MPEG-4 AVC/H.264 Motion Vector Prediction
Rev. 4

Author: Shevach Riabtsev (Israel, Haifa), riabtsev@yahoo.com

Reviewer: Mohsen Abdoli, MS.c Student of Computer Engineering,
Sharif University of Technology, Tehran, Iran. abdoli@ce.sharif.edu

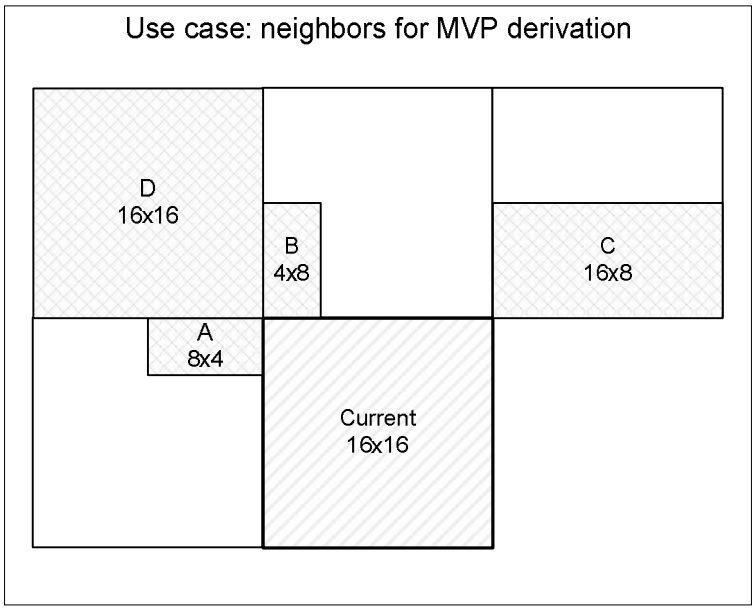
1.1 Neighborhood, Availability and Motion Vector Prediction

In MPEG-4 AVC/H.264 the motion vector predictor calculation is based on motion data of surrounding blocks: MVs, reference indexes and prediction directions (forward, backward, bi-directional). Then the motion vector predictor is used only for Luma motion vector prediction while Chroma motion vector is derived from the corresponding Luma vector. Generally speaking Luma motion vector is calculated by adding the motion vector predictor MVP to the motion vector differential MVD for every active prediction direction. Notice that MVD is extracted from the stream (see "mvd_l0"/"mvd_l1" syntax elements in the standard). In MPEG-4 AVC/H.264 there are direct and skip modes where motion vectors are not signaled and are derived from spatial neighboring blocks and a temporal co-located block.

According to MB partition the standard specifies three different types of MVP calculations:

1. for 16x8 MB partition
2. for 8x16 MB partition
3. Rest of partitions

In all above cases motion vector prediction derivation depends on availability of neighboring blocks.



Neighbor block (A,B,C or D) is considered as not available if one of the following is true:

- The block is outside the picture
- The block is outside the current slice
- The block is not yet decoded, i.e. non-causal neighbor

Notes:

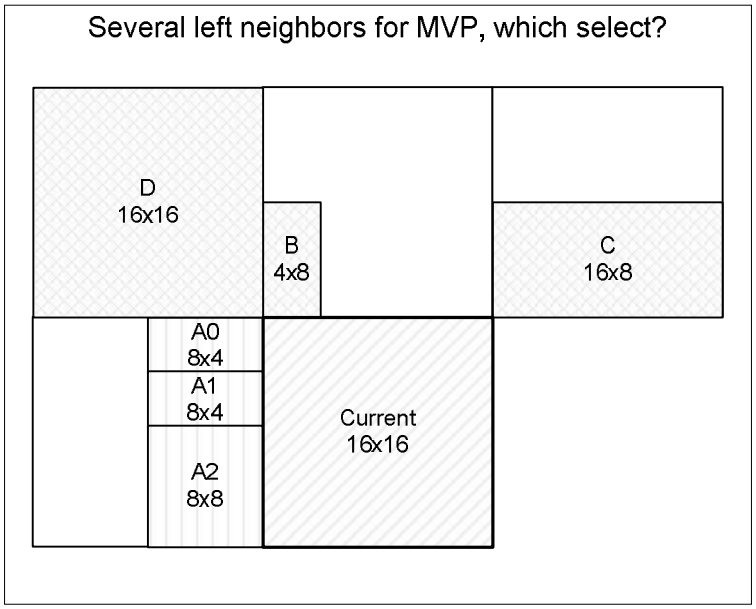
Intra partition is considered as available with MV equal to '0' and the reference index equal to '-1'.

If a neighboring block has different prediction direction, for example the current block is forward predicted while the neighboring one is backward predicted and vise versa. Then non-existing MV is treated as available with MV equal to 0 and reference index equal to '-1'.

If the neighbor C is not available then the block D is taken into consideration instead.

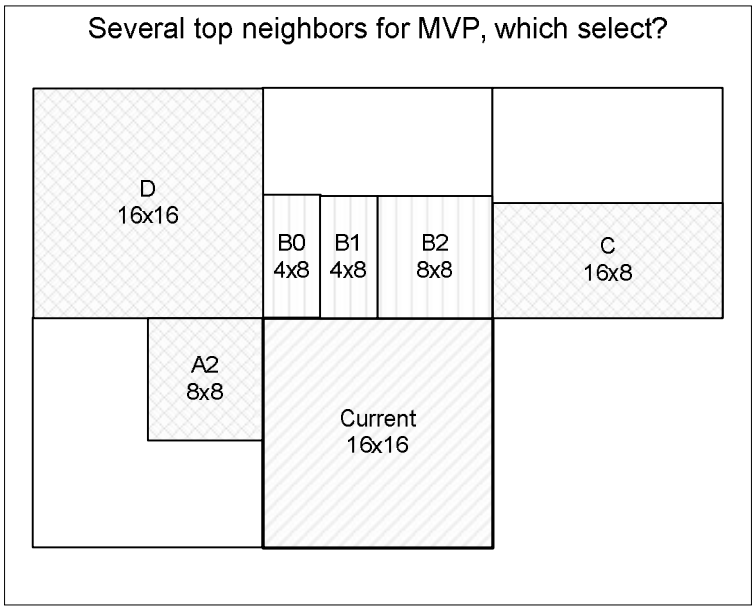
The standard denotes the block sizes as 'columns x rows' instead of commonly used 'rows x columns'. For example the block 4x8 means 4 columns and 8 rows.

In the following case the current block has several left neighbors, namely A0, A1 and A2:



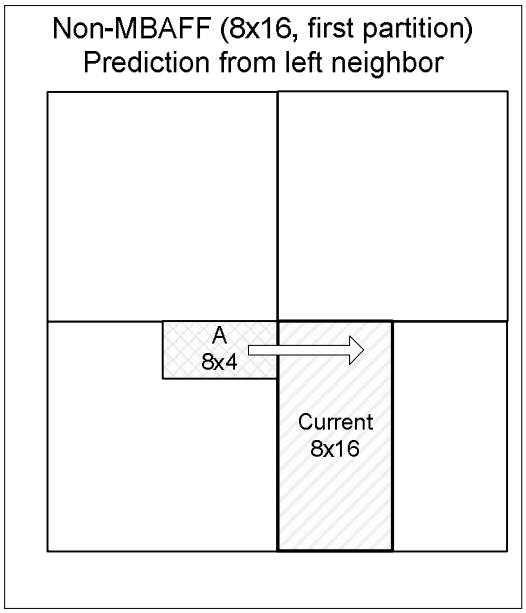
In case of several left neighbors (e.g. in the above figure the current 16x16 block has three left neighboring blocks A0, A1 and A2) the "topmost" rule is applied to select non-ambiguously the block for prediction: from all left neighboring blocks always select topmost (in the above figure the block for prediction is A0).

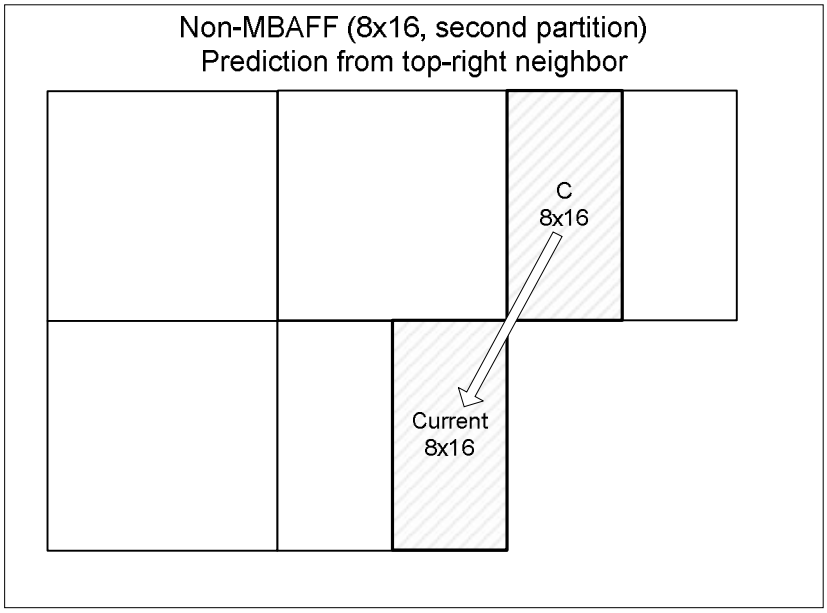
The similar situation occurs in selection of top neighbor:



In case of several top neighbors the "leftmost" rule is applied: from all top neighbors always select leftmost (in the above figure the block for prediction is B0).

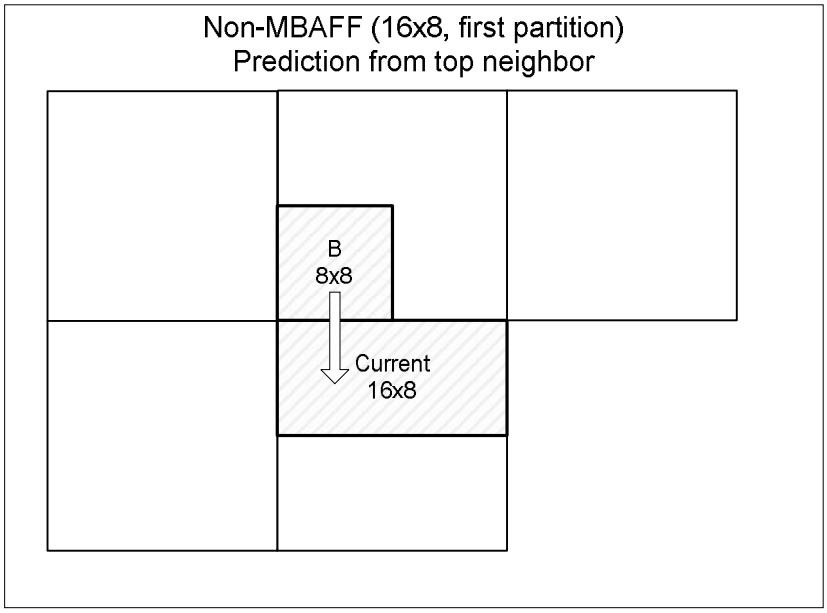
For the first two cases (16x8 and 8x16) the predictor is chosen from one of the neighboring blocks as shown in the following figures:

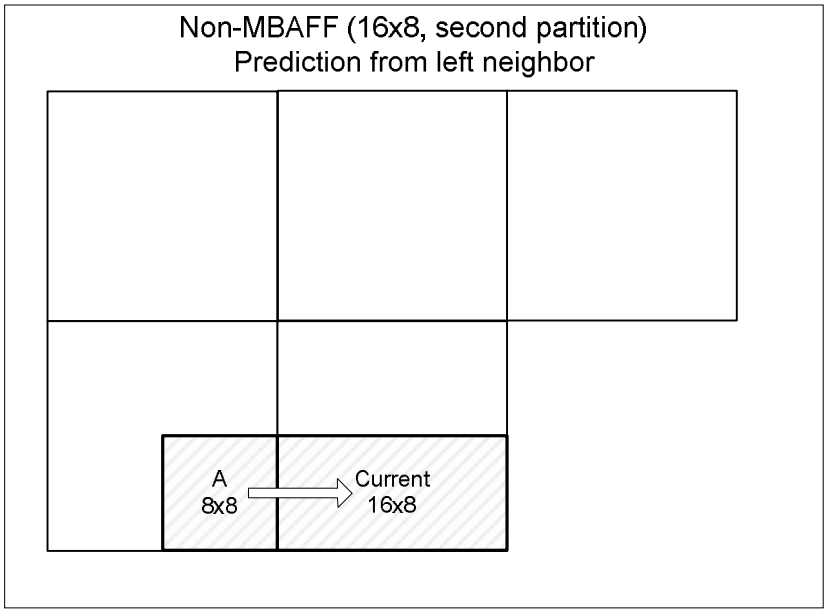




Note:

There is some drawback in MVP derivation for the second partition (the block 8x16 in the figure above) if the neighbor C is not available. In such case the neighbor D is taken but the neighbor D is more distant from the current block than the neighbor C.





For the rest cases the predicted motion vector is either identical to one of the neighboring motion vectors or it is the median of the three candidates. The derivation of the predicted motion vector (MVP) is illustrated in the following pseudo-code:

```
If ( (Neighbor B is not available) && (Neighbor C is not available) )
{
        MVP = MV of A
}
// Denote RefIdx – reference index of current block
//          RefIdxA – reference index of neighbor A
//          RefIdxB – reference index of neighbor B
//          RefIdxC – reference index of neighbor C
Else if ( ( RefIdxA == RefIdx ) && ( RefIdxB != RefIdx ) && ( RefIdxC !=  RefIdx ) )
{
        MVP = MV of A
}
Else if ( ( RefIdxB == RefIdx ) && ( RefIdxA != RefIdx ) && ( RefIdxC !=  RefIdx ) )
{
        MVP = MV of B
}
Else if ( ( RefIdxC == RefIdx ) && ( RefIdxB != RefIdx ) && ( RefIdxA !=  RefIdx ) )
{
        MVP = MV of C
}
Else
{
        MVP = MEDIAN( MV of A,  MV of B, MV of C )
}
```

MBAFF mode adds some sophistication to motion prediction process. In MBAFF case neighboring reference indexes and motion vectors has to be changed (*2 or /2) depending on field/frame type of the current MB and the neighboring MB. The decision on the scaling factor has to be done on MB-level as follows:

If current block is field and the neighboring block N is frame (here N stands for A,B or C) then

mvyN = mvyN / 2;

RefIdxN = RefIdxN * 2;

Else if current block is frame and the neighboring block N is field then

mvyN = mvyN * 2;

RefIdxN = RefIdxN / 2;

Otherwise

mvyN remains in-tact

Derivation of Chroma MV

Chroma motion vector is derived from the corresponding luma MV. In all chroma formats x-component of chroma motion MVCx equal to x-component of the corresponding luma motion vector MVx. The vertical component of the chroma motion vector MVCy is dependent on the parity of the current field (if the current picture is field) or the current MB parity (in MBAFF frame) and the reference picture parity.

In 4:2:0 format (i.e. chroma_format_idc = 1) chroma motion vector y-component MVCy is derived as follows:

Current macroblock type	Current macroblock parity	Reference picture parity	MVCy
Frame	NA	NA	MVy
Field	Bottom field	Bottom field	MVy
	Top field	Top field	MVy
	Bottom field	Top field	MVy + 2
	Top field	Bottom field	MVy – 2