

JVET-N0497

AHG12: Dependent tiles

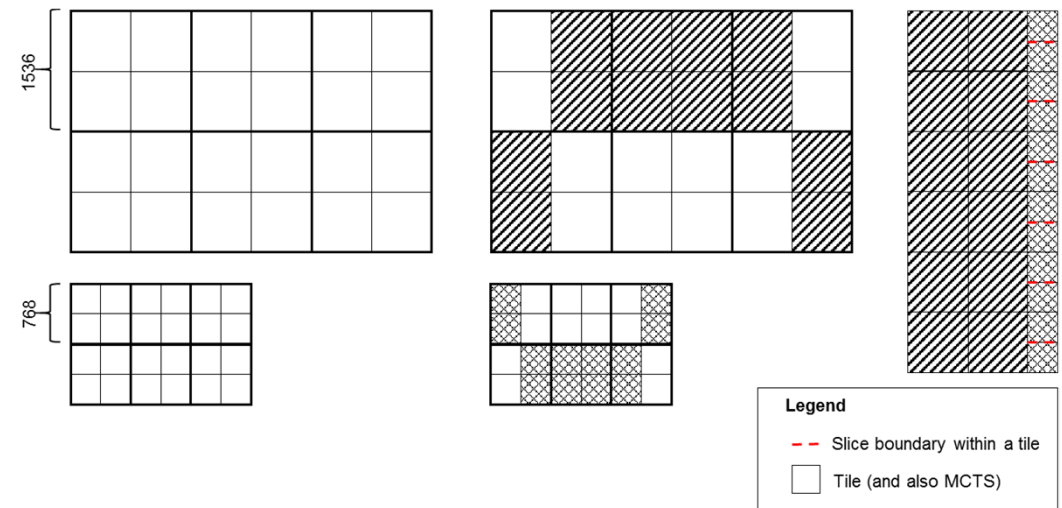


Rickard Sjöberg, Martin Pettersson, Mitra Damghanian
Ericsson

Background – VVC tiles and tile groups



- Tiles in VVC are similar to the tiles in HEVC
- Instead of slice NAL units as in HEVC, VVC uses tile group NAL units, each consisting of an integer number of tiles
- Rectangular tile groups were adopted at the last JVET meeting
- HEVC 360 video streaming with mixed resolution tiles may require a mix of tiles and slices
 - The example from OMAF and VR Industry forum (VRIF) to the right consist of 24 segments realized with a mix of tiles and slices
- In VVC, the example to the right seem to require splitting each large tile horizontally and using rectangular tile groups
 - This splits increases the number of tiles and may degrade coding



Proposal summary



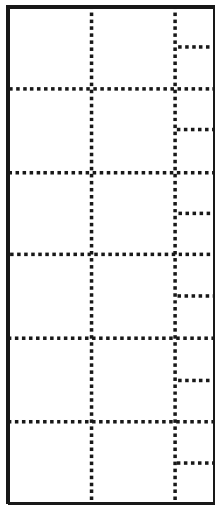
- Add a flag for each tile group that specifies whether the tiles within a tile group are dependent or independent
 - For rectangular tile groups it is proposed that the flags are added to the PPS
 - For raster scan tile groups, it is proposed to add a single flag in the tile group header
- Dependent tiles would enable prediction across tiles within a tile group and only initialize CABAC for the first tile in the tile group
 - All tiles stay byte-aligned, but for dependent tiles CABAC states are continuously updated within a tile group
- The tile groups are always independent, only the internal tile boundaries within a tile group are affected
- The proposal support flexible tile use-cases using a tile merge approach
 - The scan order of CTUs in a picture stay the same regardless if tiles are made dependent or not
 - The amount of proposed changes to the VVC draft is small

A look at the OMAF example

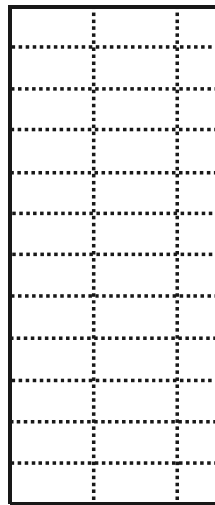


Tile boundaries are shown using dashed lines

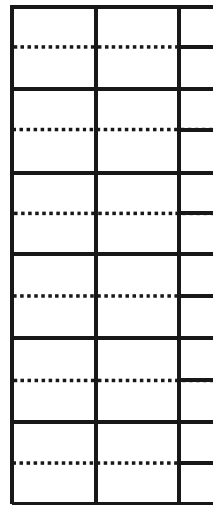
Tile group boundaries are shown using solid lines



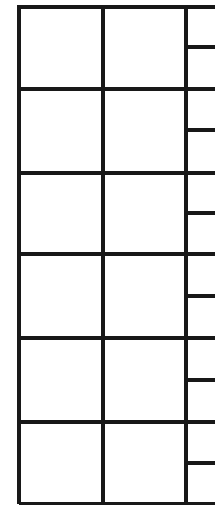
Tile partitioning not supported by VVC



Tile partitioning supported by VVC



Tile groups supported by VVC



Effect of the proposal

Proposed syntax



pic_parameter_set_rbsp() {	
pps_pic_parameter_set_id	ue(v)
...	
if(rect_tile_group_flag) {	
for(i = 0; i <= num_tile_groups_in_pic_minus1; i++)	
dependent_tiles_in_rect_tile_group_flag[i]	u(1)
...	
rbsp_trailing_bits()	
}	

dependent_tiles_in_rect_tile_group_flag[i] equal to 0 specifies that the i-th tile group consists of independent tiles. dependent_tiles_in_rect_tile_group_flag[i] equal to 1 specifies that the i-th tile group consists of dependent tiles.

tile_group_header() {	
tile_group_pic_parameter_set_id	ue(v)
if(rect_tile_group_flag NumTilesInPic > 1)	
tile_group_address	u(v)
if(!rect_tile_group_flag && !single_tile_per_tile_group_flag)	
num_tiles_in_tile_group_minus1	ue(v)
if(!rect_tile_group_flag)	
dependent_tiles_in_tile_group_flag	u(1)
tile_group_type	ue(v)
...	
byte_alignment()	
}	

dependent_tiles_in_tile_group_flag equal to 0 specifies that the tile group consists of independent tiles. dependent_tiles_in_tile_group_flag equal to 1 specifies that the tile group consists of dependent tiles. When not present, the value of dependent_tiles_in_tile_group_flag is inferred to be equal to dependent_tiles_in_rect_tile_group_flag[tileGroupIdx].

Proposed specification text



6.4.4 Derivation process for neighbouring block availability

Inputs to this process are:

- the luma location (x_{Curr} , y_{Curr}) of the top-left sample of the current block relative to the top-left luma sample of the current picture,
- the luma location (x_{NbY} , y_{NbY}) covered by a neighbouring block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring block covering the location (x_{NbY} , y_{NbY}), denoted as $availableN$.

The neighbouring block availability $availableN$ is derived as follows:

- If the neighbouring block is contained in a different tile **group** than the current block, $availableN$ is set equal to FALSE.
- Otherwise, if the neighbouring block is contained in a different tile than the current block and the value of `dependent_tiles_in_tile_group_flag` for the tile is equal to 0, $availableN$ is set equal to FALSE.
- Otherwise, [Ed. (CJ): the derivation process of $availableN$ is to be specified.]

9.5 CABAC parsing process for tile group data

9.5.1 General

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

The initialization process as specified in clause 9.5.2 is invoked when starting the parsing of the CTU syntax specified in clause 7.3.6.2 and the CTU is **either** the first CTU in a tile **group or the first CTU in a tile for which the value of `dependent_tiles_in_tile_group_flag` is equal to 0.**

