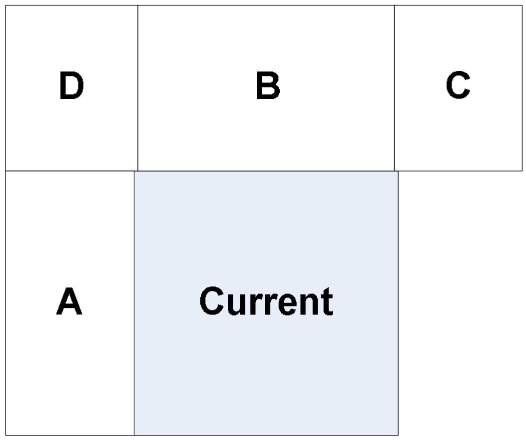
MPEG-4 AVC/H.264 Intra Prediction

Part from the book "Anatomy of H.264/AVC"
Written by Shevach Riabtsev
riabtsev@yahoo.com, Israel

1.1 Intra Prediction

Intra prediction process in AVC/H.264 utilizes spatial redundancy between neighboring blocks. Reconstructed samples (prior to deblocking) of surrounding blocks are used in calculation of predictors.
Only four neighbors are considered for predictions: A - left, D - top-left, B - top and C - top-right.



Notes:

- For chroma the neighbor C is excluded from prediction.

- Sometimes in luma prediction the corresponding C neighboring block is considered as non-available because it has not been reconstructed yet. We coin such neighbor as non-causal neighbor.

For example in Intra4x4 case for the block #3 the C-neighbor (block #4) is non-causal (since its decoding order is higher). Notice that blocks in the following figure are numbered according to decoding order.

0	**1**	**4**	**5**
2	**3**	**6**	**7**
8	**9**	**12**	**13**
10	**11**	**14**	**15**

The same is correct also for block #11, the C-neighbor #12 is non-causal.

1.1.1 Derivation of Intra Prediction Mode

In AVC/H.264 there are 9 intra prediction modes for 4x4 and 8x8 luma block and only 4 modes for chroma and 16x16 luma block.
Notice that all chroma 4x4 blocks share the same intra prediction mode. This is a drawback of the standard: sub-optimal chroma intra prediction.
Luma intra mode of Intra_16x16 MB is hidden in mb_type. For all intra MB types the chroma intra prediction mode is signaled explicitly by intra_chroma_pred_mode syntax element.

4x4 and 8x8 intra modes are predicted from intra modes of left and top blocks (common prediction schema in AVC/H.264 – from left and top). For each 4x4/8x8 block prev_intra4x4_pred_mode_flag/prev_intra8x8_pred_mode_flag is signaled.

If prev_intra4x4_pred_mode_flag/ prev_intra8x8_pred_mode_flag is 0 then rem_intra4x4_pred_mode/ rem_intra8x8_pred_mode is present.

The following pseudo-code illustrates derivation of intra prediction mode for 4x4 and 8x8 cases:

```
// ******* Determine prediction mode **********
If  LeftBlock is not available or TopBlock is not available or
   (LeftBlock is Inter and constrained_intra_pred_flag=1) or (TopBlock is Inter and
constrained_intra_pred_flag=1)
{

        PredIntraPredMode = 2    // DC mode
}
Else
{
        If LeftBlock is not Intra4x4 or Intra8x8
        {
                IntraPredModeLeft = 2 // DC mode
        }

        If TopBlock is not Intra4x4 or Intra8x8
        {
                IntraPredModeTop = 2 // DC mode
        }

        PredIntraPredMode = Min(IntraPredModeLeft, IntraPredModeTop)

}
```

```
// ******* Compute current mode ***********
If  prev_intraNxN_pred_mode_flag = 1
{
	IntraMode = PredIntraPredMode
}
Else
{
	If rem_intraNxN_pred_mode < PredIntraPredMode
	{
		IntraMode = rem_intraNxN_pred_mode
	}
	Else
	{
		IntraMode = rem_intraNxN_pred_mode + 1
	}
}
```

1.1.2 Availability of Intra Predictors

The following graph shows what surrounding MBs are used in Intra prediction (for both cases MBAFF and non-MBAFF):

Non-MBAFF case

D (top-left MB)	B (top MB)	C (top-right MB)
A (left MB)	Current MB	

MBAFF case

D (first MB in D pair)	B (first MB in B pair)	C (first MB in C pair)
D (second MB in D pair)	B (second MB in B pair)	C (second MB in C pair)
A (first MB in A pair)	Current MB pair (first MB)	
A (second MB in A pair)	Current MB pair (second MB)	

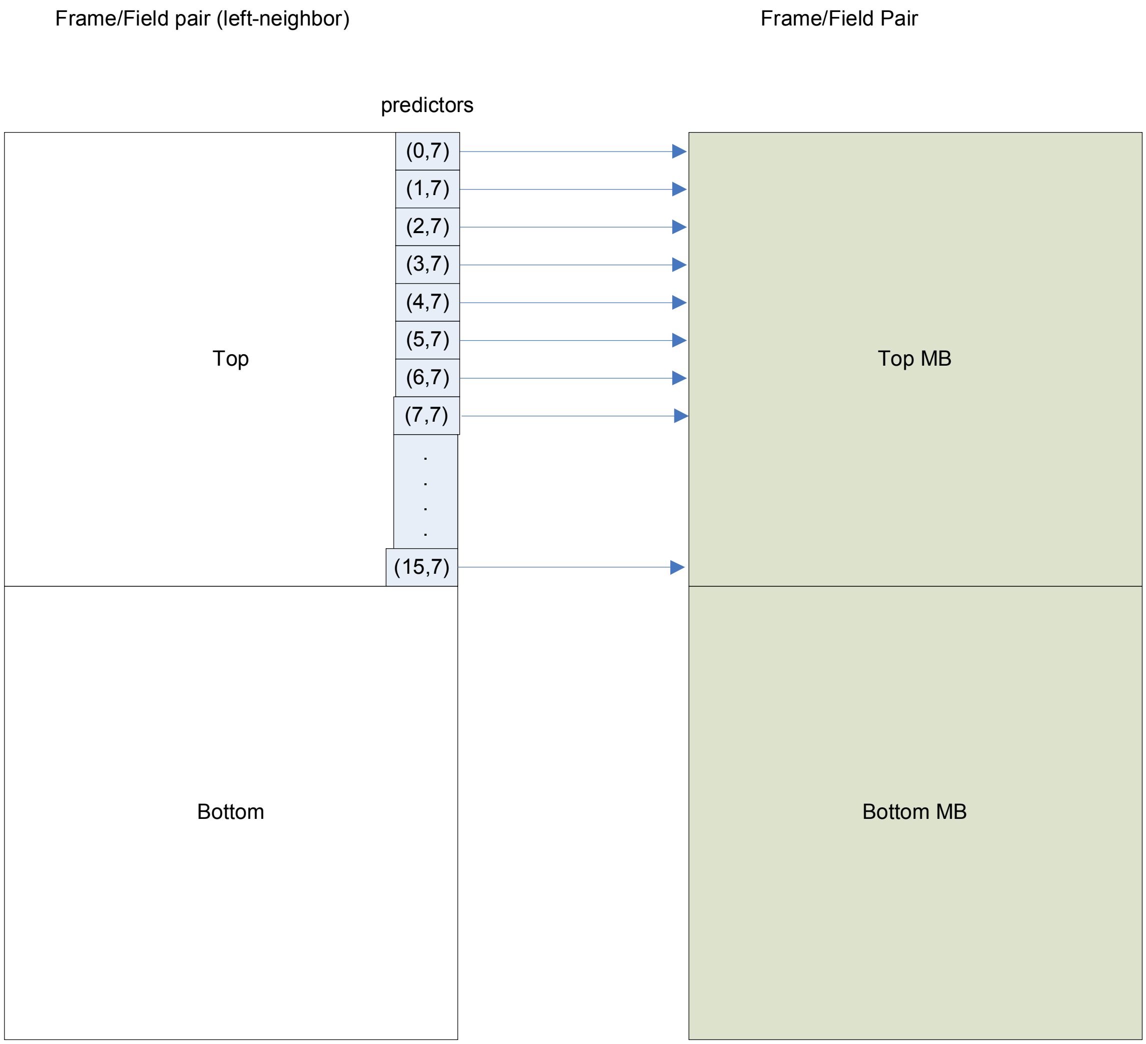
Notes

- In MBAFF case reconstructed samples of each MB in a neighboring MB pair can be used for prediction.
- Unlike luma prediction, in chroma intra prediction neighbor C (top-left) is not used for prediction

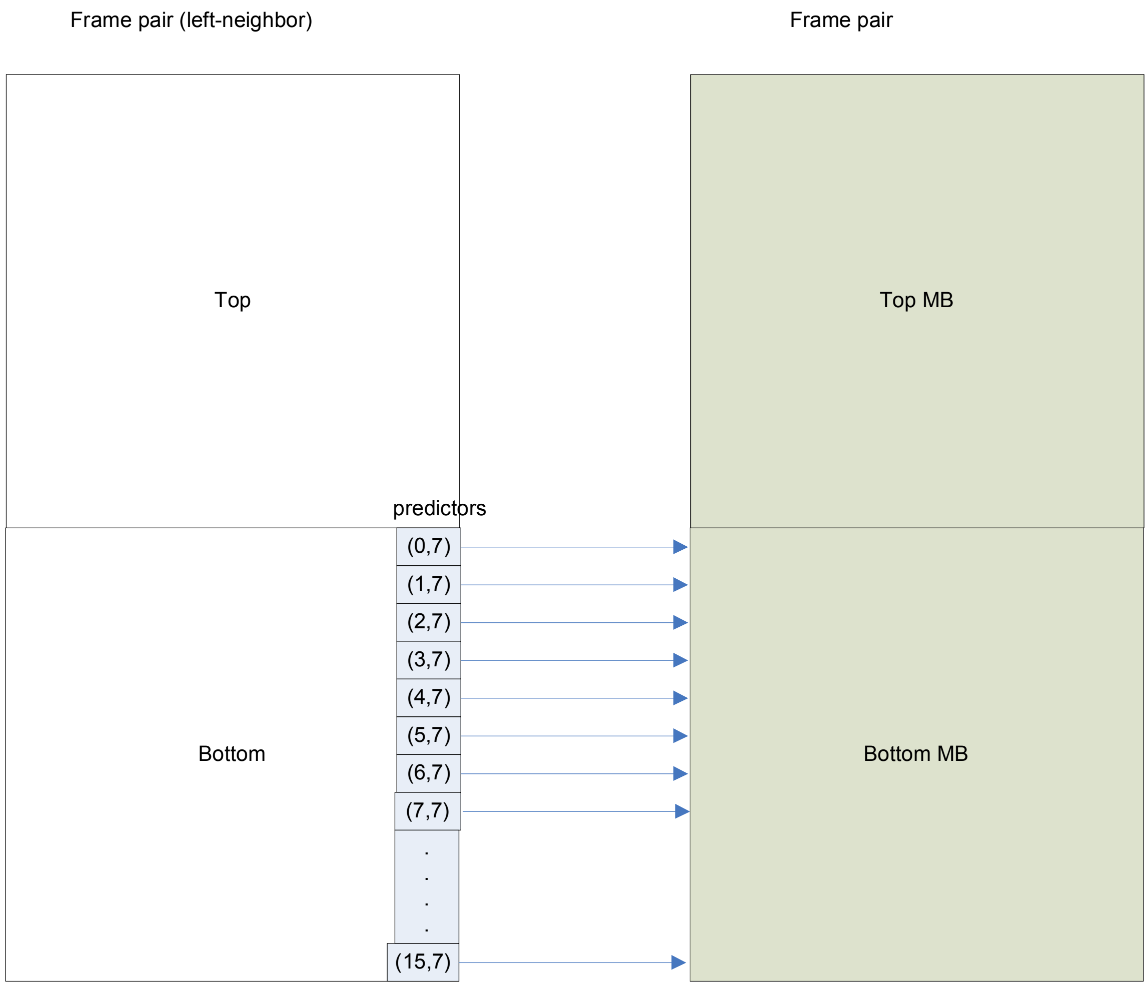
1.1.2.1 Intra Predictors in MBAFF

Unlike non-MBAFF case where neighbors for each 4x4 or 8x8 block are determined straightforward, in MBAFF case the neighbors across MB boundaries are specified in a different way according to type of current and neighboring MB pairs (frame/field). We enlist all cases of frame/field combinations and how predictors are selected for each frame/field combination. Dashed MBs denote the current MB pair.

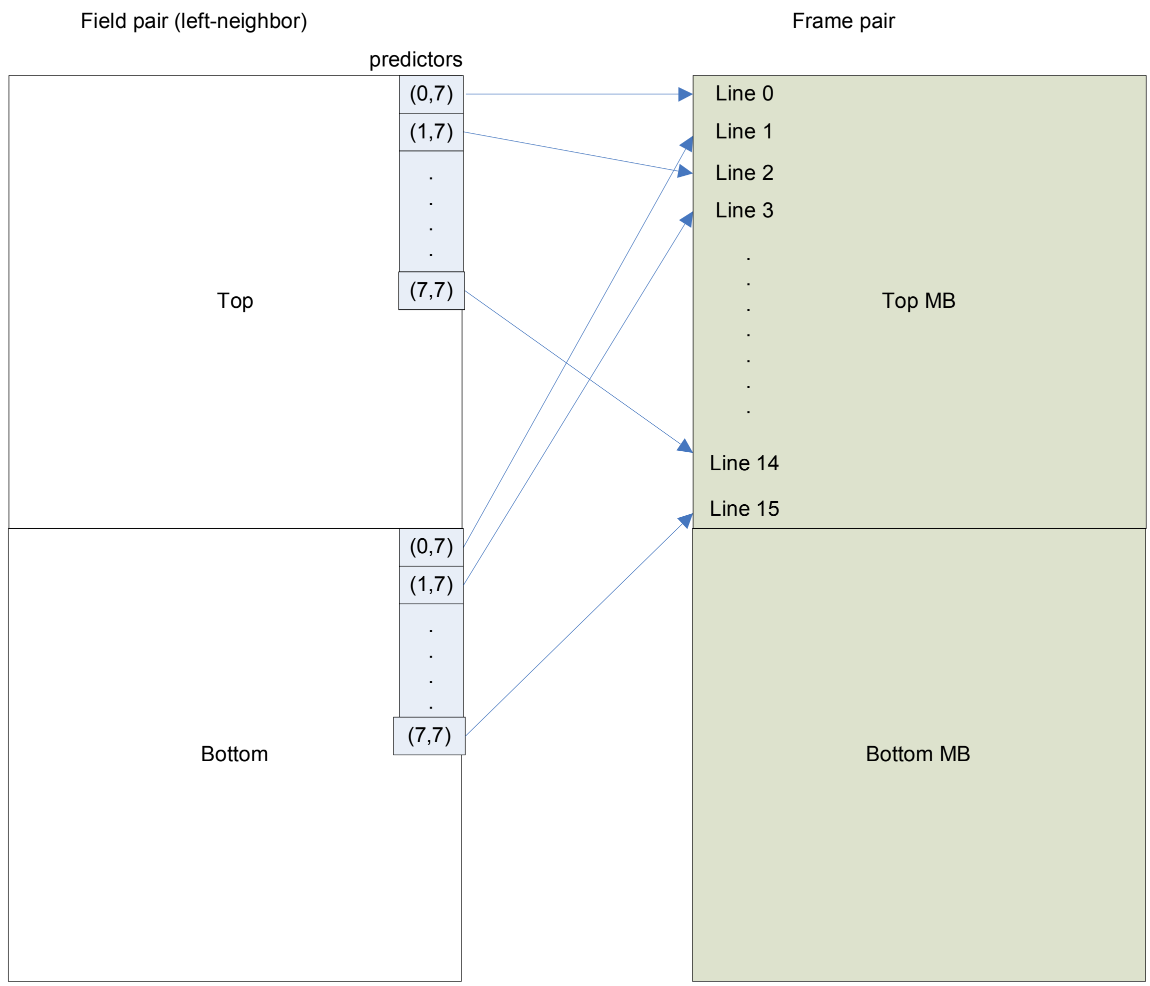
Case 1: Left MB pair is frame, current MB is top and frame or
Left MB pair is field, current MB is top and field



Case 2: Left MB pair is frame, current MB is bottom and frame or
Left MB pair is field, current MB is bottom and field



Case 3: Left MB pair is field, current MB is top and current pair is frame



Formally left predictor array is built as follows:

LeftPred[2k] = LeftTopMB[k,7], where k=0..7
LeftPred[2k+1] = LeftBotMB[k,7], where k=0..7

Case 4: Left MB pair is field, current MB is bottom and current pair is frame

LeftPred[2k] = LeftTopMB[k+8,7], where k=0..7
LeftPred[2k+1] = LeftBotMB[k+8,7], where k=0..7

Case 5: Left MB pair is frame, current MB is top and current pair is field

LeftPred[k] = LeftTopMB[2k,7], where k=0..7

LeftPred[k] = LeftBotMB[2(k-8),7], where k=8..15

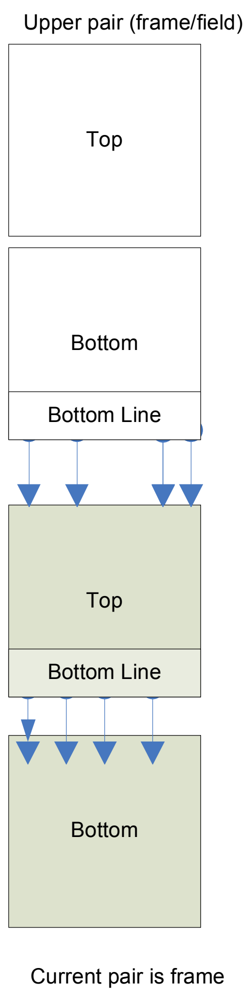
Case 6: Left MB pair is frame, current MB is bottom and current pair is field

LeftPred[k] = LeftTopMB[2k+1,7], where k=0..7

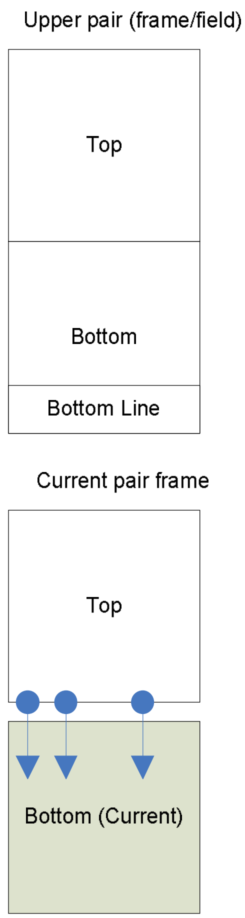
LeftPred[8+k] = LeftBotMB[2k+1,7], where k=0..7

Top predictors availability for MBAFF case:

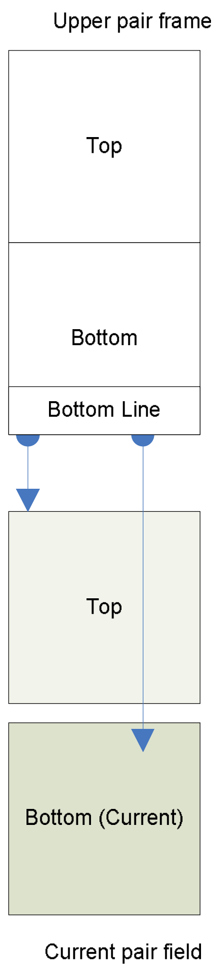
Case 1: current pair is frame



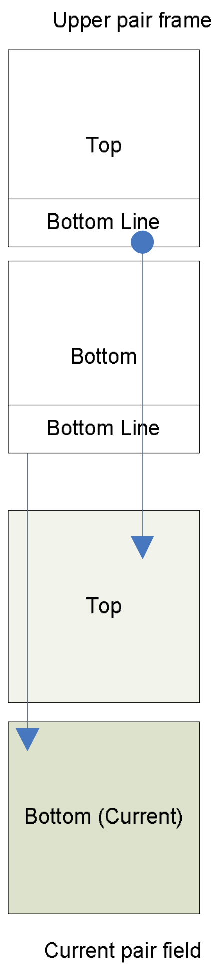
Case 2: Current MB is bottom MB, current pair is frame



Case 3: Current pair is field, upper pair is frame

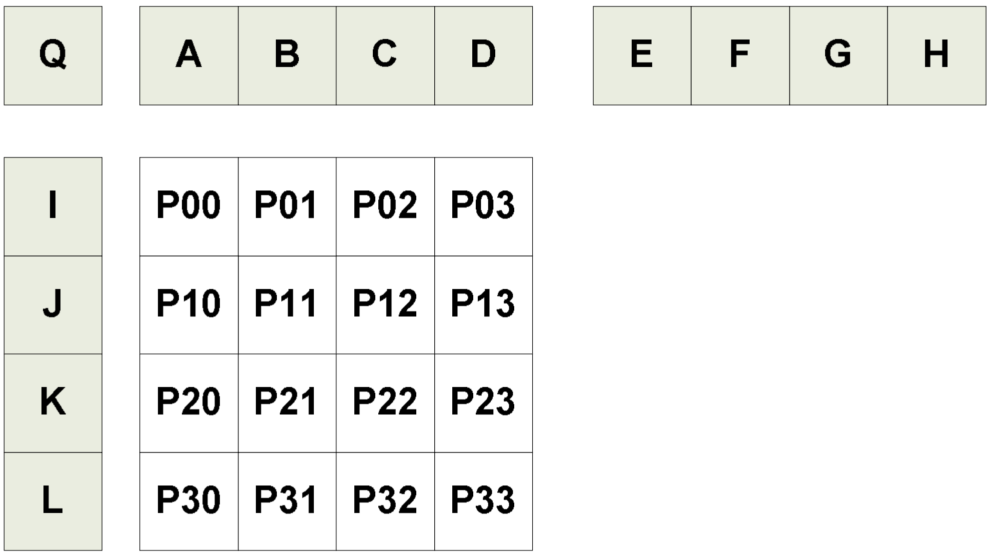


Case 4: Current pair is field, upper pair is field



1.1.3 Calculation of Intra Predictors

1.1.3.1 Luma Intra4x4



[A,B,C,D] are top samples, [E,F,G,H] – top-right samples, [I,J,K,L] – left samples and Q is top-left sample respectively. The following sub-sections describe how to calculate predictors for all 4x4 luma intra modes (except horizontal and vertical due to triviality).

1.1.3.1.1 Intra_4x4_DC (Mode 2)

If both left and top predictors are not available then

$$P_{i,j} = 128, \quad 0 \le i \le 3 \quad 0 \le j \le 3$$

Otherwise if only top predictors are not available then

$$P_{i,j} = (I + J + K + L + 2) >> 2, \quad 0 \le i \le 3 \quad 0 \le j \le 3$$

Otherwise if only left predictors are not available then

$$P_{i,j} = (A + B + C + D + 2) >> 2, \quad 0 \le i \le 3 \quad 0 \le j \le 3$$

Otherwise (all predictors are available) then

$$P_{i,j} = (A + B + C + D + I + J + K + L + 4) >> 3, \quad 0 \le i \le 3 \quad 0 \le j \le 3$$

1.1.3.1.2 Intra_4x4_Down_Left (Mode 3)

Row #0:

$$P_{0,0} = (A + 2B + C + 2) >> 2$$
$$P_{0,1} = (B + 2C + D + 2) >> 2$$
$$P_{0,2} = (C + 2D + E + 2) >> 2$$
$$P_{0,3} = (D + 2E + F + 2) >> 2$$

Row #1:

$$P_{1,0} = (B + 2C + D + 2) >> 2$$
$$P_{1,1} = (C + 2D + E + 2) >> 2$$
$$P_{1,2} = (D + 2E + F + 2) >> 2$$
$$P_{1,3} = (E + 2F + G + 2) >> 2$$

Row #2:

$$P_{2,0} = (C + 2D + E + 2) >> 2$$
$$P_{2,1} = (D + 2E + F + 2) >> 2$$
$$P_{2,2} = (E + 2F + G + 2) >> 2$$
$$P_{2,3} = (F + 2G + H + 2) >> 2$$

Row #3:

$$P_{3,0} = (D + 2E + F + 2) >> 2$$
$$P_{3,1} = (E + 2F + G + 2) >> 2$$
$$P_{3,2} = (F + 2G + H + 2) >> 2$$
$$P_{3,3} = (G + 3H + 2) >> 2$$

Notice that $P_{i,j} = P_{k,l}$ iff $i + j = k + l$. So that fact can be utilized in code optimization.

1.1.3.1.3 Intra_4x4_Diagonal_Down (Mode 4)

Row #0:

$P_{0,0} = (A + 2Q + I + 2) >> 2$

$P_{0,1} = (Q + 2A + B + 2) >> 2$

$P_{0,2} = (A + 2B + C + 2) >> 2$

$P_{0,3} = (B + 2C + D + 2) >> 2$

Row #1:

$P_{1,0} = (Q + 2I + J + 2) >> 2$

$P_{1,1} = (A + 2Q + I + 2) >> 2$

$P_{1,2} = (Q + 2A + B + 2) >> 2$

$P_{1,3} = (A + 2B + C + 2) >> 2$

Row #2:

$P_{2,0} = (I + 2J + K + 2) >> 2$

$P_{2,1} = (Q + 2I + J + 2) >> 2$

$P_{2,2} = (A + 2Q + I + 2) >> 2$

$P_{2,3} = (Q + 2A + B + 2) >> 2$

Row #3:

$P_{3,0} = (J + 2K + L + 2) >> 2$

$P_{3,1} = (I + 2J + K + 2) >> 2$

$P_{3,2} = (Q + 2I + J + 2) >> 2$

$P_{3,3} = (A + 2Q + I + 2) >> 2$

Notice that $P_{i,j} = P_{k,l}$ iff $i - j = k - l$. So that fact can be utilized in code optimization.

1.1.3.1.4 Intra_4x4_Vertical_Right (Mode 5)

Left samples, top-left (corner) sample and top samples (i.e. samples from K to D) are exploited in this mode:

Row #0:
$P_{0,0} = (A + Q + 1) >> 1$
$P_{0,1} = (B + A + 1) >> 1$
$P_{0,2} = (C + B + 1) >> 1$
$P_{0,3} = (D + C + 1) >> 1$

Row #1:
$P_{1,0} = (I + 2Q + A + 2) >> 2$
$P_{1,1} = (Q + 2A + B + 2) >> 2$
$P_{1,2} = (A + 2B + C + 2) >> 2$
$P_{1,3} = (B + 2C + D + 2) >> 2$

Row #2:
$P_{2,0} = (J + 2I + Q + 2) >> 2$
$P_{2,1} = (A + Q + 1) >> 1$
$P_{2,2} = (B + A + 1) >> 1$
$P_{2,3} = (C + B + 1) >> 1$

Row #3:
$P_{3,0} = (K + 2J + I + 2) >> 2$
$P_{3,1} = (I + 2Q + A + 2) >> 2$
$P_{3,2} = (Q + 2A + B + 2) >> 2$
$P_{3,3} = (A + 2B + C + 2) >> 2$

Notice that some predictors have the same values, e.g. $P_{0,1} = P_{2,2}$, that fact can be utilized in code optimization.

1.1.3.1.5 Intra_4x4_Horizontal_Down (Mode 6)

Left, top-left (corner) and most of top samples (i.e. samples from L to C) are exploited in this mode as follows:

Row #0:

$P_{0,0} = (Q + I + 1) >> 1$

$P_{0,1} = (I + 2Q + A + 2) >> 2$

$P_{0,2} = (B + 2A + Q + 2) >> 2$

$P_{0,3} = (C + 2B + A + 2) >> 2$

Row #1:

$P_{1,0} = (I + J + 1) >> 1$

$P_{1,1} = (Q + 2I + J + 2) >> 2$

$P_{1,2} = (Q + I + 1) >> 1$

$P_{1,3} = (I + 2Q + A + 2) >> 2$

Row #2:

$P_{2,0} = (J + K + 1) >> 1$

$P_{2,1} = (I + 2J + K + 2) >> 2$

$P_{2,2} = (I + J + 1) >> 1$

$P_{2,3} = (Q + 2I + J + 2) >> 2$

Row #3:

$P_{3,0} = (K + L + 1) >> 1$

$P_{3,1} = (J + 2K + L + 2) >> 2$

$P_{3,2} = (J + K + 1) >> 1$

$P_{3,3} = (I + 2J + K + 2) >> 2$

1.1.3.1.6 Intra_4x4_Vertical_Left (Mode 7)

Only top and top-right samples (A to G) are exploited in this mode as follows:

Row #0:

$P_{0,0} = (A + B + 1) >> 1$

$P_{0,1} = (B + C + 1) >> 1$

$P_{0,2} = (C + D + 1) >> 1$

$P_{0,3} = (D + E + 1) >> 1$

Row #1:

$P_{1,0} = (A + 2B + C + 2) >> 2$

$P_{1,1} = (B + 2C + D + 2) >> 2$

$P_{1,2} = (C + 2D + E + 2) >> 2$

$P_{1,3} = (D + 2E + F + 2) >> 2$

Row #2:

$P_{2,0} = (B + C + 1) >> 1$

$P_{2,1} = (C + D + 1) >> 1$

$P_{2,2} = (D + E + 1) >> 1$

$P_{2,3} = (E + F + 1) >> 1$

Row #3:

$P_{3,0} = (B + 2C + D + 2) >> 2$

$P_{3,1} = (C + 2D + E + 2) >> 2$

$P_{3,2} = (D + 2E + F + 2) >> 2$

$P_{3,3} = (E + 2F + G + 2) >> 2$

Notice that some predictors get the same values, e.g. $P_{0,1} = P_{2,0}$. This fact can be used for SW/HW optimization.

1.1.3.1.7 Intra_4x4_Horizontal_Up (Mode 8)

Only left samples (from I to L) are exploited in this mode as follows:

Row #0:

$P_{0,0} = (I + J + 1) >> 1$

$P_{0,1} = (I + 2J + K + 2) >> 2$

$P_{0,2} = (J + K + 1) >> 1$

$P_{0,3} = (J + 2K + L + 2) >> 2$

Row #1:

$P_{1,0} = (J + K + 1) >> 1$

$P_{1,1} = (J + 2K + L + 2) >> 2$

$P_{1,2} = (K + L + 1) >> 1$

$P_{1,3} = (K + 3L + 2) >> 2$

Row #2:

$P_{2,0} = (K + L + 1) >> 1$

$P_{2,1} = (K + 3L + 2) >> 1$

$P_{2,2} = L$

$P_{2,3} = L$

Row #3:

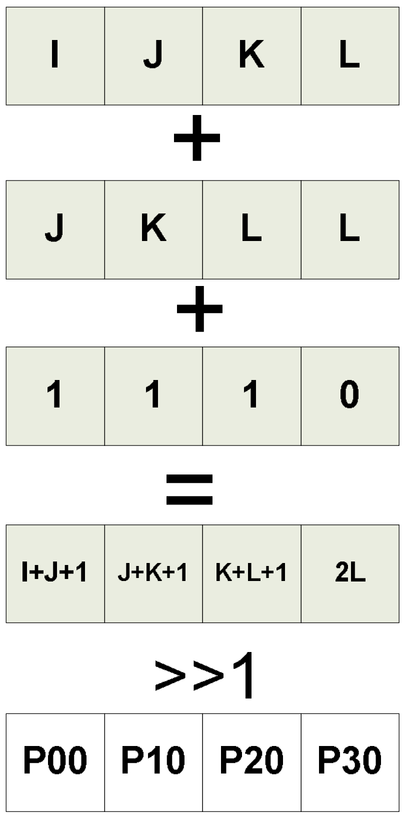
$P_{3,0} = L$

$P_{3,1} = L$

$P_{3,2} = L$

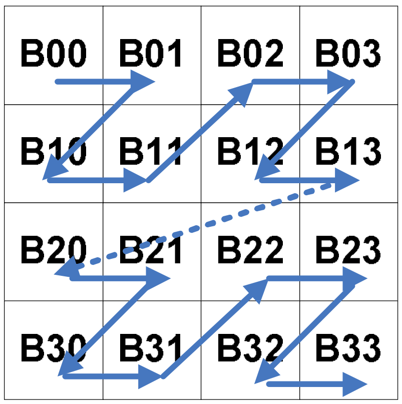
$P_{3,3} = L$

Hints for SIMD implementation:



1.1.3.1.8 Optimization Points For Intra4x4

The 4x4 luma intra prediction process is challenging for parallelization due to strong dependency among blocks.
Let's consider the following Intra4x4 MB:



In the above figure Bij means (i,j)-th 4x4 sub-block, the zig-zag-like line shows the decoding order. We can't process B01 and B10 in parallel since top-right predictors for B10 are taken from the reconstructed samples of B01. However we can process simultaneously blocks B02 and B10. Moreover blocks B03 and B11 can be processed in parallel.

The following parallezation schema can be safety applied for Intra4x4 (here '||' denotes parallel processing):

B00
B01
B02 || B10
B03 || B11
B12 || B20
B13 || B21
B22 || B30
B23 || B31
B32
B33

So, we need 10 time slots on dual-processor platform to execute 4x4 luma intra processing.

On encoder side, in order to improve parallization it’s recommended to eliminate Intra_4x4_Down_Left (mode 3) and Intra_4x4_Vertical_Left (mode 7) due to dependency on top-right samples. The new parallezation schema for dual processor platform is:

B00
B01 || B10
B02 || B11
B03 || B12
B13 || B20
B21 || B30
B22 || B31
B23 || B32
B33

Total: 9 time slots.

1.1.3.2 Luma Intra16x16

1.1.3.3 Chroma