#### 7.3.8.7 Prediction unit syntax

|  |  |
| --- | --- |
| prediction\_unit( x0, y0, nPbW, nPbH ) { | Descriptor |
| if( cu\_skip\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { /\* MODE\_INTER \*/ |  |
| **merge\_flag**[ x0 ][ y0 ] | ae(v) |
| if( merge\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx**[ x0 ][ y0 ] | ae(v) |
| } else { |  |
| **pu\_imv\_flag**[ x0 ][ y0 ] | ae(v) |
| if( slice\_type = = B ) |  |
| **inter\_pred\_idc**[ x0 ][ y0 ] | ae(v) |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L1 ) { |  |
| if( num\_ref\_idx\_l0\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l0**[ x0 ][ y0 ] | ae(v) |
| mvd\_coding( x0, y0, 0 ) |  |
| if( !pu\_imv\_flag[ x0 ][ y0 ]) { |  |
| **mvp\_l0\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } |  |
| if( inter\_pred\_idc[ x0 ][ y0 ] != PRED\_L0 ) { |  |
| if( num\_ref\_idx\_l1\_active\_minus1 > 0 ) |  |
| **ref\_idx\_l1**[ x0 ][ y0 ] | ae(v) |
| if( mvd\_l1\_zero\_flag &&   inter\_pred\_idc[ x0 ][ y0 ] = = PRED\_BI ) { |  |
| MvdL1[ x0 ][ y0 ][ 0 ] = 0 |  |
| MvdL1[ x0 ][ y0 ][ 1 ] = 0 |  |
| } else |  |
| mvd\_coding( x0, y0, 1 ) |  |
| if( !pu\_imv\_flag[ x0 ][ y0 ]) { |  |
| **mvp\_l1\_flag**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**pu\_imv\_flag**[ x0 ][ y0 ] equal to 1 specifies that motion vectors of the prediction unit are in integer-pixel precision and that mvp\_lX\_flag[ x0 ][ y0 ] is set equal to 0. pu\_imv\_flag[ x0 ][ y0 ] equal to 0 specifies that motion vectors of the prediction units belonging to the current coding unit are in quarter-pixel precision. When pu\_imv\_flag[ x0 ][ y0 ] is not present, it is inferred to be equal to 0.

#### 7.4.9.9 Motion vector difference semantics

**abs\_mvd\_greater0\_flag[** compIdx **]** specifies whether the absolute value of a motion vector component difference is greater than 0.

**abs\_mvd\_greater1\_flag[** compIdx **]** specifies whether the absolute value of a motion vector component difference is greater than 1.

When abs\_mvd\_greater1\_flag[ compIdx ] is not present, it is inferred to be equal to 0.

**abs\_mvd\_minus2[** compIdx **]** plus 2 specifies the absolute value of a motion vector component difference.

When abs\_mvd\_minus2[ compIdx ] is not present, it is inferred to be equal to −1.

**mvd\_sign\_flag[** compIdx **]** specifies the sign of a motion vector component difference as follows:

* If mvd\_sign\_flag[ compIdx ] is equal to 0, the corresponding motion vector component difference has a positive value.
* Otherwise (mvd\_sign\_flag[ compIdx ] is equal to 1), the corresponding motion vector component difference has a negative value.

When mvd\_sign\_flag[ compIdx ] is not present, it is inferred to be equal to 0.

When pu\_imv\_flag[ x0 ][ y0 ] is equal to 0, the motion vector difference lMvd[ compIdx ] for compIdx = 0..1 is derived as follows:

lMvd[ compIdx ] = abs\_mvd\_greater0\_flag[ compIdx ] \*  
 ( abs\_mvd\_minus2[ compIdx ] + 2 ) \* ( 1 − 2 \* mvd\_sign\_flag[ compIdx ] ) (7‑65)

When pu\_imv\_flag[ x0 ][ y0 ] is equal to 1, the motion vector difference lMvd[ compIdx ] for compIdx = 0..1 is derived as follows:

lMvd[ compIdx ] = abs\_mvd\_greater0\_flag[ compIdx ] \*  
 (( abs\_mvd\_minus2[ compIdx ] + 2 ) << 2) \* ( 1 − 2 \* mvd\_sign\_flag[ compIdx ] ) (7‑65)

The variable MvdLX[ x0 ][ y0 ][ compIdx ], with X being 0 or 1, specifies the difference between a list X vector component to be used and its prediction. The value of MvdLX[ x0 ][ y0 ][ compIdx ] shall be in the range of −215 to 215 − 1, inclusive. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

The variable BvdIntra[ x0 ][ y0 ][ compIdx ] specifies the difference between a vector component to be used for the intra block copying prediction mode and its prediction. The value of BvdIntra[ x0 ][ y0 ][ compIdx ] shall be in the range of −128 to 128, inclusive. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

* If refList is equal to 0, MvdL0[ x0 ][ y0 ][ compIdx ] is set equal to lMvd[ compIdx ] for compIdx = 0..1.
* Otherwise (refList is equal to 1), MvdL1[ x0 ][ y0 ][ compIdx ] is set equal to lMvd[ compIdx ] for compIdx = 0..1.
* Otherwise (refList is equal to 2), BvdIntra[ x0 ][ y0 ][ compIdx ] is set equal to lMvd[ compIdx ] for compIdx = 0..1.

…

When mvd\_sign\_flag[ compIdx ] is not present, it is inferred to be equal to 0.

The motion vector difference lMvd[ compIdx ] for compIdx = 0..1 is derived as follows:

lMvd[ compIdx ] = abs\_mvd\_greater0\_flag[ compIdx ] \*  
 ( abs\_mvd\_minus2[ compIdx ] + 2 ) \* ( 1 − 2 \* mvd\_sign\_flag[ compIdx ] ) (7‑65)

When pu\_imv\_flag[ x0 ][ y0 ] is equal to 1, the variable lMvd[ compIdx ] for compIdx = 0..1 is modified as follows:

lMvd[ compIdx ] = lMvd[ compIdx ] << 2 (7‑66)

The variable MvdLX[ x0 ][ y0 ][ compIdx ], with X being 0 or 1, specifies the difference between a list X vector component to be used and its prediction. The value of MvdLX[ x0 ][ y0 ][ compIdx ] shall be in the range of −215 to 215 − 1, inclusive. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

..

##### Derivation process for luma motion vector prediction

Inputs to this process are:

* a luma location ( xCb, yCb ) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* a luma location ( xPb, yPb ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* the reference index of the current prediction unit partition refIdxLX, with X being 0 or 1,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Output of this process is the prediction mvpLX of the motion vector mvLX, with X being 0 or 1.

The motion vector predictor mvpLX is derived in the following ordered steps:

1. The derivation process for motion vector predictor candidates from neighbouring prediction unit partitions in subclause 8.5.3.2.6 is invoked with the luma coding block location ( xCb, yCb ), the coding block size nCbS, the luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, refIdxLX, with X being 0 or 1, and the partition index partIdx as inputs, and the availability flags availableFlagLXN and the motion vectors mvLXN, with N being replaced by A or B, as output.
2. If both availableFlagLXA and availableFlagLXB are equal to 1 and mvLXA is not equal to mvLXB, availableFlagLXCol is set equal to 0. Otherwise, the derivation process for temporal luma motion vector prediction in subclause 8.5.3.2.7 is invoked with luma prediction block location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and refIdxLX, with X being 0 or 1, as inputs, and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.
3. The motion vector predictor candidate list, mvpListLX, is constructed as follows:

i = 0  
if( availableFlagLXA )  
 mvpListLX[ i++ ] = mvLXA  
if( availableFlagLXB )  
 mvpListLX[ i++ ] = mvLXB (8‑143)  
if( availableFlagLXCol )  
 mvpListLX[ i++ ] = mvLXCol

1. The motion vector predictor list is modified as follows:
   * + When mvLXA and mvLXB have the same value, mvLXB is removed from the list and the variable numMvpCandLX is set equal to the number of elements within the mvpListLX.
     + When numMvpCandLX is less than 2, the following applies repeatedly until numMvpCandLX is equal to 2:

mvpListLX[ numMvpCandLX ][ 0 ] = 0 (8‑144)

mvpListLX[ numMvpCandLX ][ 1 ] = 0 (8‑145)

numMvpCandLX = numMvpCandLX + 1 (8‑146)

* + - When numMvpCandLX is greater than 2, all motion vector predictor candidates mvpListLX[ idx ] with idx greater than 1 are removed from the list.

1. The motion vector of mvpListLX[ mvp\_lX\_flag[ xPb ][ yPb ] ] is assigned to mvpLX.

When pu\_imv\_flag[ x0 ][ y0 ] is equal to 1 and X is equal to 0, or when mvd\_l1\_zero\_flag is set to 1 and inter\_pred\_idc[ x0 ][ y0 ] is set equal to PRED\_BI, the variable mvpListL0[ 0 ][ compIdx] for compIdx = 0..1 is modified as follows:

mvpListL0[ 0 ][ compIdx] = (((Abs(mvpListL0[ 0 ][ compIdx]) + 2 ) >> 2) << 2)\* Sign(mvpListL0[ 0 ][ compIdx])

#### 9.3.2.2 Initialization process for context variables

Table 9‑4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **ctxTable** | **initType** | | |
| **0** | **1** | **2** |
| … | | | | | |
| prediction\_unit( ) | merge\_flag[ ][ ] | Table 9‑15 |  | 0 | 1 |
| merge\_idx[ ][ ] | Table 9‑16 |  | 0 | 1 |
| pu\_imv\_flag[ ][ ] | Table 9-17 |  | 0 | 1 |
| inter\_pred\_idc[ ][ ] | Table 9‑17 |  | 0..4 | 5..9 |
| ref\_idx\_l0[ ][ ], ref\_idx\_l1[ ][ ] | Table 9‑18 |  | 0..1 | 2..3 |
| mvp\_l0\_flag[ ][ ], mvp\_l1\_flag[ ][ ] | Table 9‑19 |  | 0 | 1 |
| transform\_tree( ) | split\_transform\_flag[ ][ ][ ] | Table 9‑20 | 0..2 | 3..5 | 6..8 |
| cbf\_luma[ ][ ][ ] | Table 9‑21 | 0..1 | 2..3 | 4..5 |
| cbf\_cb[ ][ ][ ], cbf\_cr[ ][ ][ ] | Table 9‑22 | 0..3 | 4..7 | 8..11 |
| mvd\_coding( ) | abs\_mvd\_greater0\_flag[ ] | Table 9‑23 |  | 0 | 2 |
| abs\_mvd\_greater1\_flag[ ] | Table 9‑23 |  | 1 | 3 |
| …. | | | | | |

Table 9‑17 – Values of initValue for ctxIdx of pu\_imv\_flag

|  |  |  |
| --- | --- | --- |
| **Initialization variable** | **ctxIdx of** pu\_imv\_flag | |
| **0** | **1** |
| **initValue** | 154 | 154 |

### 

### Binarization process

| Table 9‑34 – Syntax elements and associated binarizations | | | |
| --- | --- | --- | --- |
| **Syntax structure** | **Syntax element** | **Binarization** | |
| **Process** | **Input parameters** |
| prediction\_unit( ) | merge\_flag[ ][ ] | FL | cMax = 1 |
| merge\_idx[ ][ ] | TR | cMax = MaxNumMergeCand − 1, cRiceParam = 0 |
| inter\_pred\_idc[ x0 ][ y0 ] | 9.3.3.7 | nPbW, nPbH |
| pu\_imv\_flag[ ][ ] | FL | cMax = 1 |
| ref\_idx\_l0[ ][ ] | TR | cMax = num\_ref\_idx\_l0\_active\_minus1, cRiceParam = 0 |
| mvp\_l0\_flag[ ][ ] | FL | cMax = 1 |
| ref\_idx\_l1[ ][ ] | TR | cMax = num\_ref\_idx\_l1\_active\_minus1, cRiceParam = 0 |
| mvp\_l1\_flag[ ][ ] | FL | cMax = 1 |