Test2: VSP insertion control + IC&ARP exclusiveness + VSP inheritance control

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { | **Descriptor** |
| if( transquant\_bypass\_enable\_flag ) { |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| } |  |
| if( slice\_type != I ) |  |
| **skip\_flag**[ x0 ][ y0 ] | ae(v) |
| if( skip\_flag[ x0 ][ y0 ] ) { |  |
| prediction\_unit( x0, y0, log2CbSize ) |  |
| if ( iv\_res\_pred\_flag[ nuh\_layer\_id ] && TempRefPicInListsFlag ) |  |
| **iv\_res\_pred\_weight\_idx** | ae(v) |
| if ( icEnableFlag && iv\_res\_pred\_weight\_idx == 0 ) |  |
| **ic\_flag** | ae(v) |
| } |  |
| else { |  |
| nCbS = ( 1 << log2CbSize ) |  |
| if( slice\_type != I ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( ( PredMode[ x0 ][ y0 ] ! = MODE\_INTRA | | log2CbSize = = Log2MinCbSize ) &&  !predPartModeFlag) |  |
| **part\_mode** | ae(v) |
| if( PredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| … |  |
| } else { |  |
| if( PartMode = = PART\_2Nx2N ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| if ( iv\_res\_pred\_flag[ nuh\_layer\_id ] && TempRefPicInListsFlag ) |  |
| **iv\_res\_pred\_weight\_idx** | ae(v) |
| } else if( PartMode = = PART\_2NxN ) { |  |
| … |  |
| } |  |
| } |  |
| if ( icEnableFlag && iv\_res\_pred\_weight\_idx == 0) |  |
| **ic\_flag** | ae(v) |

* + - * 1. Derivation process for luma motion vectors for merge mode

This process is only invoked when merge\_flag[ xPb ][ yPb ] is equal to 1, where ( xPb, yPb ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

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 luma location ( xPb, yPb ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCbS specifying the size of the current luma coding block,
* two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

* the luma motion vectors mvL0 and mvL1,
* the reference indices refIdxL0 and refIdxL1,
* the prediction list utilization flags predFlagL0 and predFlagL1,
* the disparity vector availability flags ivpMvFlagL0 and ivpMvFlagL1,
* the flag vspModeFlag, specifying, whether the current PU is coded using view synthesis prediction,

The location ( xOrigP, yOrigP ) and the variables nOrigPbW and nOrigPbH are derived to store the values of ( xPb, yPb ), nPbW, and nPbH as follows:

* + - 1. ( xOrigP, yOrigP ) is set equal to ( xPb, yPb ) (‑81)
      2. nOrigPbW = nPbW (‑82)
      3. nOrigPbH = nPbH (‑83)

When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, ( xPb, yPb ), nPbW, nPbH, and partIdx are modified as follows:

* + - 1. ( xPb, yPb ) = ( xCb, yCb ) (‑84)
      2. nPbW = nCbS (‑85)
      3. nPbH = nCbS (‑86)
      4. partIdx = 0 (‑87)

NOTE – When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

* 1. The derivation process for merging candidates from neighbouring prediction unit partitions in subclause 8.5.3.2.2 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, and the partition index partIdx as inputs, and the output being the availability flags availableFlagA0, availableFlagA1, availableFlagB0, availableFlagB1, and availableFlagB2, the reference indices refIdxLXA0, refIdxLXA1, refIdxLXB0, refIdxLXB1, and refIdxLXB2, the prediction list utilization flags predFlagLXA0, predFlagLXA1, predFlagLXB0, predFlagLXB1, and predFlagLXB2, and the motion vectors mvLXA0, mvLXA1, mvLXB0, mvLXB1, and mvLXB2, with X being 0 or 1.
  2. The reference indices for the temporal merging candidate, refIdxLXCol, with X being 0 or 1, are set equal to 0.
  3. The derivation process for temporal luma motion vector prediction in subclause is invoked with the luma location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the variable refIdxL0Col as inputs, and the output being the availability flag availableFlagL0Col and the temporal motion vector mvL0Col.The variables availableFlagCol, predFlagL0Col and predFlagL1Col are derived as follows:
     + 1. availableFlagCol = availableFlagL0Col (‑88)
       2. predFlagL0Col = availableFlagL0Col (‑89)
       3. predFlagL1Col = 0 (‑90)
  4. When slice\_type is equal to B, the derivation process for temporal luma motion vector prediction in subclause is invoked with the luma location ( xPb, yPb ), the luma prediction block width nPbW, the luma prediction block height nPbH, and the variable refIdxL1Col as inputs, and the output being the availability flag availableFlagL1Col and the temporal motion vector mvL1Col. The variables availableFlagCol and predFlagL1Col are derived as follows:
     + 1. availableFlagCol = availableFlagL0Col  | |  availableFlagL1Col (‑91)
       2. predFlagL1Col = availableFlagL1Col (‑92)
  5. Depending on iv\_mv\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 0, the flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC are set equal to 0.
     + Otherwise (iv\_mv\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for the inter-view merge candidates as specified in subclause is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH, as the inputs and the output is assigned to the availability flags availableFlagIvMC, availableIvMCShift and availableFlagIvDC, the reference indices refIdxLXIvMC, refIdxLXIvMCShift and refIdxLXIvDC, the prediction list utilization flags predFlagLXIvMC, predFlagLXivMCShift and predFlagLXIvDC, and the motion vectors mvLXIvMC, mvLXIvMCShift and mvLXIvDC (with X being 0 or 1, respectively)..
  6. Depending on view\_synthesis\_pred\_flag[ nuh\_layer\_id ], the following applies.
     + If view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 0, the flag availableFlagVSP is set equal to 0.
     + Otherwise (view\_synthesis\_pred\_flag[ nuh\_layer\_id ] is equal to 1), the derivation process for a view synthesis prediction merge candidate as specified in subclause is invoked with the luma locations ( xCb, yCb ) as input and the outputs are the availability flag availableFlagVSP, the reference indices refIdxL0VSP and refIdxL1VSP, the prediction list utilization flags predFlagL0VSP and predFlagL1VSP, and the motion vectors mvL0VSP and mvL1VSP.
  7. Depending on DepthFlag, the following applies.
     + If DepthFlag is equal to 0, the variable availableFlagT is set equal to 0.
     + Otherwise ( DepthFlag is equal to 1), the derivation process for the texture merging candidate as specified in subclause is invoked with the luma location ( xPb, yPb ), the variables nPbW and nPbH as the inputs and the outputs are the flag availableFlagT, the prediction utilization flags predFlagL0T and predFlagL1T, the reference indices refIdxL0T and refIdxL1T, and the motion vectors mvL0T and mvL1T.
  8. The merge candidate lists mergeCandList and mergeCandIsVspFlag are constructed as specified by the following ordered steps:
  9. The variable numMergeCand is set equal to 0.
  10. When availableFlagT is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to T, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  11. When availableFlagIvMC is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvMC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  12. When availableFlagA1 is equal to 1, the following applies:
      + - When the following condition is true,
          * availableFlagT = = 0 && availableFlagIvMC  = = 0,
          1. or one or more of the following conditions are true, with N being replaced by T and IvMC:
          * availableFlagN = = 1 && predFlagLXN  !=  predFlagLXA1, (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && mvLXN  !=  mvLXA1 (with X being replaced by 0 and 1),
          * availableFlagN = = 1 && refIdxLXN  !=  refIdxLXA1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to A1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb + nPbH − 1 ] && !ic\_flag && !iv\_res\_pred\_weight\_idx and the variable numMergeCand is increased by 1.

* 1. When availableFlagB1 is equal to 1, the following applies:
     + - When the following condition is true ,
         * availableFlagT = = 0 && availableFlagIvMC = = 0,

or one or more of the following conditions is true, with N being replaced by T and IvMC:

* + - * + availableFlagN = = 1 && predFlagLXN  !=  predFlagLXB1, (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && mvLXN  !=  mvLXB1 (with X being replaced by 0 and 1),
        + availableFlagN = = 1 && refIdxLXN  !=  refIdxLXB1 (with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to B1, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb + nPbW − 1 ][ yPb − 1 ] && !ic\_flag && !iv\_res\_pred\_weight\_idx and the variable numMergeCand is increased by 1.

* 1. When availableFlagB0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to B0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb + nPbW ][ yPb − 1 ] && !ic\_flag && !iv\_res\_pred\_weight\_idx and the variable numMergeCand is increased by 1.
  2. When availableFlagIvDC is equal to 1, and one or more of the following conditions is true,
     + - availableFlagA1  = =  0,
       - predFlagLXA1  !=  predFlagLXIvDC, (with X being replaced by 0 and 1),
       - mvLXA1  !=  mvLXIvDC(with X being replaced by 0 and 1),
       - refIdxLXA1  !=  refIdxLXIvDC(with X being replaced by 0 and 1),

and one or more of the following conditions is true,

* + - * availableFlagB1  = =  0,
      * predFlagLXB1  !=  predFlagLXIvDC, (with X being replaced by 0 and 1),
      * mvLXB1  !=  mvLXIvDC(with X being replaced by 0 and 1),
      * refIdxLXB1  !=  refIdxLXIvDC(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvDC, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. When availableFlagVSP is equal to 1 and ic\_flag is equal to 0 and && !iv\_res\_pred\_weight\_idx is equal to 0, the entry mergeCandList[ numMergeCand ] is set equal to VSP, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal 1 and the variable numMergeCand is increased by 1.
  2. When availableFlagA0 is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to A0, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb + nPbH ] && !ic\_flag && !iv\_res\_pred\_weight\_idx and the variable numMergeCand is increased by 1.
  3. When availableFlagB2 is equal to 1 and numMergeCand is less than 4 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to B2, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to VspModeFlag[ xPb − 1 ][ yPb − 1 ] && !ic\_flag && !iv\_res\_pred\_weight\_idx and the variable numMergeCand is increased by 1.
  4. When availableFlagIvMCShift is equal to 1 and numMergeCand is less than 6, and one or more of the following conditions are true,
     + - availableFlagIvMC  = =  0,
       - predFlagLXMC  !=  predFlagLXMCShift(with X being replaced by 0 and 1),
       - mvLXMC  !=  mvLXIvMCShift(with X being replaced by 0 and 1),
       - refIdxLXMC  !=  refIdxLXMCShift(with X being replaced by 0 and 1),

the entry mergeCandList[ numMergeCand ] is set equal to IvMCShift, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. A variable availableFlagIvDCShift is set to 0 and when all of the following conditions are true
     + - DepthFlag is equal to 0, [Ed (CY): this condition is already assumed thus could be removed.]
       - availableFlagIvMCShift is equal to 0,
       - numMergeCand is less than 6,

the derivation process for the shifted disparity merging candidate as specified in subclause is invoked with the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N, of every candidate N being in mergeCandList, mergeCandList, mergeCandIsVspFlag, and numMergeCand as the inputs and the outputs are the flag availableFlagIvDCShift, the prediction utilization flags predFlagL0IvDCShift and predFlagL1IvDCShift, the reference indices refIdxL0IvDCShift and refIdxL1IvDCShift, and the motion vectors mvL0IvDCShift and mvL1IvDCShift. When availableFlagIvDCShift is equal to 1, the entry mergeCandList[ numMergeCand ] is set equal to IvDCShift, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.

* 1. When availableFlagCol is equal to 1 and numMergeCand is less than 5 + iv\_mv\_pred\_flag[ nuh\_layer\_id ] + DepthFlag, the entry mergeCandList[ numMergeCand ] is set equal to Col, the entry mergeCandIsVspFlag[ numMergeCand ] is set equal to 0 and the variable numMergeCand is increased by 1.
  2. The variable numOrigMergeCand is set equal to numMergeCand.
  3. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause is invoked with mergeCandList, mergeCandIsVspFlag, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, numCurrMergeCand, and numOrigMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0combCandk and refIdxL1combCandk, the prediction list utilization flags predFlagL0combCandk and predFlagL1combCandk, and the motion vectors mvL0combCandk and mvL1combCandk of every new candidate combCandk being added into mergeCandList. The number of candidates being added, numCombMergeCand, is set equal to ( numCurrMergeCand − numOrigMergeCand ). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive, and mergeCandIsVspFlag[ numOrigMergeCand + k ] is set equal to 0.
  4. The derivation process for zero motion vector merging candidates specified in subclause 8.5.3.2.4 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, and numCurrMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0zeroCandm and refIdxL1zeroCandm, the prediction list utilization flags predFlagL0zeroCandm and predFlagL1zeroCandm, and the motion vectors mvL0zeroCandm and mvL1zeroCandm of every new candidate zeroCandm being added into mergeCandList. The number of candidates being added, numZeroMergeCand, is set equal to ( numCurrMergeCand − numOrigMergeCand − numCombMergeCand ). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand − 1, inclusive, and mergeCandIsVspFlag[ numOrigMergeCand + numCombMergeCand + m ] is set equal to 0.
  5. The following assignments are made with N being the candidate at position merge\_idx[ xOrigP ][ yOrigP ] in the merging candidate list mergeCandList ( N = mergeCandList[ merge\_idx[ xOrigP ][ yOrigP ] ] ) and X being replaced by 0 or 1:
     + 1. mvLX[ 0 ] = mvLXN[ 0 ] (‑93)
       2. mvLX[ 1 ] = mvLXN[ 1 ] (‑94)
       3. refIdxLX = refIdxLXN (‑95)
       4. predFlagLX = predFlagLXN (‑96)
  6. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1, and ( nOrigPbW + nOrigPbH ) is equal to 12, the following applies
     + 1. refIdxL1 = −1 (‑97)
       2. predFlagL1 = 0 (‑98)
  7. The variable vspModeFlag is set equal to mergeCandIsVspFlag[ merge\_idx[ xPb][ yPb ] ]
  8. When vspModeFlag is equal to 1, the following applies:
     + The variables xVsp and yVsp are derived depending on the value of mergeCandList[ merge\_idx[ xPb][ yPb ] ] as specified in .
     + The variable vspRefViewIdx is set equal RefViewIdx[ xVsp ][ yVsp ] and the variable vspMvDisp is set equal to MvDisp[ xVsp ][ yVsp ].
     + For x in the range of xPb to ( xPb + nPbW − 1 ), inclusive, the following applies:
       - For y in the range of yPb to ( yPb + nPbH − 1 ), inclusive, the following applies:
         1. RefViewIdx[ x ][ y ] = vspRefViewIdx (‑99)
         2. MvDisp[ x ][ y ] = vspMvDisp (‑100)
  9. The disparity availability flag ivpMvFlagLX is derived as follows (with X being replace by 0 or 1).
     + If one of the following conditions are true, ivpMvFlagLX is set equal to 1

[Ed. (GT) There is some redundancy in draft and software since disparities equal for both lists.(#7) ]

* + - * predFlagLXIvMC  = =  1 && mergeCandList[ merge\_idx[ xPb][ yPb ] ]  =  =  IvMC
      * predFlagLXIvMCShift  = =  1 && and mergeCandList[ merge\_idx[ xPb][ yPb ] ]  = =  IvMCShift

[Ed. (GT): PredMode[ xCb ][ yCb ]  = =  MODE\_SKIP might be added here instead of testing it in the disparity vector derivation process]

* + - Otherwise, ivpMvFlagLX is set equal to 0.

Table ‑9 – Specification of the xVsp and yVsp depending on N = mergeCandList[ merge\_idx[ xPb][ yPb ] ]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **N** | A1 | B1 | B0 | VSP | A0 | B2 |
| **xVsp** | xPb − 1 | xPb + nPbW − 1 | xPb + nPbW | xPb | xPb − 1 | xPb − 1 |
| **yVsp** | yPb + nPbH − 1 | yPb − 1 | yPb − 1 | yPb | yPb + nPbH | yPb − 1 |