

JVET-P0530

Non-CE4: Alignment of luma and chroma weights calculation for TPM blending

Zhipin Deng, Li Zhang, Kai Zhang, Hongbin Liu, Yue Wang



Summary

■ Problems in VVC WD6

- A misalignment between luma and chroma weights in TPM blending
 - Even for 4:4:4 chroma format, the luma and chroma weights are not aligned
 - Doesn't make sense for INTER coding tools as the luma and chroma are normally aligned
- Luma and chroma weights are calculated with different computing logics
 - Additional logic would be necessary to calculate the chroma weights

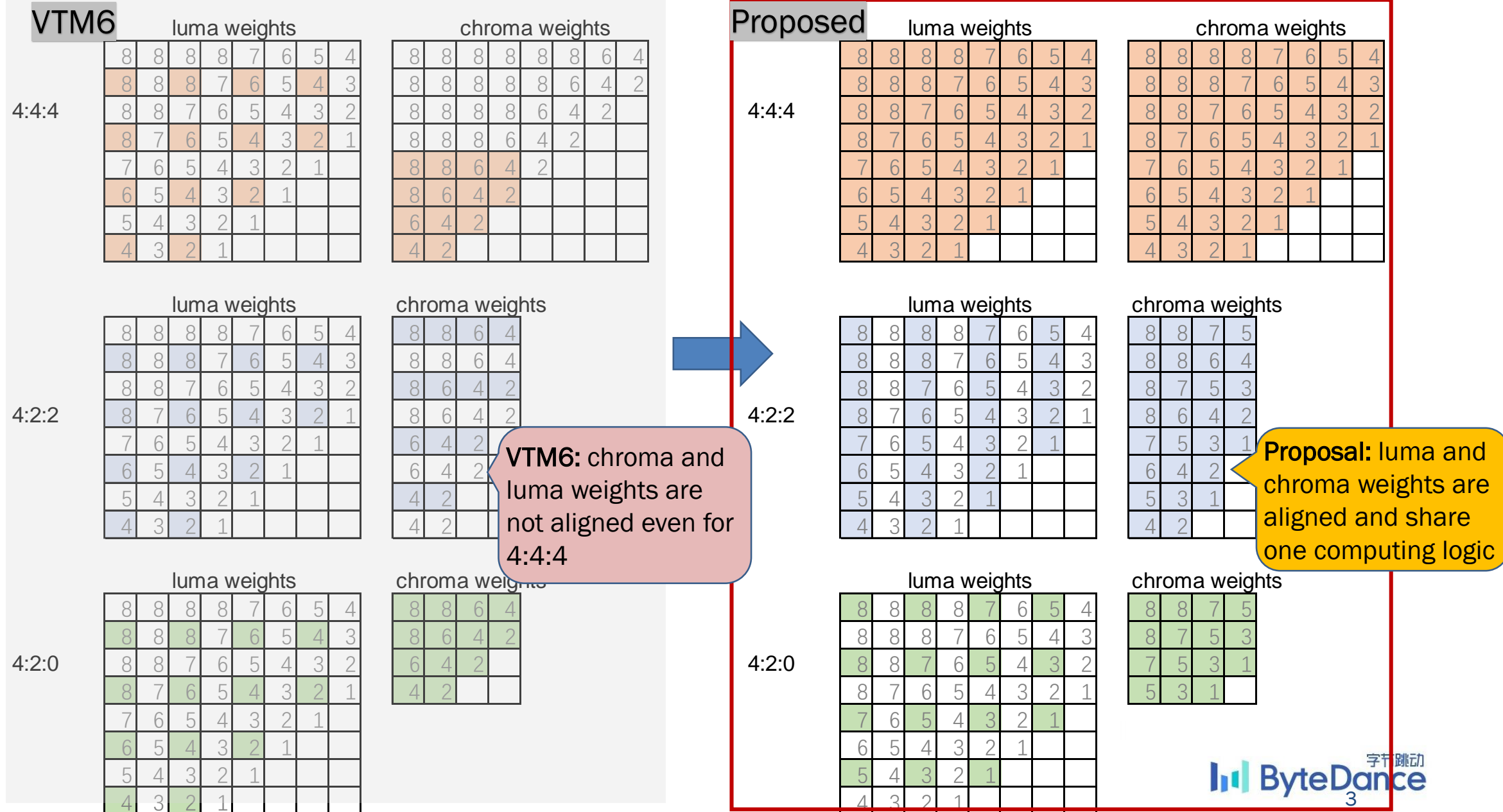
■ Proposal

- Unification of chroma and luma weights in TPM blending

■ Experimental Results

BD-rate Y	4:2:0	4:2:2	4:4:4
RA	0.01%	-0.01%	-0.01%
LDB	-0.02%	-0.06%	-0.01%

Proposal: align chroma weight to luma for TPM



Experimental results

		Y	U	V
YUV4:2:0	RA	0.01%	-0.04%	-0.03%
	LDB	-0.02%	0.10%	-0.11%
YUV4:2:2	RA	-0.01%	-0.03%	-0.02%
	LDB	-0.06%	0.02%	-0.07%
YUV4:4:4	RA	-0.01%	0.04%	0.00%
	LDB	-0.01%	-0.08%	-0.03%

Conclusions

■ Benefit

- Same blending weight for each color component of 4:4:4 chroma format
- Chroma weights can be subsampled from luma (storage saved)
- Unify the logic for luma and chroma (share one computing logic)
- No coding performance loss

■ It is recommended adopting the proposed method

Thanks InterDigital for crosschecking! (JVET-P0936)

Depending on the values of triangleDir, wS and cIdx, the prediction samples pbSamples[x][y] with $x = 0..nCbW / SubWidthC - 1$ and $y = 0..nCbH / SubHeightC - 1$ are derived as follows: ↵

- The variables xIdx and yIdx are derived as follows: ↵

$$xIdx = (cIdx == 0) ? x : x * SubWidthC ↵$$

$$yIdx = (cIdx == 0) ? y : y * SubHeightC ↵$$

- The variable wIdx specifying the weight of the prediction sample is derived as follows: ↵

- If cIdx is equal to 0 and triangleDir is equal to 0, the following applies: ↵

$$wIdx \ wValue = (nCbW > nCbH) ? (Clip3(0, 8, (xIdx / nCbR - yIdx) + 4)) : (Clip3(0, 8, (xIdx - yIdx / nCbR) + 4)) ↵ \quad (8-842)$$

- Otherwise, if cIdx is equal to 0 and (triangleDir is equal to 1), the following applies: ↵

$$wIdx \ wValue = (nCbW > nCbH) ? (Clip3(0, 8, (nCbH - 1 - xIdx / nCbR - yIdx) + 4)) : (Clip3(0, 8, (nCbW - 1 - xIdx - yIdx / nCbR) + 4)) ↵ \quad (8-843)$$

- Otherwise, if cIdx is greater than 0 and triangleDir is equal to 0, the following applies: ↵

$$wIdx = (nCbW > nCbH) ? (Clip3(0, 4, (x / nCbR - y) + 2)) : (Clip3(0, 4, (x - y / nCbR) + 2)) ↵ \quad (8-844)$$

- Otherwise (if cIdx is greater than 0 and triangleDir is equal to 1), the following applies: ↵

$$wIdx = (nCbW > nCbH) ? (Clip3(0, 4, (nCbH - 1 - x / nCbR - y) + 2)) : (Clip3(0, 4, (nCbW - 1 - x - y / nCbR) + 2)) ↵ \quad (8-845)$$

- The variable wValue specifying the weight of the prediction sample is derived using wIdx and cIdx as follows: ↵

$$wValue = (cIdx == 0) ? Clip3(0, 8, wIdx) : Clip3(0, 8, wIdx * 2) ↵ \quad (8-846)$$

- The prediction sample values are derived as follows: ↵

$$pbSamples[x][y] = Clip3(0, (1 << bitDepth) - 1, (predSamplesLA[x][y] * wValue + predSamplesLB[x][y] * (8 - wValue) + offset1) >> shift1) ↵ \quad (8-847)$$