existing” pictures are described below.

Decoding of gaps in frame\_num and storage of "non-existing" frames are removed from the operations of the DPB and decoder conformance specifications.

Removal of pictures from the DPB before possible insertion of the current picture in the operation of the DPB is changed, such that the conditions for removing picture m from the DPB are modified as follows:

|  |
| --- |
| All pictures m in the DPB, for which all of the following conditions are true, are removed from the DPB:  – picture m is marked as "unused for reference" or picture m is a non-reference picture.  – picture m ~~is marked as "non-existing" or it~~ has OutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the current picture n; i.e., to,dpb( m ) <= tr( n )  When a frame in a frame buffer is removed from the DPB, the DPB fullness is decremented by one |

The "bumping" process of decoder conformance is changed, such that the cases for invoke the “bumping” process is modified as follows:

|  |
| --- |
| The "bumping" process is invoked in the following cases  ~~– There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and an empty frame buffer is needed for storage of an inferred "non-existing" frame, as specified in subclause .~~  – The current picture is an IDR picture and no\_output\_of\_prior\_pics\_flag is not equal to 1 and is not inferred to be equal to 1, as specified in subclause .  – The current picture has memory\_management\_control\_operation equal to 5, as specified in subclause .  – There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and an empty frame buffer is needed for storage of a decoded (non-IDR) reference picture, as specified in subclause .   * There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and the current picture is a non-reference picture that has OutputFlag equal to 1 and there are pictures in the DPB that are marked as "needed for output" that precede the current non-reference picture in output order, as specified in subclause , so an empty buffer is needed for storage of the current picture. |

# References

1. Qiu Shen, Ye-Kui Wang, Miska M. Hannuksela, Houqiang Li, and Yi Wang, "[Buffer requirement analysis and reference picture management for temporal scalable video coding](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4397030)", 16th International Packet Video Workshop (PV 2007), Nov. 2007, Lausanne, Switzerland.

# Patent rights declaration(s)

**Huawei Technologies Co., Ltd. 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).**