|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, 14-22July, 2011 | Document: JCTVC-F712 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Additional results on JCTVC-F356 (MC complexity reduction)** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Hui Yong Kim,  \*Kyung Yong Kim, \*Gwang Hoon Park, Sung-Chang Lim, Jinho Lee,  and Jin Soo Choi  ETRI, 138 Gajeong-Ro, Yuseong-Gu, Daejeon, 305-700, Korea  \*Media Lab., Kyung Hee University 1, Seochun, Kiheung, Youngin, Gyonggi, 446-701, Korea | Tel: Email: | +82-42-860-5069 {hykim5, sclim, jinosoul, jschoi} @etri.re.kr  +82-31-201-3680 {\*kimky, \*ghpark} @khu.ac.kr |
| *Source:* | ETRI and Kyung Hee University | | |

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

# Abstract

This contribution provides two additional results related to JCTVC-F356 (Motion compensation complexity reduction for bi-prediction). In Method-A, if the L0 and L1 collocated motion information of a bi-predicted Merge/SKIP PU are identical, the neighboring PUs are searched for a non-zero motion vector. If such an MV is found, it is used as mvL1Col. In Method-B, Method-A and Method-2 of JCTVC-F356 are combined for motion compensation complexity reduction. The combined method reportedly achieves average coding gain of 0.3% and 0.5% for LD-HE and LD-LC configurations, respectively, with decoding time reduction of 2~5% in LD.

# Introduction

This contribution provides additional results related to JCTVC-F356. JCTVC-F356 presents several methods for handling the identical motion cases, mainly focusing on the aspect of motion compensation complexity reduction. This contribution provides more options for handling the identical motion cases, mainly focusing on the aspect of coding efficiency improvements.

# Method-A: Searching the neighboring PUs for mvL1Col

In this method, when a bi-predicted Merge/SKIP PU has identical temporal motion information (i.e., mvL0Col==mvL1Col and RefPicOrderCnt(currPic, refIdxL0, L0)==RefPicOrderCnt(currPic, refIdxL1, L1)), a non-zero motion vector is searched through the neighboring PUs. The search order is as follows:

1. A (left PU): the PU just left to the top-left corner pixel of the current PU
2. B (above PU): the PU just above the top-left corner pixel of the current PU
3. E (corner PU): the PU just up-left to the top-left corner pixel of the current PU

Figure 1 illustrates the search order.

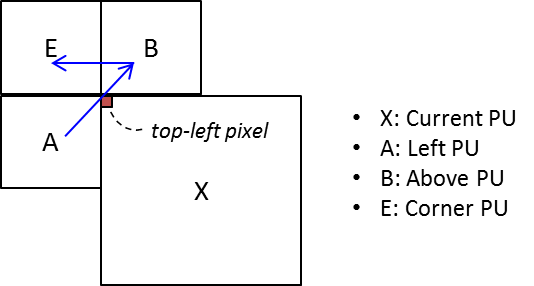
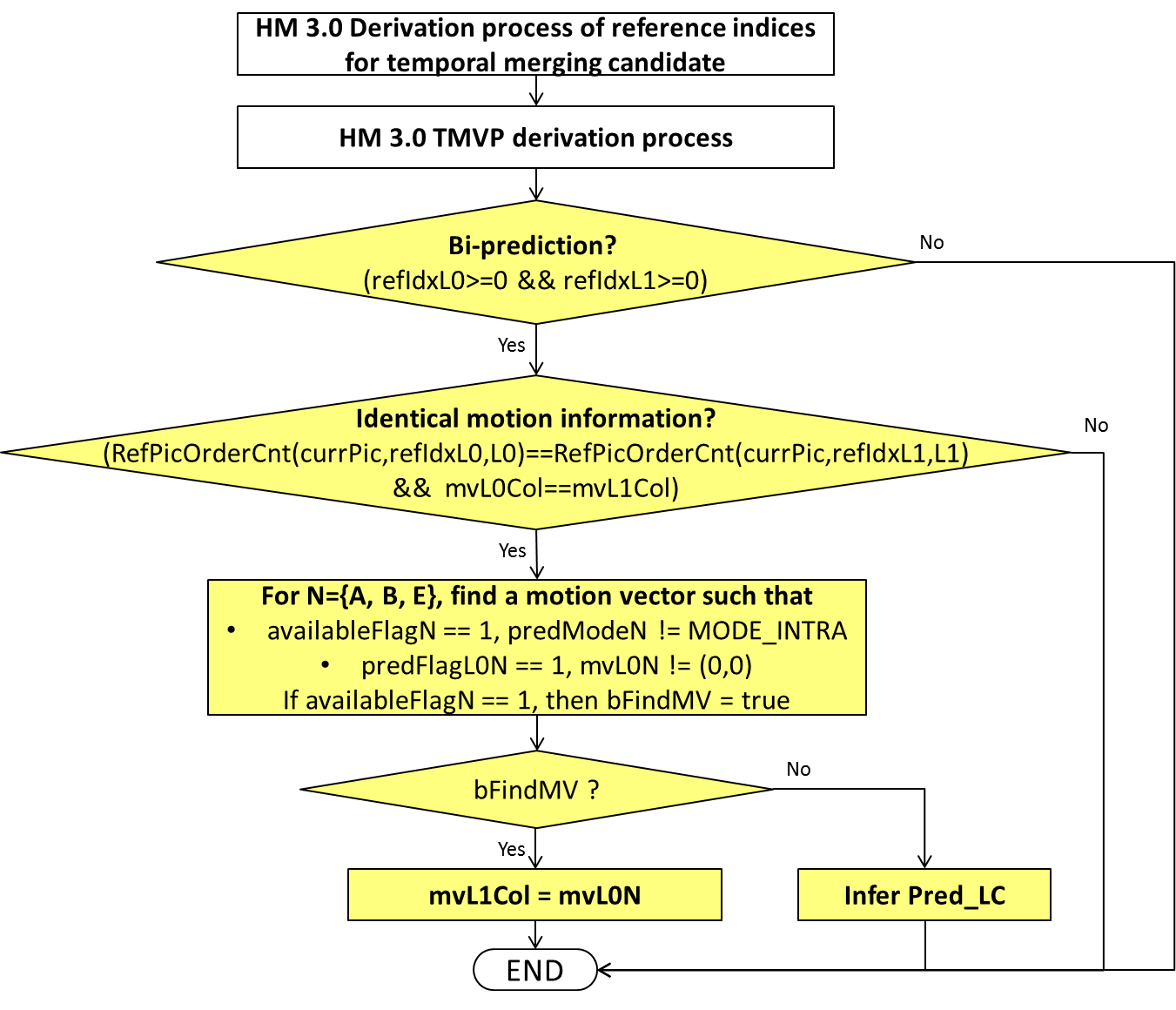


Figure 1–

Detailed algorithm is shown below:



Appendix A describes the required modification to WD3 text for Method-A (yellow-marked part only).

Table 1 shows the experimental results for Method-A.

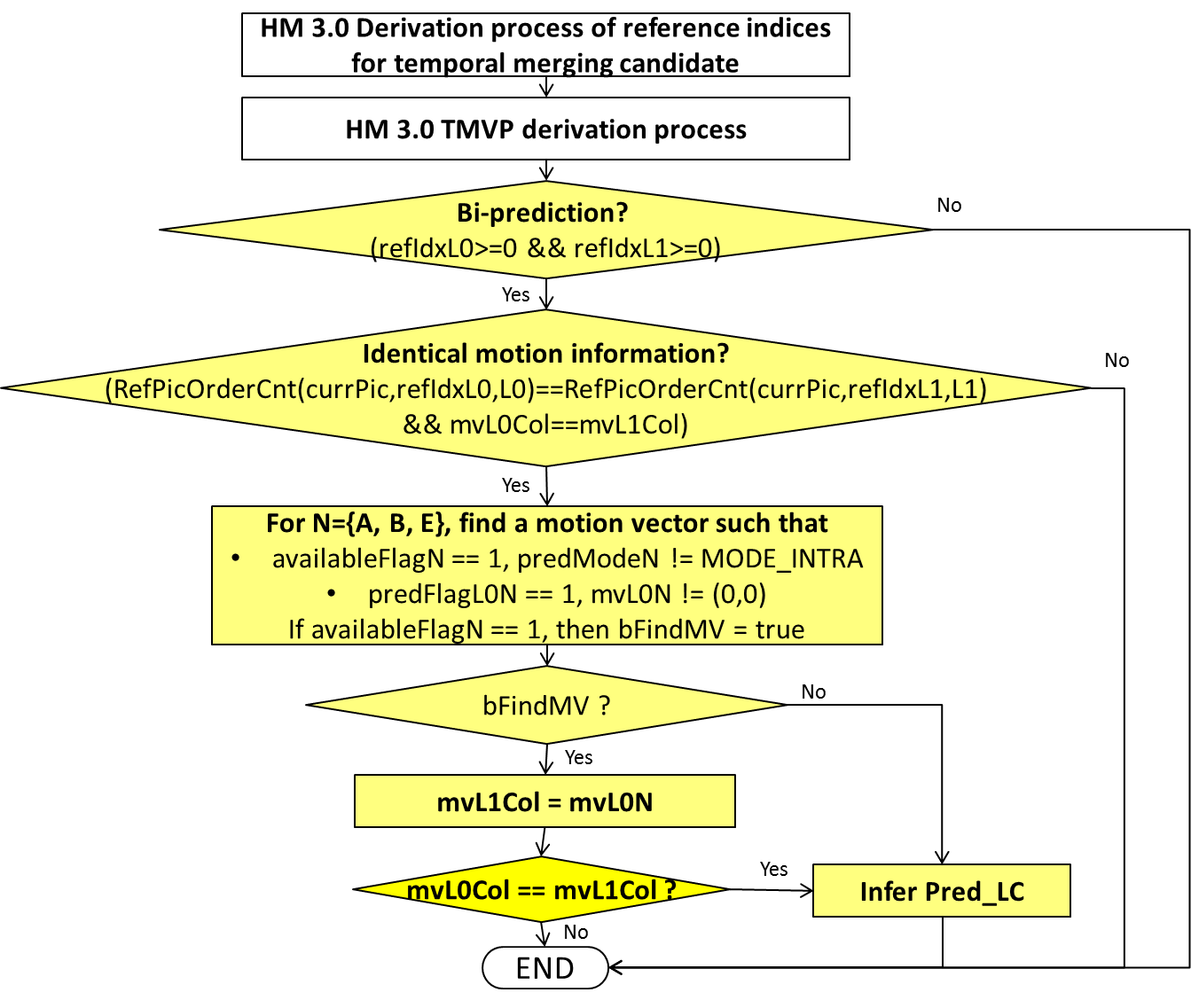
**Table1 - Method-A results (anchor: HM3.0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Access HE | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.1 | 0.0 | 0.1 | 0.0 | -0.1 | 0.1 |
| Class B | 0.0 | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class D | 0.0 | -0.1 | -0.1 | 0.0 | -0.1 | -0.1 |
| Class E |  |  |  |  |  |  |
| **Overall** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 101% | | | 101% | | |
|  |  |  |  |  |  |  |
|  | Low delay B HE | | | Low delay B LC | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | -0.3 | -0.5 | -0.7 | -0.6 | -1.0 | -1.2 |
| Class C | -0.3 | -0.6 | -0.6 | -0.3 | -0.6 | -0.6 |
| Class D | -0.1 | -0.1 | -0.1 | -0.2 | -0.2 | -0.3 |
| Class E | -0.2 | -0.7 | -0.9 | -0.6 | -1.1 | -1.6 |
| **Overall** | **-0.2** | **-0.5** | **-0.5** | **-0.4** | **-0.7** | **-0.9** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 94% | | | 95% | | |

# Method-B: Method-A combined with Method-2 of JCTVC-F356

Even with Method-A, the identical motion information cases could still remain. In this case, Method-2 of JCTVC-F356 could be utilized for reducing motion compensation complexity. In this regard, availableFlagL1Col is set to ‘0’, if the resulting mvL1Col is still equal to mvL0Col. (In the HM software, this assignment corresponds to setting refIdxL1Col=-1)

Detailed algorithm is shown below:



Appendix A describes the required modification to WD3 text for Method-B (both yellow-marked and green-marked parts).

Table 2 shows the experimental results for Method-B.

**Table2 - Method-B results (anchor: HM3.0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Random Access HE | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.0 | 0.1 | 0.0 | 0.0 | -0.1 | 0.2 |
| Class B | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class D | 0.0 | -0.1 | -0.1 | 0.0 | -0.1 | -0.1 |
| Class E |  |  |  |  |  |  |
| **Overall** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 100% | | | 101% | | |
|  |  |  |  |  |  |  |
|  | Low delay B HE | | | Low delay B LC | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | -0.3 | -0.4 | -0.4 | -0.7 | -0.9 | -1.0 |
| Class C | -0.3 | -0.6 | -0.7 | -0.4 | -0.6 | -0.8 |
| Class D | -0.2 | -0.3 | -0.1 | -0.2 | -0.7 | 0.0 |
| Class E | -0.3 | -0.9 | -1.1 | -0.7 | -1.8 | -1.5 |
| **Overall** | **-0.3** | **-0.5** | **-0.5** | **-0.5** | **-0.9** | **-0.8** |
| Enc Time[%] | 101% | | | 101% | | |
| Dec Time[%] | 95% | | | 98% | | |

# Conclusions

This contribution proposes a method for improving coding efficiency for bi-predictive Merge/SKIP PUs. Combined with a motion compensation reduction method in JCTVC-F356, the proposed method achieves both coding performance improvement and decoding time reduction. It is recommended that the propose method be integrated into the next version of HM.

# Reference

1. F.Bossen, "Common test conditions and software reference configurations", JCTVC-E700, March 2011, Geneva, CH

# Patent rights declaration(s)

**ETRI and Kyung Hee University 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).**

# Appendix A: WD3modification

***Insert the yellow marked text into section 8.4.2.1.1.***

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

This process is only invoked when 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

– theluma 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 subclause8.4.2.1.4 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 motion vectors mvL0N and mvL1N, the reference indices refIdxL0N and refIdxL1N and the prediction list utilization flags predFlagL0N and predFlagL1N with N being replaced by A, B, C or D.
2. The derivation process of reference indices for temporal merging candidate in subclause8.4.2.1.3 is invoked with luma location ( xP, yP ), nPSW, nPSH as the inputs and the output is directly assigned to refIdxLX.
3. The derivation process for temporal luma motion vector prediction in subclause8.4.2.1.7 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.  
     
   If mvL0Col is equal to mvL1Col and RefPicOrderCnt( currPic, refIdxL0, L0 ) is equal to RefPicOrderCnt( currPic, refIdxL1, L1 ), the following applies.  
     
   – If the prediction unit A covering luma location ( xP-1, yP ) is available and PredModeAis not MODE\_INTRAand predFlagL0Ais equal to 1 and mvL0A is not equal to (0,0):  
    mvL1Col = mvL0A  
     
   – Otherwise, if the prediction unit B covering luma location ( xP, yP-1 ) is available and PredModeB is not MODE\_INTRA and predFlagL0B is equal to 1 and mvL0B is not equal to (0,0):  
    mvL1Col = mvL0B  
   – Otherwise, if the predictionunit E covering luma location ( xP-1, yP-1 ) is available and PredModeE is not MODE\_INTRA and predFlagL0E is equal to 1 and mvL0E is not equal to (0,0):  
    mvL1Col = mvL0E  
     
   – If mvL0Col is equal to mvL1Col, the following applies.  
   availableFlagL1Col = 0  
     
   The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.

availableFlagCol = availableFlagL0Col || availableFlagL1Col

predFlagLXCol = availableFlagLXCol

1. The merging candidate list, mergeCandList, is constructed of which elements are given as specified order:
   1. A, if availableFlagA is equal to 1
   2. B, if availableFlagB is equal to 1
   3. Col, if availableFlagCol is equal to 1
   4. C, if availableFlagC is equal to 1
   5. D, if availableFlagD is equal to 1
2. If several merging candidates have the 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. If the number of elements NumMergeCand within the mergeCandList is equal to 1, mergeIdx is set equal to 0, otherwise, mergeIdx is set equal to merge\_idx[ xP][ yP ].
4. The following assignments are made with N being the candidate at position mergeIdx in the merging candidate list mergeCandList ( N = mergeCandList[ mergeIdx ] ) and X being replaced by 0 or 1:

mvLX[ 0 ] = mvLXN[ 0 ]

mvLX[ 1 ] = mvLXN[ 1 ]

refIdxLX = refIdxLXN

predFlagLX = predFlagLXN

1. If all availability flags availableFlagN (with N being replaced by A, B, Col, C, or D) are equal to 0, mergeIdx is set equal to 0 and the variables mvLX, refIdxLX and predFlagLX (with X being replaced by 0 or 1) are inferred as follows.

If slice\_type is equal to P, the following applies.

mvLX[ 0 ] = 0

mvLX[ 1 ] = 0

refIdxL0 = 0

refIdxL1 = -1

predFlagL0 = 1

prefFlagL1 = 0

Otherwise ( slice\_type is equal to B ), the following applies.

mvLX[ 0 ] = 0

mvLX[ 1 ] = 0

refIdxL0 = 0

refIdxL1 = 0

predFlagL0 = 1

prefFlagL1 = 1