|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  8th Meeting: San Jose, CA, USA, 1-10 February, 2012 | Document: JCTVC-H0082  M22954 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG10: Configurable and CU-group level parallel merge/skip** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Minhua Zhou | Tel: Email: | +1 214.480.3816  [zhou@ti.com](mailto:zhou@ti.com) |
| *Source:* | Texas Instruments Inc | | |

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

# Abstract

# The current HEVC merge/skip mode design is highly sequential and introduces dependency among neighboring PUs, which can lead to significant quality loss if motion estimation (ME) is performed in parallel at different block levels for throughput or implementation cost reasons. For parallel ME level of 16x16 that is used today, the measured average loss is 2.5% in RA-HE, 3.1% in RA-LC, 3.1% in LB-HE and 4.2% in LB-LC. The loss is caused by fact that the merge/skip mode cannot be tested for those PUs inside the 16x16 block whose neighboring motion data are still unavailable during the parallel processing process. It is proposed to add a high-level syntax element to signal the parallel level of merge/skip mode, divide a LCU into parallel motion estimation regions (MERs) and allow only those neighboring PUs which belong to different MERs from the current PU to be included in the merge/skip MVP list construction process. With the proposed change, the average loss for parallel ME level of 16x16 has been reduced to 0.5% in RA-HE, 0.8% in RA-LC, 0.5% in LB-HE and 0.9% in LB-LC. The proposed design is configurable and backward compatible to the current design but offers flexibility for high throughput and high quality encoder designs.

# Introduction

# The current HEVC merge/skip mode design [2] is highly sequential and introduces dependency among neighboring PUs, which creates difficulty for the motion estimation on the encoder side for the pipelined architecture, in which motion vectors might be estimated in parallel for multiple PUs. The parallelized merge/skip mode [3][4] is further investigated. This document reports test results on parallel merge/skip mode.

# Problem statement

Current PU

**4**

**2**

**5**

**1**

Co-located PU

**CR 6**

**H 6**

**3**

Figure 1. Illustration of the merge/skip MVP list construction of HM5.0

Figure 1 illustrates the MVP list construction of the merge/skip mode defined in HM5.0. For the merge/skip mode of the current PU, a total of five neighboring PUs (i.e. left (1), upper (2), upper-right (3), bottom-left (4) and corner (5) neighboring PUs) and temporally co-located PUs (CR 6 and H6) are checked to form the initial merge/skip MVP list. A MVP candidate of a spatial/temporal neighboring PU of lower index in Figure 1 has higher priority in the MVP list. The reference index for temporal TMVP derivation also depends on the availability and reference index of left neighbor of the current PU.

Figure 2. For a block of size 2N x 2N which groups multiple CUs/PUs, merge/skip search is only possible for the PUs marked in green if motion estimation is carried out in parallel for entire 2N x 2N region.

With the current HM5.0 design, if motion estimation is carried out in parallel for entire 2N x 2N block region which may contain multiple CUs, only the PUs marked in green color of the first CU (see CU0 in Figure 2) are able to have merge/skip mode searched in parallel to the regular motion estimation process (the regular motion estimation process estimates motion vectors for a PU and is independent of neighboring PUs), the rest of PUs/CUs will have to drop testing of the merge and skip mode because the neighboring motion data (motion vector, prediction direction, reference index, prediction size, intra/inter flag) are still unavailable when the parallel motion estimation is taking place. Figure 3 provides an example to elaborate the problem. In Figure 3, a 2Nx2N CU contains four NxN CUs, motion estimation is required to carry out for all the PUs/CUs in entire 2N x 2N region in parallel for throughput reason, and throughput is dictated by regular motion estimation for NxN CU0, CU1, CU2, CU3 and 2Nx2N CU. Because inter-dependency of CU/PU merge/skip MVP list derivation (MCL), only the MCL and merge estimation (MME) of the first CU (CU0) and 2Nx2N CU can run in parallel to the regular motion estimation, the MCL and merge estimation (MME) for the rest of CUs (CU1, CU2, CU3) has to run sequentially. Because cycle budget in this case can only afford to perform regular ME in time, the MCL/MME has to be skipped for CU1, CU2, CU3 to meet throughput requirement, which causes quality loss.

**Figure 3. An example of parallel motion estimation with HM5.0 design.**

The quality loss can be significant because merge and skip mode have great impact on video quality. As shown in Table 1, significant quality loss is observed if the parallel motion estimation is carried out at 16x16 block level and above.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parallel ME level 2N x 2N**  **(LCU = 64x64)** | **RA-HE (%)** | **RA-LC (%)** | **LB-HE (%)** | **LB-LC (%)** |
| 64 x 64 | 7.3 | 8.4 | 8.4 | 10.9 |
| 32 x 32 | 5.3 | 6.3 | 6.6 | 8.4 |
| 16 x 16 | 2.5 | 3.1 | 3.1 | 4.2 |
| 8 x 8 | 0.2 | 0.3 | 0.3 | 0.4 |

Table 1, BD-rate increase relative toHM5.0 anchor when the parallel motion estimation is performed at 2N x 2N block level (N = 64, 32, 16 and 8, LCU size is fixed to 64x64)

Detailed results are reported in attached spreadsheets:

HM-5.0-TI-anchor\_vs\_HM8x8.xls

HM-5.0-TI-anchor\_vs\_HM16x16.xls

HM-5.0-TI-anchor\_vs\_HM32x32.xls

HM-5.0-TI-anchor\_vs\_HM64x64.xls

Parallel motion estimation is needed for practical video encoder implementation for meeting throughput requirements and lowering implementation cost of the motion estimation engine. For example, for the previous standards including AVC the typical parallel motion estimation level is 16x16, which is aligned with the macroblock size. For AVC, some encoders are doing 16 x 32 level parallel motion estimation for MBAFF support. For HEVC in which LCU size can go up to 64 x 64 and UHD (4K x 2K or 8K x 4K ) application is expected, performing parallel motion estimation at 32 x 32 can become a minimum requirement to meet real-time requirements and cost constraints.

# Algorithm description

To reduce the quality loss in the parallel motion estimation environment, we propose to modify the spatial MVP availability rules for merge/skip MVP list to enhance the parallelism of merge/skip mode, so that merge/skip mode can be tested for all the PUs during the parallel motion estimation. The best idea here is to decouple merge/skip MVP list derivation from regular motion estimation. The key parts of the algorithm are as follows:

1. Define a high level syntax element (**log2\_parallel\_merge\_level\_minus2)** in picture parameter set to indicate parallel processing level of merge/skip mode.
2. Based on the value of **log2\_parallel\_merge\_level\_minus2,** divide a LCU into a number of non-overlapped of motion estimation regions (MER). A MER has square shape.
3. Modify the spatial MVP availability rules: if a neighboring PU and the current PU belong to a same MER, this neighboring PU is treated as unavailable for spatial MVP derivation of the merge/skip MVP list construction process.
4. Modify the temporal MVP reference index derivation rules: if a neighboring PU and the current PU belong to a same MER, this neighboring PU is treated as unavailable for the temporal MVP reference index derivation process.

## Definition of log2\_parallel\_merge\_level\_minus2

A high level syntax element **log2\_parallel\_merge\_level\_minus2** is added to picture parameter set to signal the parallel merge/skip level. **log2\_parallel\_merge\_level\_minus2** can have value of 0, 1, 2, 3 and 4, and 1 <<( log2\_parallel\_merge\_level\_minus2+2) shall not exceed the LCU size defined in the sequence parameter set.

The parallel merge/skip levels defined by **log2\_parallel\_merge\_level\_minus2** are present in Table 3.

|  |  |  |
| --- | --- | --- |
| **log2\_parallel\_merge\_level\_minus2** | MER size | Remark |
| 0 | 4x4 | Sequential merge/skip mode for all PUs in a LCU because minimum PU size allowed by HEVC is 4x4 |
| 1 | 8x8 | Parallel merge/skip mode search allowed for all PUs inside an 8x8 block |
| 2 | 16x16 | Parallel merge/skip mode search allowed for all PUs inside a 16x16 block |
| 3 | 32x32 | Parallel merge/skip mode search allowed for all PUs inside a 32x32 block |
| 4 | 64x64 | Parallel merge/skip mode search allowed for all PUs inside a 64x64 block |

Table 3, parallel merge/skip level defined by log2\_parallel\_merge\_level\_minus2

## Partitioning a LCU into motion estimation regions (MERs)

As shown in Figure 4, a LCU (e.g. 64x64 size) is divided into a number non-overlapped motion estimation regions of equal size. From MER to MER, the motion estimation is carried out sequentially, i.e. motion estimation is performed on the next MER after the motion estimation of the current MER is finished. Within a MER, the motion estimation is carried out in parallel for all the PUs residing in the MER. The MER size should be less or equal to LCU size. In the example shown in Figure 3, a LCU is divided into four MERs.

**MER3**

**MER2**

PU0

PU1

PU2

**MER0**

**MER1**

Not unavailable MVPs

(different MERs)

unavailable MVPs

(same MER)

unavailable MVPs

(neighboring PUs are not coded yet)

**LCU**

Figure 4. Illustration of motion estimation region (MER) partitioning, and availability rules of spatial MVPs for PUs at different locations

## Spatial MVP availability check for merge/skip MVP list construction

To enable testing merge/skip mode for every PU of a MER during the parallel motion estimation process, an additional check is introduced to define the MVP availability for spatial MVP derivation and for temporal MVP reference index derivation of merge/skip mode. Let (xP, yP) and (xN, yN) be the coordinates of the top-left corner pixel of the current PU and a neighboring PU (see Figure 5), respectively, whether the current PU and its neighboring PU belong to a same MER can be easily determined. This neighboring PU is treated unavailable if the following condition satisfies.

(xP >>( **log2\_parallel\_merge\_level\_minus2+2))**

**== (**xN **>> ( log2\_parallel\_merge\_level\_minus2+2))**

and

(yP >>( **log2\_parallel\_merge\_level\_minus2+2))**

**== (**yN **>> ( log2\_parallel\_merge\_level\_minus2+2))**

Current PU

**N**

(xN, yN)

(xP, yP)

Figure 5. Illustration of the current PU and its neighboring PU

|  |  |  |
| --- | --- | --- |
| **Spatial MVP index (see figure 1)** | **Position** | **(xN , yN)** |
| 1 | Left-down | (xP -1 , yP + nPSH -1) |
| 2 | Above | (xP+nPSW-1, yP – 1) |
| 3 | Above Right | (xP+nPSW , yP – 1) |
| 4 | Left bottom | (xP – 1 , yP + nPSH) |
| 5 | Above left | (xP – 1 , yP – 1) |
| \* nPSW x nPSH is the current PU size | | |

Table 3. (xN, yN) determination for merge/skip spatial MVP candidates

Otherwise, the spatial MVP availability rules defined in current HM5.0 [2] (see section 8.4.2) are used to determine the availability of the spatial MVP for merge/skip MVP list and for reference index determination of temporal MVP in merge/skip mode. The determination of (xN, yN) is provided in Table 3 which is exactly same as defined in the current version of WD.

For example, in Figure 3 for PU0 all its spatial neighboring PUs are all located in different MERs, for PU1 all its spatial neighboring PUs are unavailable as they all belong to a same MER as the current PU, while for PU2 only part of its spatial neighboring PUs belong to different MERs. In the case of all the spatial neighboring PUs belonging to a same MER, only temporal MVP is used for the merge/skip mode.

The proposed algorithm is configurable. Note that log2\_parallel\_merge\_level\_minus2 equal to zero indicates 4x4 block level parallel motion estimation, which is essentially same as what being done in the HM reference software. Therefore, the proposed algorithm is backward compatible to the existing HM5.0 design, but offers more flexibility on the encoder side to meet throughput/quality requirements and cost constraints.

# Test Settings and Conditions

The simulations of this document have used HM5.0 software with the AMP encoder speedup disabled, the simulation platform is LSF equipped with Intel(R) Xeon(R) CPU X5570 64 bits Linux machines of different frequencies, the common test conditions and reference configurations specified in [1] are followed,

# Experimental results

Following the common testing conditions specified in [1], simulation is carried out to verify the effectiveness of the proposed algorithm. As shown in Table 4, with the proposed change in the merge/skip list construction, two third of loss caused by the parallel motion estimation can be gained back. Note that after employing the proposed parallelized merge/skip mode the coding loss becomes insignificant (less than 1%, see Table 4 16x16 row) for motion estimation parallel level 16x16 that is commonly used in video encoder implementations today.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parallel ME (merge/skip) level 2N x 2N**  **(LCU = 64x64)** | **RA-HE (%)** | **RA-LC (%)** | **LB-HE (%)** | **LB-LC (%)** |
| 64 x 64 | 2.6 | 3.3 | 3.4 | 4.7 |
| 32 x 32 | 1.6 | 2.1 | 1.9 | 2.7 |
| 16 x 16 | 0.5 | 0.8 | 0.5 | 0.9 |
| 8 x 8 | 0.0 | 0.0 | 0.0 | 0.0 |

Table 4, BD-rate increase of the proposed algorithm relative to HM5.0 anchor when the parallel motion estimation is performed at 2N x 2N block level (N = 64, 32, 16 and 8, LCU size is fixed to 64x64)

Detailed results are reported in attached spreadsheets:

HM-5.0-TI-anchor\_vs\_PME8x8.xls

HM-5.0-TI-anchor\_vs\_PME16x16.xls

HM-5.0-TI-anchor\_vs\_PME32x32.xls

HM-5.0-TI-anchor\_vs\_PME64x64.xls

The relative bit-rate saving of the proposed algorithm compared to the current HM5.0 design in the parallel motion estimation environment is summarized in Table 5. For typical parallel motion estimation level of 32x32, an average gain of 3.7% in RA-HE, 4.2% in RA-LC, 4.7% in LB-HE and 5.7% in LB-LC are observed. For detailed results please refer to attached spreadsheets. For parallel level 16x16 that is used today, the average gain is 2.0% in RA-HE, 2.3% in RA-LC, 2.6% in LB-HE and 3.3% in LB-LC.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parallel ME (merge/skip) level 2N x 2N**  **(LCU = 64x64)** | **RA-HE (%)** | **RA-LC (%)** | **LB-HE (%)** | **LB-LC (%)** |
| 64 x 64 | -4.7 | -5.1 | -5.0 | -6.2 |
| 32 x 32 | -3.7 | -4.2 | -4.7 | -5.7 |
| 16 x 16 | -2.0 | -2.3 | -2.6 | -3.3 |
| 8 x 8 | -0.2 | -0.3 | -0.3 | -0.4 |

Table 5, BD-rate saving of the proposed algorithm relative to the current HM5.0 design when the parallel motion estimation is performed at 2N x 2N block level (N = 64, 32, 16 and 8, LCU size is fixed to 64x64)

# Combined study on unified design with JCTVC-H0092 and JCTVC-H0240

This section presents the simulation results on a unified design with JCTVC-H0092 and JCTVC-H0240.

Added to the current design are

1. Fixing the temporal MVP reference index to zero to remove PU dependency for TMVP refIdx derivation (JCTVC-H0092).
2. Limiting all the PUs inside an 8x8 CU to have a same merge/skip MVP list to reduce the number of MCL to be derived (JCTVC-H0240). This constraint is enabled when **log2\_parallel\_merge\_level\_minus2 > 0.**

The results are provided in Table 6. The integrated tools from JCTVC-H0092 and JCTVC-H0240 causes about 0.1% ~ 0.2% additional loss. The detailed results are provided in

HM-5.0-TI-anchor\_vs\_PME8x8\_TMVPrefIdx0\_SingleMCLin8x8CU.xls

HM-5.0-TI-anchor\_vs\_PME16x16\_TMVPrefIdx0\_SingleMCLin8x8CU.xls

HM-5.0-TI-anchor\_vs\_PME32x32\_TMVPrefIdx0\_SingleMCLin8x8CU.xls

HM-5.0-TI-anchor\_vs\_PME64x64\_TMVPrefIdx0\_SingleMCLin8x8CU.xls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parallel ME (merge/skip) level 2N x 2N**  **(LCU = 64x64)** | **RA-HE (%)** | **RA-LC (%)** | **LB-HE (%)** | **LB-LC (%)** |
| 64 x 64 | 2.7 | 3.3 | 3.4 | 4.8 |
| 32 x 32 | 1.6 | 2.2 | 2.0 | 2.8 |
| 16 x 16 | 0.6 | 0.9 | 0.7 | 1.1 |
| 8 x 8 | 0.1 | 0.1 | 0.2 | 0.3 |

Table 6, BD-rate increase of the unified design relative to HM5.0 anchor when the parallel motion estimation is performed at 2N x 2N block level (N = 64, 32, 16 and 8, LCU size is fixed to 64x64)

# Complexity analysis

# The proposed parallel merge/skip mode adds an additional check to determine whether the current PU and neighboring PU belong to a same MER during the spatial MVP derivation process, this is a negligible increase when compared to the amount of availability checks already needed for the HM5.0 spatial MVP derivation process. This is reflected in the attached WD text.

# In terms of bit-rate overhead, one-bit will be needed in picture parameter set to signal encoder/decoder running in fully sequential merge/skip mode (i.e. the current HM common test conditions).

# Conclusion and recommendations

The proposed algorithm achieves significant quality improvement in the parallel ME environment as compared to the current HM5.0 design. It improves parallelism of the merge/skip mode and offers flexibility for high quality and high throughput HEVC encoder designs. It is recommended to review this proposal with the contributions in the same category and adopt this or other similar solution into the HEVC test model.

# References

[1] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC Document, JCTVC-G1200, Geneva, CH, Nov. 2011.

[2] [B. Bross](mailto:benjamin.bross@hhi.fraunhofer.de), [W.-J. Han](mailto:wjhan.han@samsung.com), [J.-R. Ohm](mailto:ohm@ient.rwth-aachen.de), [G. J. Sullivan](mailto:garysull@microsoft.com), [T. Wiegand](mailto:thomas.wiegand@hhi.fraunhofer.de) “High Efficiency Video Coding (HEVC) text specification Working Draft 5” JCT-VC Document, JCTVC-G1103, Geneva, CH, Nov. 2011

[3] M. Zhou, “Parallelized merge/skip mode for HEVC”. JCT-VC Document, JCTVC-F069, Torino, IT, July 2011.

[4] M. Zhou, “CE9: Test results on parallel merge/skip mode”, JCT-VC Document, JCTVC-G085, Geneva, CH, Nov. 2011

# Patent rights declaration(s)

**Texas Instruments, Inc. may have IPR 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).**

# WD Text Changes (changes marked in yellow)

#### 7.3.2.2 picture parameter set RBSP syntax

#### 7.4.2.2 picture parameter set semantics

##### 8.4.2.1.1 Derivation process for luma motion vectors for merge mode

##### 8.4.2.1.2 Derivation process for spatial merging candidates

#### 

#### Replace 7.3.2.2. Picture parameter set RBSP syntax

With

|  |  |
| --- | --- |
| pic\_parameter\_set\_rbsp( ) { | Descriptor |
| **pic\_parameter\_set\_id** | ue(v) |
| **seq\_parameter\_set\_id** | ue(v) |
| **entropy\_coding\_synchro** | u(v) |
| **cabac\_istate\_reset\_flag** | u(1) |
| if( entropy\_coding\_synchro ) |  |
| **num\_substreams\_minus1** | ue(v) |
| **num\_temporal\_layer\_switching\_point\_flags** | ue(v) |
| for( i = 0; i < num\_temporal\_layer\_switching\_point\_flags; i++ ) |  |
| **temporal\_layer\_switching\_point\_flag**[ i ] | u(1) |
| **num\_ref\_idx\_l0\_default\_active\_minus1** | ue(v) |
| **num\_ref\_idx\_l1\_default\_active\_minus1** | ue(v) |
| [Ed. (BB): not present in HM software, depends on reference list construction decision ] |  |
| **pic\_init\_qp\_minus26** **/**\* relative to 26 \*/ | se(v) |
| [Ed. (BB): not present in HM software, signaled in slice header as absolute value slice\_qp, should be implemented to be used for slice\_qp\_delta] |  |
| **constrained\_intra\_pred\_flag** | u(1) |
| **slice\_granularity** | u(2) |
| **max\_cu\_qp\_delta\_depth** | ue(v) |
| [Ed. (BB): not present in HM software, should be implemented ] |  |
| **weighted\_pred\_flag** | u(1) |
| **weighted\_bipred\_idc** | u(2) |
| **tile\_info\_present\_flag** | u(1) |
| if( tile\_info\_present\_flag = = 1 **) {** |  |
| **num\_tile\_columns\_minus1** | ue(v) |
| **num\_tile\_rows\_minus1** | ue(v) |
| if( num\_tile\_columns\_minus1 != 0 | | num\_tile\_rows\_minus1 != 0 ) { |  |
| **tile\_boundary\_independence\_flag** | u(1) |
| **uniform\_spacing\_flag** | u(1) |
| if( !uniform\_spacing\_flag ) { |  |
| for( i = 0; i < num\_tile\_columns\_minus1; i++ ) |  |
| **column\_width[**i**]** | ue(v) |
| for( i = 0; i < num\_tile\_rows\_minus1; i++ ) |  |
| **row\_height[**i**]** | ue(v) |
| } |  |
| } |  |
| **}** |  |
| if (slice\_type ==P || slice\_type == B) |  |
| log2\_parallel\_merge\_level\_minus2 | ue(v) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

In 7.4.2.2 add

**log2\_parallel\_merge\_level\_minus2** specifies parallel processing level of merge/skip mode. The value of **log2\_parallel\_merge\_level\_minus2** should be in the range of [0:log2\_min\_coding\_block\_size\_minus 3 + 1 + log2\_diff\_max\_min\_coding\_block\_size], inclusive.

##### Replace 8.4.2.1.1 Derivation process for luma motion vectors for merge mode

With

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 luma sample of the current prediction unit relative to the top-left luma sample of the current picture.

Inputs of this process are

* a luma location ( xP, yP ) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,
* variables specifying the width and the height of the prediction unit for luma, 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 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 is invoked with luma location ( xP, yP ), the width and the height of the prediction unit nPSW and nPSH and the partition index PartIdx as inputs and the output is assigned to the availability flags availableFlagN, 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 is derived as follows..

* If (xP >> (log2\_parallel\_merge\_level\_minus2 +2)) is equal to ((xP - 1) >> (log2\_parallel\_merge\_level\_minus2 +2)), and (yP >> (log2\_parallel\_merge\_level\_minus2 +2)) is equal to ((yP + nPSH - 1) >> (log2\_parallel\_merge\_level\_minus2 +2)), refIdxLXA is set equal to 0
* Otherwise, if the prediction unit covering luma location ( xP − 1, yP + nPSH − 1 ) is available and PredMode is not MODE\_INTRA, refIdxLX is set equal to refIdxLX[ xP − 1, yP + nPSH − 1 ].
* Otherwise, refIdxLX is set equal to 0.

1. The derivation process for temporal luma motion vector prediction in subclause is invoked with luma location ( xP, yP ), 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 (8‑71)

predFlagLXCol = availableFlagLXCol (8‑72)

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. When merging candidates have the same motion vectors and the same reference indices, the merging candidates are removed from the list except the merging candidate which has the smallest order in the mergeCandList.
3. The variable numMergeCand and numOrigMergeCand are set to the number of merging candidates in the mergeCandList.
4. When slice\_type is equal to B, the following applies.
   * The derivation process for combined bi-predictive merging candidates specified in subclause 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.
   * The derivation process for non-scaled bi-predictive merging candidates specified in subclause 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 refIdxL0nscaleCandl and refIdxL1nscaleCandl, the prediction list utilization flags predFlagL0nscaleCandl and predFlagL1nscaleCandl, the motion vectors mvL0nscaleCandl and mvL1nscaleCandl of every new candidate nscaleCandl being added in mergeCandList. The number of candidates being added numNscaleMergeCand is set equal to ( numMergeCand – numOrigMergeCand – numCombMergeCand ). When numNscaleMergeCand is greater than 0, l ranges from 0 to numNscaleMergeCand – 1, inclusive.
5. The derivation process for zero motion vector merging candidates specified in subclause 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 – numNscaleMergeCand ). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand – 1, inclusive.
6. 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 ] (8‑88)

mvLX[ 1 ] = mvLXN[ 1 ] (8‑89)

refIdxLX = refIdxLXN (8‑90)

predFlagLX = predFlagLXN (8‑91)

*Replace 8.4.2.1.2* Derivation process for spatial merging candidates

With

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,
* 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 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.

– If one of the following conditions is true, the availableFlagN is set equal to 0, both components mvLXN are set equal to 0, refIdxLXN and predFlagLX[ xN, yN ] of the prediction unit covering luma location ( xN, yN ) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

* + If (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 unit covering luma location ( xN, yN ) is not available or PredMode is MODE\_INTRA.
  + 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
  + 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

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