|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, SE, 11 – 20 July 2012 | Document: JCTVC-J0122 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG10: Hooks related to motion for the 3DV extension of HEVC** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Ying Chen Ye-Kui Wang Li Zhang  5775 Morehouse Drive 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)  1-858-651-6660 [lizhang@qualcomm.com](mailto:lizhang@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

Based on the current HEVC design, a reference picture from a different view in the 3DV extension of HEVC may be marked as a long-term reference picture. The purpose of this document is to discuss and propose motion related hooks in the HEVC base specification for the 3DV extension of HEVC. For comparison purpose, alternative solutions that do not need changes to the base specification but reportedly have drawbacks are also discussed. Among all the alternative solutions, the proponents prefer the fifth solution, for which the summary is provided in Section 6 and the proposed specification text changes are provided in Section 7 in the proposal document.

# Introduction

## Multiview video coding

In MPEG HTM (HEVC based 3DV Test Model), one view can be predicted from the other by the so-called disparity motion compensation. It is realized in a way that a decoded view component from a different view but in the same time instance can be added in a reference picture list of the current view component. For example, as shown in Figure 1, wherein the vertical indices (V1, V0 and V2) of the pictures correspond to view identifiers and the horizontal indices (T0 through T11) correspond to POC values. Each square is a view component and when a vertical arrow links view component A (e.g., T5/V0) to another view component B (e.g., T5/V1, in the same time instance), view component A can be added into the reference picture list 0 of view component B.

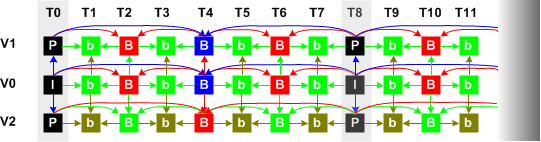


Figure 1: Typical motion prediction structure in 3DV (texture only).

Note that in the above example, both view component A and view component B have the same POC value (5).

A motion vector pointing to a view component of a different view is called disparity motion vector in this document and this view component is referred to as an inter-view reference picture.

## HLS-only extension and low-level hooks

In video codec implementations, the modules for handling high-level syntax (syntaxes at and above the slice header level) and related decoding processes can typically use firmware, while the low-level modules can typically be realized by hardware. A high-level syntax (HLS) only extension of a video codec involves only HLS changes relative to the base specification, such that existing, low-level hardware modules of the base specification can be reused without any changes, thus low-cost implementations of the extension becomes possible. If the HEVC base decoder hasn’t been implemented, an HLS-only extension decoder and the base decoder can be jointly implemented with the same efforts as those required for base decoder implementation from the hardware modules perspective.

The MVC extension of AVC was specified in such a way. In our understanding, the main reason for this was due to that the low-level tools specific for the MVC extension did not provide sufficient coding performance gain. MVC was developed after the finalization of the AVC base specification, and consequently, the advantage of considering MVC during the development of the AVC low-level design was not there. During the MVC development, to avoid problematic situations such as handling of zero motion related scaling in low-level decoding processes, some decoding processes belonging to low-level tools had to be disabled at the HLS design of MVC, by the following constrains:

* Temporal direct mode is disabled at the slice level when the co-located picture is an inter-view reference picture or an inter-view only reference picture
* Implicit weighted prediction related to inter-view (only) reference picture is disabled
* Considering an inter-view (only) reference picture as not a short-term: during the spatial direct mode, the accuracy of the motion vectors of the co-located short-term reference pictures is checked, this way, such checking is not needed of the co-located picture is an inter-view (only) reference picture.

If the AVC base specification had been designed taking into count MVC, it would have been possible not to disable the above tools, which could have led to better coding performance for MVC, by inclusion of some low-level hooks in the base specification that do not affect the coding performance of the base codec but could enable the extensions to achieve better coding performance.

As can be seen from above, an optimal design of an HLS-only video codec extension requires "collaborative" features in both the base specification and the extension specification based on the base specification. Obviously, if the extension is developed completely after the finalization of base specification, such "collaborative" features would not be possible.

In a stage that the designs of both the base and extension specifications can be modified, as is the case right now for HEVC and its 3DV extension, design of such "collaborative" features is perfectly possible. This is a significant benefit of the current overlapping schedules for the development of HEVC and its extensions.

In such a stage, a trade-off should be considered between base specification modifications and extension specification modifications. It seems preferable that the base specification should have lightweight modifications to includes hooks that enable all desirable tools or features to be used without problem in the extension, while still keeping the all (or at least most) of the tools or features in the base specification effectively unchanged.

Similarly as AVC and its MVC extension, depending on how well the being-developed low-level tools specific to the 3DV extension would work, it is possible that one or more profiles in this extension would involve only HLS changes. An HLS-only extension consists of possibly both high level syntax changes in the HEVC extension and hooks in the HEVC base specification.

### HLS-only changes in a potential HEVC multiview extension

The modifications in an extension must only be in the high-level, thus no modifications for syntax elements under slice header, and no CU level decoding process changes for the extension specification, for example, the motion vector prediction of the HEVC extension specification should be exactly the same as that in the HEVC base specification.

The HLS changes of the potential HEVC multiview extension are normative decoder changes of the extension specification. However, such changes do not need to be made to the base specification, though some may be included in the base specification as informative examples.

### Hooks to be enabled in HEVC base specification for an HLS-only extension

To enable tools or features such as efficient inter-view prediction, both modifications in the HEVC extension and base specifications might be needed.

The base specification changes, which do not impact the typical decoding processes or coding efficiency of the base HEVC decoders, but target at enabling tools or features in the extension specification, are called hooks. To achieve optimal design for an HLS-only extension, we may need both hooks in base specification and HLS changes in the extension specification.

If the hooks in base specifications are not defined well, certain desirable tools or features might have to either be disallowed in the extension specification or requires significant design overhead to implement. The above drawbacks may also lead to noticeable coding efficiency drop of the HLS-only extension.

## Terms used in this document

For simplicity in description, a motion vector is considered as consisting of both the reference index (the pointer to the reference picture) and the horizontal (x) and vertical (y) values of a motion vector itself. Equivalently, the reference picture corresponding to the reference index is also referred to as the reference picture the motion vector refers to.

If a motion vector refers to a reference picture in the same view, it is referred to as a temporal motion vector. If a motion vector refers to a reference picture of a different view, it is referred to as a disparity motion vector. A temporal motion vector can be either a short-term temporal motion vector (short-term motion vector in short) or a long-term temporal motion vector (long-term motion vector in short). For example, a motion vector is short-term if it refers to a short-term reference picture.

Unless otherwise mentioned, a disparity motion vector and a long-term motion vector belong to different types of motion vectors.

Short-term and long-term reference pictures from the same view are collectively referred to as temporal reference pictures.

## Organization of the document

The rest of this document is organized as follows. In Section 2, we discuss motion coding related tools or features that are desirable for the 3DV extension of HEVC. Then in Section 3, places in the current HEVC base specification that may require changes for inclusion of low-level hooks to enable the desirable tools or features in an HLS-only 3DV extension of HEVC are discussed. Various solutions, based whether inter-view (only) reference pictures are consistently indicated as long-term reference pictures and whether inclusion of additional hooks in the base specification is needed, are discussed in Sections 4, 5 and 6. Section 7 provides the proposed specification text changes for the most preferable solution.

# Desirable motion related features for the 3DV extension of HEVC

Essentially, for an optimal design related to motion vector prediction in both the base specification and the extension specification, the following should be supported:

* Advanced motion vector prediction (including both AMVP and merge) between temporal short-term motion vectors in different layers should still be possible.
* Predictions between motion vectors of different types (temporal short-term, temporal long-term and inter-view motion vectors) should be enabled in some scenarios and should be disabled in some other scenarios.

Listed below are the scenarios we identified that may be related to motion vector prediction in HEVC and its HLS-only 3DV extension.

1. A temporal motion vector should be used to predict temporal motion vectors with necessary scaling, as defined in the current HEVC base specification. This scenario is named “ST Scale” for simplicity.
2. A disparity motion vector should not be scaled to predict a temporal motion vector, as the POC based scaling might either fail or return zero motion. Similarly, a temporal motion vector should not be scaled to predict a disparity motion vector. This scenario is named “No LT scale” for simplicity.
3. It should be possible to disable predicting a disparity motion vector from a short-term motion vector, e.g., during AMVP, and to disable prediction a temporal short-term motion vector from a disparity motion vector, as the prediction does not yield good coding performance due to the different natural characteristics of these two types of motion vectors, as shown in Figure 2, wherein a dashed line with a “x” denotes that the prediction from two motion vectors are to be disabled. This scenario is named “No ST pred. for inter-view” for simplicity.
   1. Disparity motion vectors typically correspond to the local disparity of the same object in different views. However, temporal motion vectors typically correspond to the motion of an object.
   2. In HTM, the 3DV reference software, the prediction between motion vectors of the above two types is disabled.

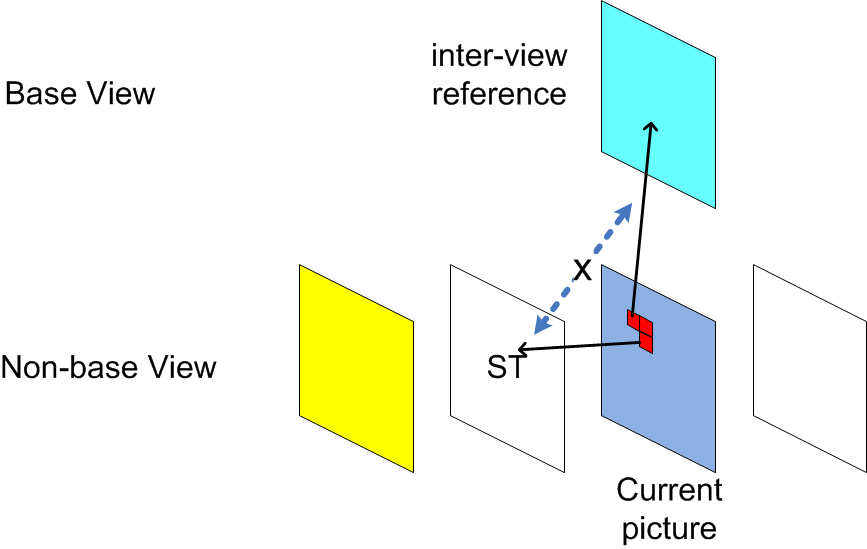


Figure 2: Neighboring blocks with both short-term mv and inter-view MV.

1. It should be possible to identify whether two disparity motion vectors correspond to the same reference picture. Note that two entries with the same POC value might not correspond to the same reference picture (view component). This scenario is named “multiple inter-view” for simplicity, as illustrated in Figure 3.
   1. When an entry in RefPicList0 and an entry in RefPicList1 are both inter-view reference pictures, it should be possible during the AMVP, to identify whether these two reference pictures are the same. If they are the same, the corresponding motion vectors can be used to predict each other directly. Otherwise, predicting the corresponding motion vectors may not be helpful.
   2. When a RefPicListX contains two entries that are inter-view reference pictures, it should be possible during the AMVP, to identify whether these two reference pictures are not the same.

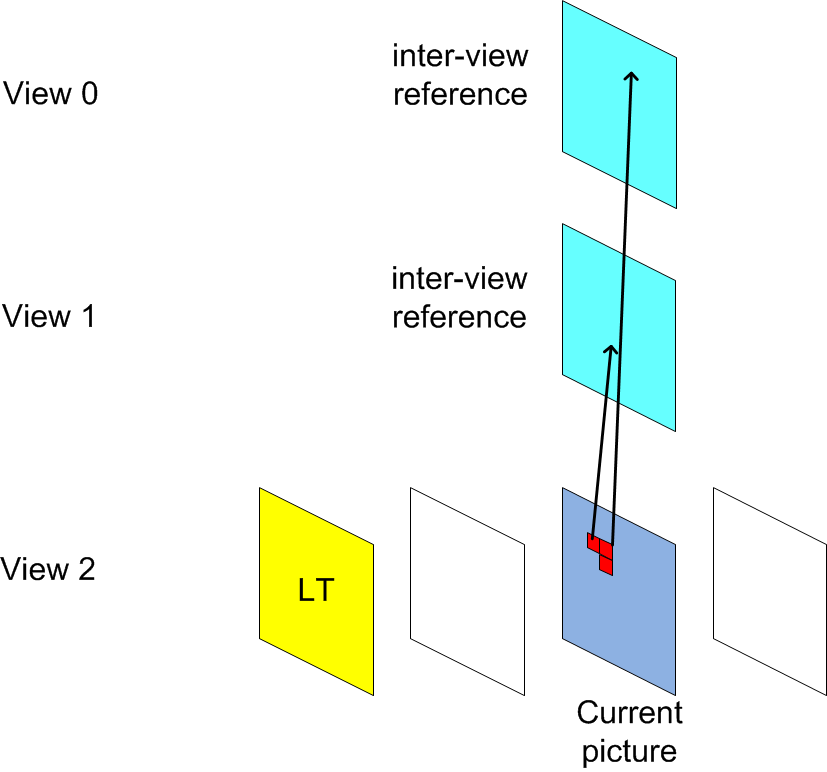


Figure 3: Neighboring blocks with MVs referring to different inter-view pictures

1. It should be possible to disable predicting a disparity motion vector from a temporal long-term motion vector, and to disable predicting a temporal long-term motion vector from a disparity motion vector, based on the similar reason as the “No ST pred. for inter-view” scenario. This scenario is named “No LT pred. for inter-view” for simplicity.

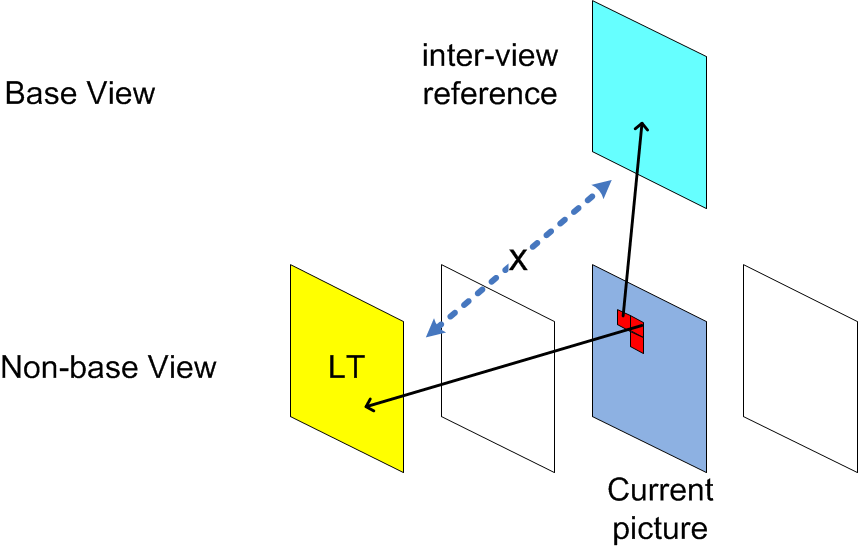


Figure 4: Neighboring blocks with both long-term mv and inter-view MV.

1. It should be possible to disable predicting between temporal short-term motion vector and temporal long-term motion vector, e.g., during AMVP, since most likely the two reference pictures may have significantly long POC distance. This scenario is named “No LT pred. for ST” for simplicity, as shown in Figure 5.

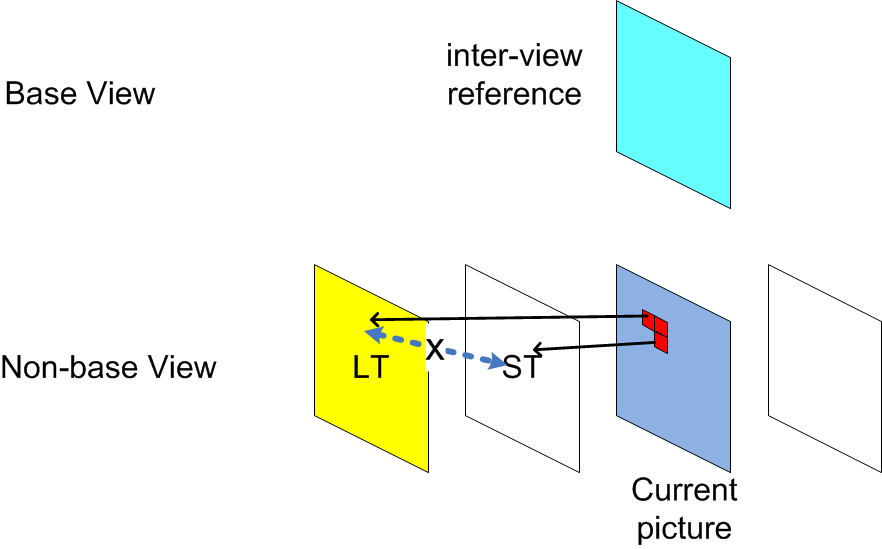


Figure 5: Neighboring blocks with both long-term mv and short-term MV

1. It should be possible to disable predicting temporal long-term motion vectors during AMVP. This scenario is named “No pred. between LTs”, as shown in Figure 6. Two long-term pictures typically have a long POC distance thus predicting two long-term motion vectors referring to two different long-term reference pictures can be inefficient.

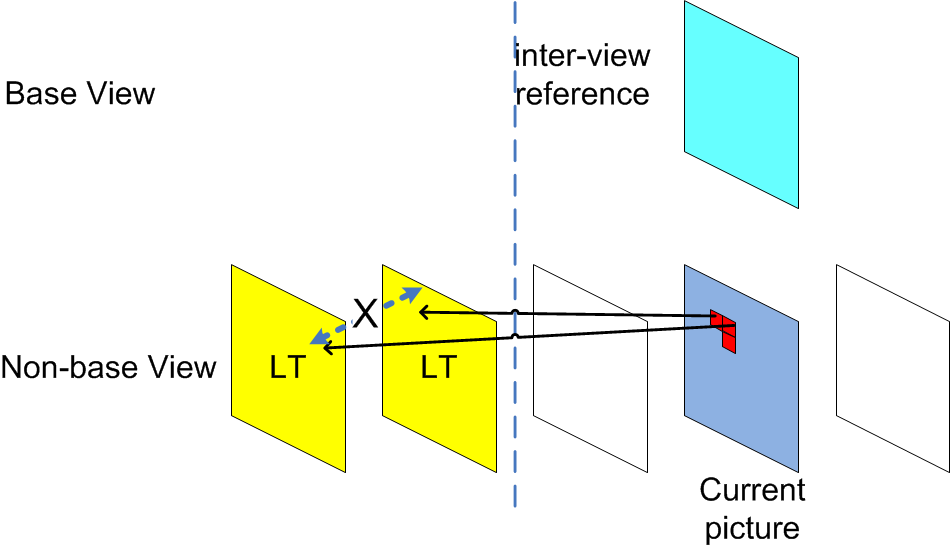


Figure 6: Neighboring blocks with two long-term MVs.

# Potential problematic places in the current HEVC specification

In the design of HEVC base specification as well as its 3DV extension, constrains in AVC may be similarly applied to create an HEVC MVC extension.

However, the places in the current HEVC specification for the following features require changes for inclusion of low-level hooks to enable the desirable motion related features in an HLS-only 3DV extension of HEVC:

* POC based scaling is required during AMVP for spatial neighboring motion candidates. (note that in AVC, only temporal motion vector predictors may be scaled).
* The AMVP and the merge modes require the MV candidates to be as closer to the final motion vector as possible, thus the motion vector coding efficiency is sensitive to the type of the motion vectors (see the 3rd scenario described in Section 2). However, the current HEVC specification enables prediction between any types of motion vectors, while the motivation of the motion vector scaling based on POC distance is mainly for temporal motion vectors, especially between motion vectors referring to reference pictures in a relatively close POC distance.
* The current merge and AMVP designs have multiple function calls related to POC values, especially on whether two reference pictures are identical, in the following listed sub-clauses:

– 8.5.2.1.3 Derivation process for combined bi-predictive merging candidates (2 times)

* To check if the possible combined bi-predictive merging candidate have the same reference picture and the motion vector for RefPicList0 and RefPicList1.

– 8.5.2.1.6 Derivation process for motion vector predictor candidates (10 times)

* To check if two reference pictures are identical when there are in different reference picture lists.
* To use the POC values to scale the motion vector when the reference picture is not the same: disabled between long-term and short-term.

– 8.5.2.1.7 Derivation process for temporal luma motion vector prediction (11 times)

* To check if every reference picture has a smaller or equal POC value as that of the current picture.
* To check the POC distance of the current reference picture to the current picture and the POC distance of the co-located picture and its reference picture.
* To use the POC values to scale the motion vector when the reference picture is not the same: disabled between long-term and short-term.

# Solutions based on long-term pictures and no additional hooks

Multiple solutions are proposed for an HLS-only 3DV extension of HEVC, each containing HLS syntax changes to be included in the extension but no additional hook in the HEVC base specification.

In the solutions described in this section, each inter-view reference picture is considered as a long-term reference picture. So even the “long-term” and “inter-view” motion vectors are essentially different types, in the HEVC base decoder or HEVC HLS-only 3DV decoder, they are treated transparently just as the same type of motion vectors: long-term motion vectors, during motion vector prediction, e.g., AMVP.

## Solution 1

An inter-view reference picture has the same POC value as the current picture during the entire decoding process of the current picture. Two inter-view reference pictures are considered as identical, meaning that although they are not the same picture, but the decoder treat them as if they were the same picture, because POC is the only picture identification.

In this solution, each picture, if it is to be used for inter-view reference (this is typically the case for each picture in the base view), is indicated as long-term reference pictures. A picture, if it is not used for inter-view prediction, may be indicated as either short-term or long-term.

### Changes in the extension (informative)

A picture (view component) is identified by two properties: POC and a second-dimension picture identification, e.g., view\_id.

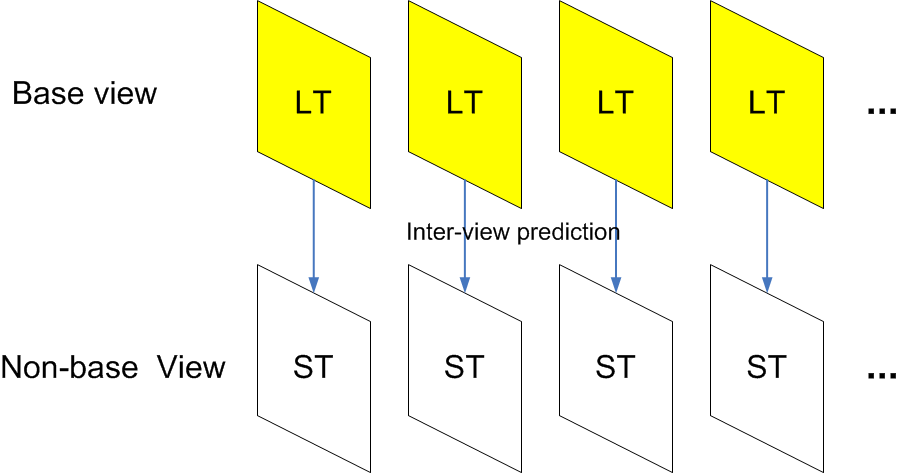
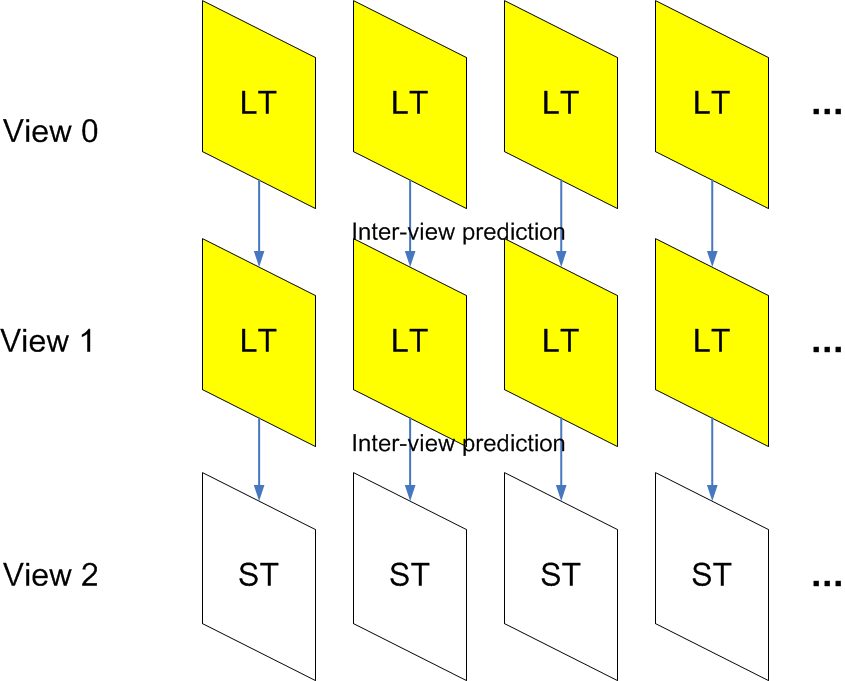
 

Figure 7: Pictures in dependent views are coded as long-term.

### Pros and cons

#### Pros.

* No more hooks in the HEVC base specification are needed.

#### Cons.

* Large amount of bits to signal the long-term pictures for all views used for inter-view prediction, including the base view.
* Sub-optimal efficiency for AMVP or merge.
  + Temporal motion vectors in all views used for inter-view prediction, including the base view, are not scaled at all since all the pictures are coded as long-term reference pictures if to be used for inter-view prediction. This is because although practically the motion vectors in the base view are short-term, they are forced to be indicated as long-term.

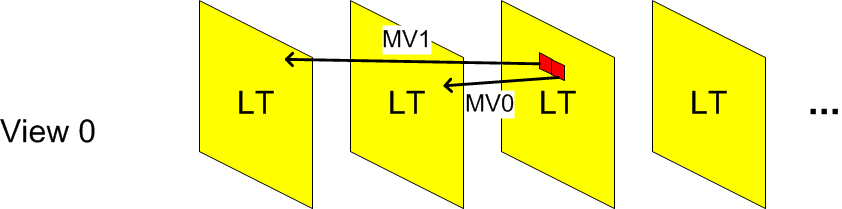


Figure 8: MV1 is directed used to predict MV0 in the base view.

* + Prediction between long-term (inter-view) motion vectors and short-term motion vectors at the enhancement view is currently enabled/used in the base specification but that is not desirable.
  + Two inter-view reference pictures are considered identical in the following sub-clauses
    - 8.5.2.1.3 Derivation process for combined bi-predictive merging candidates
    - 8.5.2.1.6 Derivation process for motion vector predictor candidates

### Supported scenarios

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 | Scenario 7 |
| ST Scale | No LT scale | No ST pred. for inter-view | Multiple Inter-view | No LT pred. for inter-view | No LT pred. for ST | No pred. between LTs |
| Partially | Y | N | N | N | N | N |

## Solution 2

An inter-view reference picture has the same POC value as the current picture during the entire decoding process of the current picture.

In this solution, each picture, if it is to be used for inter-view reference (this is typically the case for each picture in the base view), may be indicated as either short-term or long-term reference picture. However, during the moments when it is being used for inter-view prediction, if it was indicated as short-term, it is temporally marked as long-term.

### Changes in the extension (informative)

A picture (view component) is identified by two properties: POC and the second-dimension picture identification, e.g., view\_id.

An additional picture marking process is introduced before coding a current view component, to mark all the inter-view reference pictures as long-term.

Another picture marking process is introduced after coding a current view component, to mark each inter-view reference picture to either long-term, short-tem, or “unused for reference” depending or its previous status before the current view component is coded.

### Pros and cons

#### Pros.

* No more hooks in the HEVC base specification are needed.

#### Cons.

* The marking status of view components need to be frequently switched between long-term and short-term, and additional decoding processes are needed to make such frequently changes of marking.
* Sub-optimal efficiency for AMVP or merge.
  + Prediction between long-term (inter-view) motion vectors and short-term motion vectors at the enhancement view is currently enabled/used but not desirable.
  + Prediction between temporal long-term motion vectors and inter-view motion vectors (considered as long-term) at the enhancement view is currently enabled/used but not desirable.
  + Two inter-view reference pictures are considered identical in the following sub-clauses
    - 8.5.2.1.3 Derivation process for combined bi-predictive merging candidates
    - 8.5.2.1.6 Derivation process for motion vector predictor candidates

### Supported scenarios

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 | Scenario 7 |
| ST Scale | No LT scale | No ST pred. for inter-view | Multiple Inter-view | No LT pred. for inter-view | No LT pred. for ST | No pred. between LTs |
| Y | Y | N | N | N | N | N |

## Solution 3

Similar to solution 2, however, inter-view reference picture, after marked as long-term also have its POC value mapped to a new POC value, which is not equivalent to any existing reference picture.

For example, the current view component belongs to view 3 (assuming view identifier is equal to view order index), and has POC value equal to 5. The two inter-view reference pictures will have their POC values (which are both 5) converted to e.g., 1025 and 2053.

After decoding the current view component, the POC values of the inter-view pictures are converted back to both 5.

### Changes in the extension (informative)

In additional to solution 2, HLS changes are introduced such that any two inter-view reference pictures can be distinguished.

### Pros and cons

#### Pros.

* No more hooks in the HEVC base specification are needed.

#### Cons.

* The marking status of view components need to be frequently switched between long-term and short-term.
* The POC value of a view component is changing frequently, and additional decoding processes are needed to make such frequently changes of POC values.
* Sub-optimal efficiency for AMVP or merge
  + Prediction between long-term (inter-view) motion vectors and short-term motion vectors at the enhancement view is currently enabled/used but not desirable.
  + Prediction between temporal long-term motion vectors and inter-view motion vectors (considered as long-term) at the enhancement view is currently enabled/used but not desirable.

### Supported scenarios

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 | Scenario 7 |
| ST Scale | No LT scale | No ST pred. for inter-view | Multiple Inter-view | No LT pred. for inter-view | No LT pred. for ST | No pred. between LTs |
| Y | Y | N | Y | N | N | N |

# Solutions based on long-term pictures and additional hooks

An inter-view reference picture is considered as a long-term reference picture. However, prediction between short-term and long-term MVs during AMVP is disabled.

## Solution 4

### Changes in the extension (informative)

Similar to solution 2.

### Hooks in the HEVC base specification

The same as proposed in JCTVC-J0121, wherein prediction between long-term motion vectors and short-term motion vectors are disabled.

### Pros and cons

#### Pros.

* Consistent to HTM in terms of disabling prediction between inter-view motion vectors and short-term motion vectors.

#### Cons.

* Some changes to the base specification are needed
* Sub-optimal efficiency for AMVP or merge
  + Two inter-view reference pictures are considered identical in the following sub-clauses
    - 8.5.2.1.3 Derivation process for combined bi-predictive merging candidates
    - 8.5.2.1.6 Derivation process for motion vector predictor candidates

#### Supported scenarios

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 | Scenario 7 |
| ST Scale | No LT scale | No ST pred. for inter-view | Multiple Inter-view | No LT pred. for inter-view | No LT pred. for ST | No pred. between LTs |
| Y | Y | Y | N | N | N | N |

# Solutions not based on long-term pictures and with additional hooks

An inter-view reference picture is considered as neither short-term reference picture nor a long-term reference picture. In addition, prediction between short-term and long-term pictures during AMVP is disabled.

## Solution 5

A new motion vector/reference picture type and additional picture identification is introduced are introduced in the HEVC base specification.

### Changes in the extension (informative)

A picture (view component) is identified by two properties: POC and a second-dimension picture identification, e.g., view\_id or view order index. An inter-view reference picture has the same POC value as the current picture in the same access unit throughout the decoding process.

In addition, an inter-view reference picture is marked as a different type of reference pictures (non-temporal reference picture), and a function AddPicId( ) to return the additional picture identification is defined as part of the high-level decoding process, and the return value could be the following:

* View order index in multiview context
* A generic layer ID (layer\_id), e.g., the value of reserved\_one\_5bits minus 1, wherein reserved\_one\_5bits is as specified in the HEVC base specification

### Hooks in the HEVC base specification

The following hooks are required to be included in the HEVC base specification:

* When identifying a reference picture during AMVP and merge, additional picture identification, through the return of a function, is checked in addition to POC to identify a picture. In the context of 2D video decoding, the value of the additional picture identification is always equal to 0.
* Prediction between temporal motion vector and non-temporal motion vector is disabled during AMVP (including TMVP).
* Prediction between temporal short-term motion vector and temporal long-term motion vector is enabled.

### Pros and cons

#### Pros.

* Optimal efficiency for both AMVP and merge thanks to no non-desirable motion prediction decoding processes and support of all the desirable scenarios
* Minimum amount of bits to signal the long-term pictures for all views used for inter-view prediction, no need to force the e.g., base view to code most of the pictures as long-term, when frequently switching the status of the long-term reference pictures is not used.
* No need of frequently switched marking or additional marking processes when disabling zero based scaling of motion vectors
* No need of frequently changing POC values or additional POC decoding processes when distinguishing two inter-view motion vectors

#### Cons.

* Some changes to the base specification are needed, including definition of functions for reference picture types and additional picture identification

### Supported scenarios

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario 1 | Scenario 2 | Scenario 3 | Scenario 4 | Scenario 5 | Scenario 6 | Scenario 7 |
| ST Scale | No LT scale | No ST pred. for inter-view | Multiple Inter-view | No LT pred. for inter-view | No LT pred. for ST | No pred. between LTs |
| Y | Y | Y | Y | Y | Y | Y |

# Proposed text for the most preferable solution

The proposed specification text changes for solution 5 described above (which is the most preferable solution by the authors) is provided in this section. The text of solution 4 is included in JCTVC-J0121.

## Changes in subclause 8.3.1

8.3.1 Decoding process for picture order count

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

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, to represent picture order differences between pictures for motion vector derivation, for implicit mode weighted prediction in B slices (see subclause 8.5.2.2.3), and for decoder conformance checking (see subclause C.5).

Each coded picture is associated with one picture order count, denoted as PicOrderCntVal.

When none of the following conditions is true:

* The current picture is an IDR
* The current picture is a BLA picture
* The current picture is a CRA picture and is the first coded picture in the bitstream

the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows. Let prevRefPic be the previous reference picture in decoding order that has temporal\_id equal to or less than the temporal\_id of the current picture. The variable prevPicOrderCntLsb is set equal to pic\_order\_cnt\_lsb of prevRefPic, and the variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevRefPic.

The variable PicOrderCntMsb of the current picture is derived as follows.

* If the current picture is an IDR or a BLA picture, or if the first coded picture in the bitstream is a CRA picture and the current picture is the first coded picture in the bitstream, PicOrderCntMsb is set equal to 0.
* Otherwise, PicOrderCntMsb 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 (8‑1)  
else if( (pic\_order\_cnt\_lsb > prevPicOrderCntLsb ) &&  
 ( (pic\_order\_cnt\_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb  
else  
 PicOrderCntMsb = prevPicOrderCntMsb

PicOrderCntVal is derived as

PicOrderCntVal = PicOrderCntMsb + pic\_order\_cnt\_lsb (8‑2)

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since pic\_order\_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

The value of PicOrderCntVal shall be in the range of −263 to 263 − 1, inclusive. In one coded video sequence, the PicOrderCntVal values for any two coded pictures shall be different.

NOTE 2 – Retention of only the 32 least significant bits of PicOrderCntVal is sufficient for proper decoder operation, enabling unique identification of pictures.

The function PicOrderCnt( picX ) is specified as follows:

PicOrderCnt( picX ) = PicOrderCntVal of the picture picX (8‑3)

The function DiffPicOrderCnt( picA, picB ) is specified as follows:

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (8‑4)

The bitstream shall not contain data that result in values of DiffPicOrderCnt( picA, picB ) used in the decoding process that exceed the range of −215 to 215 − 1, inclusive.

NOTE 3 – Let X be the current picture and Y and Z be two other pictures in the same sequence, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt( X, Y ) and DiffPicOrderCnt( X, Z ) are positive or both are negative.

NOTE 4 – Many encoders assign PicOrderCntVal proportional to the sampling time of the corresponding picture relative to the sampling time of the previous IDR or BLA picture.

The function AddPicId( pic ) returns 0.

The function AddPicId( picX, refIdx, LX ) returns AddPicId( pic ), wherein pic is the reference picture with index refIdx from reference picture list LX of the picture picX.

## Changes in AMVP

8.5.2.1.6 Derivation process for motion vector predictor candidates

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are (with N being replaced by A, or B)

* the motion vectors mvLXN of the neighbouring prediction units,
* the availability flags availableFlagLXN of the neighbouring prediction units.



Figure 8‑3 – Spatial motion vector neighbours

The variable isScaledFlagLX with X being 0 or 1 is set equal to 0.

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. Let a set of two sample locations be (xAk, yAk), with k = 0, 1, specifies sample locations with xAk = xP − 1, yA0 = yP + nPSH and yA1 = yA0 - MinPuSize. The set of sample locations ( xAk, yAk ) represent the sample locations immediately to the left side of the left partition boundary and it’s extended line.
2. Let the availability flag availableFlagLXA be initially set equal to 0 and the both components of mvLXA are set equal to 0.
3. When one or more of the following conditions are true, the variable isScaledFlagLX is set equal to 1.

* the prediction unit covering luma location ( xA0, yA0 ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] and PredMode is not MODE\_INTRA.
* the prediction unit covering luma location ( xA1, yA1 ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] and PredMode is not MODE\_INTRA.

1. For ( xAk, yAk ) from ( xA0, yA0 ) to ( xA1, yA1 ) where yA1 = yA0 − MinPuSize, if availableFlagLXA is equal to 0, the following applies:

* If the prediction unit covering luma location ( xAk,yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xAk ][ yAk ] is equal to 1 and the reference index refIdxLX[ xAk ][ yAk ] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXA is set equal to 1 and the motion vector mvLXA is set equal to the motion vector mvLX[ xAk ][ yAk ], refIdxA is set equal to refIdxLX[ xAk ][ yAk ] and ListA is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xAk ][ yAk ] (with Y = !X) is equal to 1, AddPicId( RefPicListX[ refIdxLX ] ) is equal to AddPicId( RefPicListY[ refIdxLY[ xAk ][ yAk ] ] ), and PicOrderCnt( RefPicListY[ refIdxLY[ xAk ][ yAk ] ] ) is equal to PicOrderCnt( RefPicListX[ refIdxLX ] ), availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[ xAk ][ yAk ], refIdxA is set equal to refIdxLY[ xAk ][ yAk ] ,  ListA is set equal to ListY and mvLXA is set equal to mvLXA.
* When availableFlagLXA is equal to 1, availableFlagLXA is set to 0 if one or more of the following conditions are true:
  + One and only one of RefPicListX[ refIdxLX ] and ListA[ refIdxA ] is a long-term reference picture, the variable availableFlagLXA is set to 0;
  + Both RefPicListX[ refIdxLX ] and ListA[ refIdxA ] are long-term reference pictures and PicOrderCnt( ListA[ refIdxA ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ).

1. When availableFlagLXA is equal to 0, for ( xAk, yAk ) from ( xA0, yA0 ) to ( xA1, yA1 ) where yA1 = yA0 - MinPuSize, if availableFlagLXA is equal to 0, the following applies:

* If the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xAk ][ yAk ] is equal to 1, and AddPicId( RefPicListLX[ refIdxLX ] ) is equal to AddPicId( RefPicListLX[ refIdxLX[ xAk ][ yAk ] ] ), availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLX[ xAk ][ yAk ], refIdxA is set equal to refIdxLX[ xAk ][ yAk ], ListA is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xAk, yAk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xAk ][ yAk ] (with Y = !X) is equal to 1, and AddPicId( RefPicListLX[ refIdxLX ] ) is equal to AddPicId( RefPicListLY[ refIdxLY[ xAk ][ yAk ] ] ), availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[ xAk ][ yAk ], refIdxA is set equal to refIdxLY[ xAk ][ yAk ], ListA is set equal to ListY.
* When availableFlagLXA is equal to 1, availableFlagLXA is set to 0 if one or more of the following conditions are true:
  + One and only one of RefPicListX[ refIdxLX ] and ListA[ refIdxA ] is a long-term reference picture, the variable availableFlagLXA is set to 0;
  + Both RefPicListX[ refIdxLX ] and ListA[ refIdxA] are long-term reference pictures and PicOrderCnt( ListA[ refIdxA ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ).
* When availableFlagLXA is equal to 1, and both RefPicListA[ refIdxA ] and RefPicListX[ refIdxLX ] are short-term reference pictures, mvLXA is derived as specified below.

tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td (8‑126)

DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑127)

mvLXA = Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvLXA ) \*    
 ( (Abs( DistScaleFactor \* mvLXA ) + 127 ) >> 8 ) ) (8‑128)

where td and tb are derived as

td = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListA[ refIdxA ] ) ) (8‑129)

tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ) ) (8‑130)

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. Let a set of three sample location (xBk, yBk), with k = 0,1,2, specifies sample locations with xB0 = xP + nPSW, xB1 = xB0− MinPuSize , xB2 = xP − MinPuSize and yBk = yP − 1. The set of sample locations ( xBk, yBk ) represent the sample locations immediately to the upper side of the above partition boundary and its extended line. [Ed. (BB): Define MinPuSize in the SPS but the derivation should depend on the use of an AMP flag ]
2. When yP−1 is less than (( yC >> Log2CtbSize ) << Log2CtbSize), the following applies.

xB0 = (xB0>>3)<<3) + ((xB0>>3)&1)\*7 (8‑131)  
xB1 = (xB1>>3)<<3) + ((xB1>>3)&1)\*7 (8‑132)  
xB2 = (xB2>>3)<<3) + ((xB2>>3)&1)\*7 (8‑133)

1. Let the availability flag availableFlagLXB be initially set equal to 0 and the both components of mvLXB are set equal to 0.
2. For ( xBk, yBk ) from ( xB0, yB0 ) to ( xB2, yB2 ) where xB0 = xP + nPSW, xB1 = xB0 − MinPuSize , and xB2 =  xP − MinPuSize, if availableFlagLXB is equal to 0, the following applies:

* If the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xBk ][ yBk ] is equal to 1, and the reference index refIdxLX[ xBk ][ yBk ] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXB is set equal to 1 and the motion vector mvLXB is set equal to the motion vector mvLX[ xBk ][ yBk ], refIdxB is set equal to refIdxLX[ xBk ][ yBk ] and ListB is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xBk ][ yBk ] (with Y = !X) is equal to 1, AddPicId( RefPicListX[ refIdxLX ] ) is equal to AddPicId( RefPicListLY[ refIdxY[ xBk ][ yBk ] ] ), and PicOrderCnt( RefPicListY[ refIdxLY[ xBk ][ yBk ] ] ) is equal to PicOrderCnt( RefPicListX[ refIdxLX ] ), availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[ xBk ][ yBk ], refIdxB is set equal to refIdxLY[ xBk ][ yBk ],  and ListB is set equal to ListY.
* When availableFlagLXA is equal to 1, availableFlagLXA is set to 0 if one or more of the following conditions are true:
  + One and only one of RefPicListX[ refIdxLX ] and ListB[ refIdxB ] is a long-term reference picture;
  + Both RefPicListX[ refIdxLX ] and ListB[ refIdxB ] are long-term reference pictures and PicOrderCnt( ListB[ refIdxB ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ).

1. When isScaledFlagLX is equal to 0 and availableFlagLXB is equal to 1, mvLXA is set equal to mvLXB and refIdxA is set equal to refIdxB and availableFlagLXA is set equal to 1.
2. When isScaledFlagLX is equal to 0, availableFlagLXB is set equal to 0 and for ( xBk, yBk ) from ( xB0, yB0 ) to ( xB2, yB2 ) where xB0 = xP +nPSW, xB1 = xB0 - MinPuSize , and xB2 =  xP - MinPuSize, if availableFlagLXB is equal to 0, the following applies:

* If the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLX[ xBk ][ yBk ] is equal to 1, and AddPicId( RefPicListX[ refIdxLX ] ) is equal to AddPicId( RefPicListX[ refIdxLX[ xBk ][ yBk ] ] ), availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLX[ xBk ][ yBk ], refIdxB is set equal to refIdxLX[ xBk ][ yBk ], ListB is set equal to ListX.
* Otherwise, if the prediction unit covering luma location ( xBk, yBk ) is available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ], PredMode is not MODE\_INTRA, predFlagLY[ xBk ][ yBk ] (with Y = !X) is equal to 1, and AddPicId( RefPicListX[ refIdxLX ] ) is equal to AddPicId( RefPicListY[ refIdxLY[ xBk ][ yBk ] ] ), availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[ xBk ][ yBk ], refIdxB is set equal to refIdxLY[ xBk ][ yBk ], ListB is set equal to ListY.
* When availableFlagLXA is equal to 1, availableFlagLXA is set to 0 if one or more of the following conditions are true:
  + One and only one of RefPicListX[ refIdxLX ]  and ListB[ ( refIdxB ] ) is a long-term reference picture;
  + Both RefPicListX[ refIdxLX ] and ListB[ refIdxB ] are long-term reference pictures and PicOrderCnt( ListB[ refIdxB ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ).
* When availableFlagLXB is equal to 1 and PicOrderCnt( RefPicListB[ refIdxB ] ) is not equal to PicOrderCnt( RefPicListX[ refIdxLX ] ) and both RefPicListB[ refIdxB ] and RefPicListX[ refIdxLX ] are short-term reference pictures, mvLXB is derived as specified below.

tx = ( 16384 + ( Abs( td ) >> 1 ) ) / td (8‑134)

DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑135)

mvLXB =Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvLXA ) \*   
 ( (Abs( DistScaleFactor \* mvLXA ) + 127 ) >> 8 ) ) (8‑136)

[Ed. (GJS): I believe the thing that is being clipped is an integer, so does that make sense?]

where td and tb are derived as

td = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListB[ refIdxB ] ) ) (8‑137)

tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ) ) (8‑138)

## Changes in TMVP

This sub-section includes changes for both aspects of this proposal, highlighted in different colors.

8.5.2.1.7 Derivation process for temporal luma motion vector prediction

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
* the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are

* the motion vector prediction mvLXCol,
* the availability flag availableFlagLXCol.

The function RefPicOrderCnt( picX, refIdx, LX ) returns the picture order count PicOrderCntVal of the reference picture with index refIdx from reference picture list LX of the picture picX and is specified as follows.

RefPicOrderCnt( picX, refIdx, LX ) = PicOrderCnt(RefPicListX[ refIdx ] of the picture picX) (8 141)

Depending on the values of slice\_type, collocated\_from\_l0\_flag, and collocated\_ref\_idx, the variable colPic, specifying the picture that contains the collocated partition, is derived as follows.

* If slice\_type is equal to B and collocated\_from\_l0\_flag is equal to 0, the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList1[ collocated\_ref\_idx ].
* Otherwise (slice\_type is equal to B and collocated\_from\_l0\_flag is equal to 1 or slice\_type is equal to P) , the variable colPic specifies the picture that contains the collocated partition as specified by RefPicList0[ collocated\_ref\_idx ].

Variable colPu and its position ( xPCol, yPCol ) are derived in the following ordered steps:

1. The variable colPu is derived as follows

yPRb = yP + nPSH (8‑139)

* + If ( yP >> Log2CtbSize ) is equal to ( yPRb >> Log2CtbSize ), the horizontal component of the right-bottom luma position of the current prediction unit is defined by

xPRb = xP + nPSW (8‑140)

and the variable colPu is set as the prediction unit covering the modified position given by ( ( xPRb >> 4 ) << 4, ( yPRb >> 4 ) << 4 ) inside the colPic.

* + Otherwise ( ( yP >> Log2CtbSize ) is not equal to ( yPRb >> Log2CtbSize ) ), colPu is marked as "unavailable".

1. When colPu is coded in an intra prediction mode or colPu is marked as "unavailable", the following applies.
   * Central luma position of the current prediction unit is defined by

xPCtr = ( xP + ( nPSW >> 1 ) (8‑141)

yPCtr = ( yP + ( nPSH >> 1 ) (8‑142)

* + The variable colPu is set as the prediction unit covering the modified position given by ( ( xPCtr >> 4 ) << 4, ( yPCtr >> 4 ) << 4 ) inside the colPic.

1. ( xPCol, yPCol ) is set equal to the top-left luma sample of the colPu relative to the top-left luma sample of the colPic.

The function LongTermRefPic( picX, refIdx, LX ) is defined as follows. If the reference picture with index refIdx from reference picture list LX of the picture picX was marked as "used for long term reference" at the time when picX was the current picture, LongTermRefPic( picX, refIdx, LX ) returns 1; otherwise LongTermRefPic( picX, refIdx, LX ) returns 0.

The variables mvLXCol and availableFlagLXCol are derived as follows.

* If one or more of the following conditions are true, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
  + colPu is coded in an intra prediction mode.
  + colPu is marked as "unavailable".
  + pic\_temporal\_mvp\_enable\_flag is equal to 0.
* Otherwise, the motion vector mvCol, the reference index refIdxCol, and the reference list identifier listCol are derived as follows.
  + If PredFlagL0[ xPCol ][ yPCol ] is equal to 0, mvCol, refIdxCol, and listCol are set equal to MvL1[ xPCol ][ yPCol ], RefIdxL1[ xPCol ][ yPCol ], and L1, respectively.
  + Otherwise (PredFlagL0[ xPCol ][ yPCol ] is equal to 1), the following applies.
  + If PredFlagL1[ xPCol ][ yPCol ] is equal to 0, mvCol, refIdxCol, and listCol are set equal to MvL0[ xPCol ][ yPCol ], RefIdxL0[ xPCol ][ yPCol ], and L0, respectively.
  + Otherwise (PredFlagL1[ xPCol ][ yPCol ] is equal to 1), the following assignments are made.
    - * If PicOrderCnt( pic ) of every picture pic in every reference picture lists is less than or equal to PicOrderCntVal, mvCol, refIdxCol, and listCol are set equal to MvLX[ xPCol ][ yPCol ], RefIdxLX[ xPCol ][ yPCol ] and LX, respectively with X being the value of X this process is invoked for.
      * Otherwise (PicOrderCnt( pic ) of at least one picture pic in at least one reference picture list is greater than PicOrderCntVal, mvCol, refIdxCol and listCol are set equal to MvLN[ xPCol ][ yPCol ], RefIdxLN[ xPCol ][ yPCol ] and LN, respectively with N being the value of collocated\_from\_l0\_flag.
  + If one of the following conditions is true, the variable availableFlagLXCol is set equal to 0.
    - * AddPicId( RefPicListX[ refIdxLX ] ) is not equal to AddPicId( colPic, refIdxCol, listCol );
      * RefPicListX[ refIdxLX ] is a short-term reference picture and LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 1;
      * RefPicListX[ refIdxLX ] is a long-term reference picture and LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 0;
      * RefPicListLX[ refIdxLX ] is a long-term reference picture, LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 1 and RefPicOrderCnt( colPic, refIdxCol, listCol ) is not equal to PicOrderCnt( RefPicListLX [ refIdxLX ] ).
  + Otherwise, the variable availableFlagLXCol is set equal to 1, and the following applies.
  + If RefPicListX[ refIdxLX ] is a long-term reference picture, or LongTermRefPic( colPic, refIdxCol, listCol ) is equal to 1, or PicOrderCnt( colPic ) – RefPicOrderCnt( colPic, refIdxCol, listCol ) is equal to PicOrderCntVal – PicOrderCnt( RefPicListX[ refIdxLX ] ),  
     mvLXCol = mvCol (8‑143)
  + Otherwise, mvLXCol is derived as scaled version of the motion vector mvCol as specified below  
     tx = ( 16384 + ( Abs( td ) >>1 ) ) / td (8‑144)
  + DistScaleFactor = Clip3( −4096, 4095, ( tb \* tx + 32 ) >> 6 ) (8‑145)

mvLXCol =  Clip3( −8192, 8191.75, Sign( DistScaleFactor \* mvCol ) \*   
 ( (Abs( DistScaleFactor \* mvCol ) + 127 ) >> 8 ) ) (8‑146)

where td and tb are derived as  
td = Clip3( −128, 127, PicOrderCnt( colPic ) – RefPicOrderCnt( colPic, refIdxCol, listCol ) ) (8‑147)  
tb = Clip3( −128, 127, PicOrderCntVal – PicOrderCnt( RefPicListX [ refIdxLX ] ) ) (8‑148)

## Changes in merge

8.5.2.1.3 Derivation process for combined bi-predictive merging candidates

Inputs of this process are

* a merging candidate list mergeCandList,
* reference indices refIdxL0N and refIdxL1N of every candidate N being in mergeCandList,
* prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N being in mergeCandList,
* motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList,
* the number of elements numMergeCand within mergeCandList,
* the number of elements numOrigMergeCand within the mergeCandList after the spatial and temporal merge candidate derivation process,

Outputs of this process are

* the merging candidate list mergeCandList,
* the number of elements numMergeCand within mergeCandList.
* reference indices refIdxL0combCandk and refIdxL1combCandk of every new candidate combCandk being added in mergeCandList during the invokation of this process,
* prediction list utilization flags predFlagL0combCandk and predFlagL1combCandk of every new candidate combCandk being added in mergeCandList during the invokation of this process,
* motion vectors mvL0combCandk and mvL1combCandk of every new candidate combCandk being added in mergeCandList during the invokation of this process,

When numOrigMergeCand is greater than 1 and less than MaxNumMergeCand, the variable numInputMergeCand is set to numMergeCand, the variables combIdx and combCnt are set to 0, the variable combStop is set to FALSE and the following steps are repeated until combStop is equal to TRUE.

1. The variables l0CandIdx and l1CandIdx are derived using combIdx as specified in Table 8‑8.
2. The following assignments are made with l0Cand being the candidate at position l0CandIdx and l1Cand being the candidate at position l1CandIdx in the merging candidate list mergeCandList ( l0Cand = mergeCandList[ l0CandIdx ] , l1Cand = mergeCandList[ l1CandIdx ] ).
3. When all of the following conditions are true,
   * + predFlagL0l0Cand = = 1
     + predFlagL1l1Cand = = 1
     + AddPicId( RefPicListL0[ refIdxL0l0Cand ] ) != AddPicId( RefPicListL1[ refIdxL1l1Cand ] ) | | PicOrderCnt( RefPicList0[ refIdxL0l0Cand ] ) != PicOrderCnt( RefPicList1[ refIdxL1l1Cand ] ) | | mvL0l0Cand != mvL1l1Cand

the following applies.

* + - …

1. …
2. …

# 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).**