

AHG9: Fix on high-level syntax related to coding tree constraints (JVET-0174)

Shih-Ta Hsiang, Chen-Yen Lai, Ching-Yeh Chen, Yu-Wen Huang, Shaw-Min Lei

Problem with Signaling Max MTT Depth

- `pic_width_in_luma_samples`, `pic_height_in_luma_samples` shall be integer multiples of $\text{Max}(8, \text{MinCbSizeY})$
- Each out-of-bounds CU along picture boundaries is inferred to be further split (by QT or BT)
- Problem statement (ticket # 584)
 - When $\text{MinQtSize} > \text{MinCbSize}$ and $\text{MaxMttDepth} == 0$, no split type is allowed for splitting a CU with size less than or equal to `MinQtSize`
 - An out-of-bounds CU with size less than or equal to `MinQtSize` is inferred to be further split while having all split types disabled
- Example
 - $\text{MaxMttDepth} = 0$, $\text{MinQtSize} = 16$, $\text{MinCbSize} = 4$ for coding HD video sequences with picture size 1920 x 1080 or $(256 \times 7 + 128) * (128 \times 8 + 32 + 16 + 8)$

Proposed Solution 1

- Proposed that the maximum MTT depth shall be greater than 0 when the minimum QT size is greater than the minimum coding block size
 - The BT split is thus enabled for further splitting the out-of-bounds CUs with block size less than or equal to the minimum QT size
- Proposed syntax modifications
 - The maximum MTT depth is signaled before the minimum QT size
 - When the specified maximum MTT depth is equal to 0 (MTT split is disabled), the minimum QT size is inferred to be equal to the minimum coding block size
 - The resulting CU partitioning structure is reduced to the quadtree-based CU partitioning structure in HEVC, where each CU can only be further partitioned by QT split and the minimum QT size is equal to the specified minimum coding block size.

Modified SPS

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
...	
if(ChromaArrayType != 0)	
qtbtt_dual_tree_intra_flag	u(1)
log2_min_luma_coding_block_size_minus2	ue(v)
partition_constraints_override_enabled_flag	u(1)
sps_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
sps_log2_diff_min_qt_min_cb_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
sps_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
sps_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(sps_max_mtt_hierarchy_depth_inter_slice != 0) {	
sps_log2_diff_min_qt_min_cb_inter_slice	ue(v)
sps_log2_diff_max_bt_min_qt_inter_slice	ue(v)
sps_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
sps_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)
}	
}	
...	
}	

Modified PH

picture_header_rbsp() {	Descriptor
non_reference_picture_flag	u(1)
...	
if(partition_constraints_override_enabled_flag) {	
partition_constraints_override_flag	u(1)
if(partition_constraints_override_flag) {	
pic_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
pic_log2_diff_min_qt_min_cb_inter_slice	ue(v)
pic_max_mtt_hierarchy_depth_inter_slice	ue(v)
pic_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(pic_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
pic_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
pic_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
pic_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(pic_max_mtt_hierarchy_depth_inter_slice != 0) {	
pic_log2_diff_min_qt_min_cb_inter_slice	ue(v)
pic_log2_diff_max_bt_min_qt_inter_slice	ue(v)
pic_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
pic_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
pic_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if(pic_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
pic_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
pic_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
pic_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)
}	
}	
}	
}	
...	
}	

Proposed Solution 2

- When MTT depth is equal to 0 and the current CU is across the right or bottom picture boundary, quad split is always allowed for partitioning the current CU
 - The minimum QT size is allowed to be greater than the specified minimum CB size when MTT depth is equal to 0
 - The constraints on the minimum QT size is still enforced for the CUs that are entirely inside picture boundaries
 - Quad split is always allowed for partitioning the CUs along picture boundaries

Modifications to Allowed quad split process in VVC Draft 7

- The variable allowSplitQt is derived as follows:
 - If one or more of the following conditions are true, allowSplitQt is set equal to TRUE:
 - $x0 + cbSize$ is greater than `pic_width_in_luma_samples` and `cbSize` is greater than 64
 - $y0 + cbSize$ is greater than `pic_height_in_luma_samples` and `cbSize` is greater than 64
 - $x0 + cbSize$ is greater than `pic_width_in_luma_samples` and `maxMttDepth` is equal to 0
 - $y0 + cbSize$ is greater than `pic_height_in_luma_samples` and `maxMttDepth` is equal to 0
 - Otherwise, if one or more of the following conditions are true, allowSplitQt is set equal to FALSE:
 - `treeType` is equal to `SINGLE_TREE` or `DUAL_TREE_LUMA` and `cbSize` is less than or equal to `MinQtSizeY`
 - `treeType` is equal to `DUAL_TREE_CHROMA` and `cbSize / SubWidthC` is less than or equal to `MinQtSizeC`
 - `mttDepth` is not equal to 0
 - `treeType` is equal to `DUAL_TREE_CHROMA` and $(cbSize / SubWidthC)$ is less than or equal to 4
 - `treeType` is equal to `DUAL_TREE_CHROMA` and `modeType` is equal to `MODE_TYPE_INTRA`
 - Otherwise, allowSplitQt is set equal to TRUE