### Derivation process for block vector components in intra block copying prediction mode

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 variable log2CbSize specifying the size of the current luma coding block.

Output of this process is the (nCbS)x(nCbX) array [Ed. some callees still call this a single vector] of block vectors bvIntra.

The variables nCbS, nPbSw, and nPbSh are derived as follows:

nCbS = 1  <<  log2CbSize (8‑25)

nPbSw = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_2NxN ? 1 : 2 ) (8‑25)

nPbSh = nCbS / ( PartMode = = PART\_2Nx2N | | PartMode = = PART\_Nx2N ? 1 : 2 ) (8‑25)

The variable BvpIntra[ compIdx ] specifies a block vector predictor. The horizontal block vector component is assigned compIdx = 0 and the vertical block vector component is assigned compIdx = 1.

Depending upon PartMode, the variable numPartitions is derived as follows:

– If PartMode is equal to PART\_2Nx2N, numPartitions is set equal to 1.

– Otherwise, if PartMode is equal to either PART\_2NxN or PART\_Nx2N, numPartitions is set equal to 2.

– Otherwise (PartMode is equal to PART\_NxN), numPartitions is set equal to 4.

The array of block vectors bvIntra is derived by the following ordered steps, for the variable blkIdx proceeding over the values 0..( numPartitions − 1 ):

1. The variable blkInc is set equal to ( PartMode = = PART\_2NxN ? 2 : 1 ).
2. The variable xPb is set equal to xCb + nPbSw \* ( blkIdx \* blkInc % 2 ).
3. The variable yPb is set equal to yCb + nPbSh \* ( blkIdx \* blkInc / 2 )
4. The following ordered steps apply, for the variable compIdx proceeding over the values 0..1:
5. The variable LastBvIntra[ 0 ][ compIdx ] and LastBvIntra[ 1 ][ compIdx ]specifies the last two block vector predictor. If this process is invoked for the first time for the current coding tree unit, LastBvIntra[ compIdx ] is derived as follows:

LastBvIntra[ 0 ][ 0 ] = −2\* nCbS; LastBvIntra[ 0 ][ 1 ] = 0

LastBvIntra[ 1 ][ 0 ] = −nCbS; LastBvIntra[ 1 ][ 1 ] = 0

Depending upon the number of times this process has been invoked for the current coding tree unit, subclause 8.4.4.1 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 nPbSw, the luma prediction block height nPbSh, the last block vectors LastBvIntra, and the partition index blkIdx as inputs, and the block vector predictor BvpIntra[ xPb ][ yPb ] as the output, and bvIntra[ xPb ][ yPb ][ compIdx ] is set equal to BvdIntra[ xPb ][ yPb ][ compIdx ] + BvpIntra[ xPb ][ yPb ][ compIdx ] [Ed. (GJS): Needs further formatting cleanup.]

When bvIntra[ xPb ][ yPb ][ 0 ] is not equal to LastBvIntra[ 0 ][ 0 ] or bvIntra[ xPb ][ yPb ][ 1 ] is not equal to LastBvIntra[ 0 ][ 1 ], the value of LastBvIntra[ 1 ][ compIdx ] is updated to be LastBvIntra[ 0 ][ compIdx ], and the value of LastBvIntra[ 0 ][ compIdx ] is updated to be bvIntra[ xPb ][ yPb ][ compIdx ].

1. For use in derivation processes of variables invoked later in the decoding process, the following assignment is made for x = 0..nPbSw − 1 and y = 0..nPbSh − 1:

bvIntra[ xPb + x ][ yPb + y ][ compIdx ] = bvIntra[ xPb ][ yPb ][ compIdx ] (8‑25)

– When the derivation process for z-scan order block availability as specified in subclause 6.4.1 is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + bvIntra[ xPb ][ yPb ][ 0 ], yPb + bvIntra[ xPb ][ yPb ][ 1 ] ) as inputs, the output shall be equal to TRUE.

– When the derivation process for z-scan order block availability as specified in subclause  is invoked with ( xCurr, yCurr ) set equal to ( xCb, yCb ) and the neighbouring luma location ( xNbY, yNbY ) set equal to ( xPb + bvIntra[ xPb ][ yPb ][ 0 ] + nPbSw − 1, yPb + bvIntra[ xPb ][ yPb ][ 1 ] + nPbSh – 1 ) as inputs, the output shall be equal to TRUE.

– One or both of the following conditions shall be true: [Ed. (GJS): Clarify that this is a bitstream constraint (if that is the correct interpretation).]

– bvIntra[ xPb ][ yPb ][ 0 ] + xPb − xCb is less than or equal to 0

– bvIntra[ xPb ][ yPb ][ 1 ] + yPb − yCb is less than or equal to 0

– It is a requirement of bitstream conformance that the following condition shall be true:

( xPb + bvIntra[ xPb ][ yPb ][ 0 ] + nPbSw − 1 ) / CtbSizeY − xCurr / CtbSizeY <= yCurr/CtbSizeY − ( yPb + bvIntra[ xPb ][ yPb ][ 1 ] + nPbSh − 1 ) / CtbSizeY (8‑25)