**8.5.2.1.1 Derivation process for luma motion vectors for merge mode**

This process is only invoked when PredMode is equal to MODE\_SKIP or PredMode is equal to MODE\_INTER and merge\_flag [ xP ][ yP ] is equal to 1, where ( xP, yP ) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs of this process are

* a luma location ( xC, yC ) 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 ( xP, yP ) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable nCS specifying the size of the current luma coding block,
* variables specifying the width and the height of the luma prediction block, nPSW and nPSH,
* 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 variable spatialCandCnt is set equal to 0.

The variables singleMCLFlag is derived as follows.

* If log2\_parallel\_merge\_level\_minus2 is greater than 0 and nCS is equal to 8, singleMCLFlag is set to 1.
* Otherwise, singleMCLFlag is set to 0.

When singleMCLFlag is equal to 1, xP is set equal to xC, yP is set equal to yC, and both nPSW and nPSH are set equal to nCS.

NOTE – When singleMCLFlag is equal to 1, 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 as specified by the following ordered steps:

1. The derivation process for merging candidates from neighboring prediction unit partitions in subclause 8.5.2.1.2 is invoked with luma location ( xP, yP ), the variable spatialCandCnt, the variable singleMCLFlag, the width and the height of the luma prediction block nPSW and nPSH and the partition index partIdx as inputs and the output is assigned to the availability flags availableFlagN, the variable spatialCandCnt, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N and the motion vectors mvL0N and mvL1N with N being replaced by A0, A1, B0, B1 or B2.
2. The reference index for temporal merging candidate refIdxLX is set equal to 0.
3. The derivation process for temporal luma motion vector prediction in subclause 8.5.2.1.7 is invoked with luma location ( xP, yP ), the width and the height of the luma prediction block nPSW and nPSH, and refIdxLX as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector mvLXCol. The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.

availableFlagCol = availableFlagL0Col | | availableFlagL1Col (8‑81)

predFlagLXCol = availableFlagLXCol (8‑82)

1. The merging candidate list, mergeCandList, is constructed as follows.
   1. A1, if availableFlagA1 is equal to 1
   2. B1, if availableFlagB1 is equal to 1
   3. B0, if availableFlagB0 is equal to 1
   4. A0, if availableFlagA0 is equal to 1
   5. B2, if availableFlagB2 is equal to 1
   6. Col, if availableFlagCol is equal to 1
2. The variable numMergeCand ~~and numOrigMergeCand are~~is set to the number of merging candidates in the mergeCandList and the variable numOrigMergeCand is set equal to spatialCandCnt.
3. When slice\_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in subclause 8.5.2.1.3 is invoked with 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 being in mergeCandList, numMergeCand and numOrigMergeCand given as input and the output is assigned to mergeCandList, numMergeCand, 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 in mergeCandList. The number of candidates being added numCombMergeCand is set equal to ( numMergeCand – numOrigMergeCand ). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive.
4. The derivation process for zero motion vector merging candidates specified in subclause 8.5.2.1.4 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 being in mergeCandList and the NumMergeCand as the inputs and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0zeroCandm and refIdxL1zeroCandm, the prediction list utilization flags predFlagL0zeroCandm and predFlagL1zeroCandm, the motion vectors mvL0zeroCandm and mvL1zeroCandm of every new candidate zeroCandm being added in mergeCandList. The number of candidates being added numZeroMergeCand is set equal to ( numMergeCand – numOrigMergeCand – numCombMergeCand ). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand − 1, inclusive.
5. The following assignments are made with N being the candidate at position merge\_idx[ xP][ yP ] in the merging candidate list mergeCandList ( N = mergeCandList[ merge\_idx[ xP][ yP ] ] ) and X being replaced by 0 or 1:

mvLX[ 0 ] = mvLXN[ 0 ] (8‑83)

mvLX[ 1 ] = mvLXN[ 1 ] (8‑84)

refIdxLX = refIdxLXN (8‑85)

predFlagLX = predFlagLXN (8‑86)

**8.5.2.1.2 Derivation process for spatial merging candidates**

Inputs to this process are

* a luma location ( xP, yP ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
* a variable spatialCandCnt,
* a variable singleMCLFlag,
* variables specifying the width and the height of the luma prediction block, nPSW and nPSH,
* a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are (with N being replaced by A0, A1, B0, B1 or B2 and with X being replaced by 0 or 1)

* the variable spatialCandCnt,
* the availability flags availableFlagN of the neighbouring prediction units,
* the reference indices refIdxLXN of the neighbouring prediction units,
* the prediction list utilization flags predFlagLXN of the neighbouring prediction units,
* the motion vectors mvLXN of the neighbouring prediction units.

For the derivation of availableFlagN, with N being A0, A1, B0, B1 or B2 and ( xN, yN ) being ( xP − 1,  yP + nPSH ), ( xP − 1,  yP + nPSH − 1 ), ( xP + nPSW,  yP − 1 ), ( xP + nPSW − 1,  yP − 1 ) or ( xP − 1,  yP − 1 ), the following applies.

– When yP−1 is less than (( yC >> Log2CtbSize ) << Log2CtbSize), the following applies.

xB0 = ((xB0>>3)<<3) + ((xB0>>3)&1)\*7 (8‑87)  
xB1 = ((xB1>>3)<<3) + ((xB1>>3)&1)\*7 (8‑88)  
xB2 = ((xB2>>3)<<3) + ((xB2>>3)&1)\*7 (8‑89)

– If one or more of the following conditions are true with X being replaced by 0 and 1, the availableFlagN is set equal to 0, both components mvLXN are set equal to 0, refIdxLXN and predFlagLX[ xN, yN ] of the prediction block covering luma location ( xN, yN ) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

* (xP >> (log2\_parallel\_merge\_level\_minus2 + 2)) is equal to (xN >> (log2\_parallel\_merge\_level\_minus2 + 2)) and (yP >> (log2\_parallel\_merge\_level\_minus2 + 2)) is equal to (yN >> (log2\_parallel\_merge\_level\_minus2 + 2)).
* N is equal to B2 and availableFlagA0 + availableFlagA1 + availableFlagB0 + availableFlagB1 is equal to 4.
* The prediction block covering luma location ( xN, yN ) is not available [Ed. (BB): Rewrite it using MinCbAddrZS[ ][ ] and the availibility process for minimum coding blocks ] or PredMode is MODE\_INTRA.
* singleMCLFlag is equal to 0 and PartMode of the current prediction unit is PART\_2NxN or PART\_2NxnU or PART\_2NxnD and partIdx is equal to 1 and N is equal to B1
* singleMCLFlag is equal to 0 and PartMode of the current prediction unit is PART\_Nx2N or PART\_nLx2N or PART\_nRx2N and partIdx is equal to 1 and N is equal to A1
* N is equal to B1 and the prediction units covering luma location ( xP − 1, yP + nPSH − 1 ) ( N = A1 ) and luma location ( xN, yN ) (Cand. N) have the same motion vectors and the same reference indices
* N is equal to B0 and the prediction units covering luma location ( xP + nPSW − 1, yP − 1 ) ( N = B1 ) and luma location ( xN, yN ) (Cand. N) have the same motion vectors and the same reference indices
* N is equal to A0 and the prediction units covering luma location ( xP − 1, yP + nPSH − 1 ) ( N = A1 ) and luma location ( xN, yN ) (Cand. N) have the same motion vectors and the same reference indices
* N is equal to B2 and the prediction units covering luma location ( xP + nPSW − 1, yP − 1 ) ( N = B1 ) and luma location ( xN, yN ) (Cand. N) have the same motion vectors and the same reference indices
* N is equal to B2 and the prediction units covering luma location ( xP − 1, yP + nPSH − 1 ) ( N = A1 ) and luma location ( xN, yN ) (Cand. N) have the same motion vectors and the same reference indices

– Otherwise, spatialCandCnt is incremented by 1 and availableFlagN is set equal to 1 and the variables mvLX[ xN, yN ], refIdxLX[ xN, yN ] and predFlagLX[ xN, yN ] of the prediction block covering luma location ( xN, yN ) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

**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 variable combIdx, 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 8‑7.
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,
     + PicOrderCnt( RefPicList0[ refIdxL0l0Cand ] ) != PicOrderCnt( RefPicList1[ refIdxL1l1Cand ] ) | | mvL0l0Cand != mvL1l1Cand,

the candidate combCandk with k equal to ( numMergeCand − numInputMergeCand ) is added at the end of mergeCandList ( mergeCandList[ numMergeCand ] = combCandk ) and the reference indices, the prediction list utilization flags and the motion vectors of combCandk are derived as follows and numMergeCand is incremented by 1.

refIdxL0combCandk = refIdxL0l0Cand (8‑90)

refIdxL1combCandk = refIdxL1l1Cand (8‑91)

predFlagL0combCandk = 1 (8‑92)

predFlagL1combCandk = 1 (8‑93)

mvL0combCandk[ 0 ] = mvL0l0Cand[ 0 ] (8‑94)

mvL0combCandk[ 1 ] = mvL0l0Cand[ 1 ] (8‑95)

mvL1combCandk[ 0 ] = mvL1l1Cand[ 0 ] (8‑96)

mvL1combCandk[ 1 ] = mvL1l1Cand[ 1 ] (8‑97)

numMergeCand = numMergeCand + 1 (8‑98)

1. The variable combIdx is incremented by 1.
2. When combIdx is equal to ( numOrigMergeCand \* ( numOrigMergeCand − 1 ) ) or numMergeCand is equal to MaxNumMergeCand, combStop is set to TRUE.

**Table 8‑7 – Specification of l0CandIdx and l1CandIdx**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **combIdx** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **l0CandIdx** | 0 | 1 | 0 | 2 | 1 | 2 | 0 | 3 | 1 | 3 | 2 | 3 |
| **l1CandIdx** | 1 | 0 | 2 | 0 | 2 | 1 | 3 | 0 | 3 | 1 | 3 | 2 |