



AHG4/AHG9: Syntax modifications for tile width constraint

Chia-Yang Tsai, Chih-Wei Hsu, Yu-Wen Huang, Yung-Chang Chang, Chih-Ming Wang,
Chia-Yun Cheng, Shawmin Lei



Presented by Shawmin Lei
10th JCT-VC Meeting in Stockholm
11-20 July, 2012

Overall Summary

- In HEVC, encoders shall obey a normative constraint on tile width:
 - Tile width shall be equal to or greater than 384 pixels
- Practical decoders may still have to deal with bitstreams that unintentionally violate this constraint.
- Proposed to modify the syntax elements that only legal tile widths can be coded with corresponding codewords
- Better guide encoders to follow the constraint
 - Unintentional violations of the constraint can be avoided.

Introduction

- In HEVC, tile width is specified in PPS
 - With normative constraint on the encoder side
 - Tile width shall be equal to or greater than 384 pixels

pic_parameter_set_rbsp() {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
.....	
tiles or entropy coding sync idc	u(2)
if(tiles or entropy coding sync idc == 1) {	
num tile columns minus1	ue(v)
num tile rows minus1	ue(v)
uniform spacing flag	u(1)
if(!uniform spacing flag) {	
for(i = 0; i < num tile columns minus1; i++)	
column width[i]	ue(v)
for(i = 0; i < num tile rows minus1; i++)	
row height[i]	ue(v)
}	
loop filter across tiles enabled flag	u(1)
} else if(tiles or entropy coding sync idc == 3)	
.....	
rbps_trailing_bits()	
}	

Two syntax elements related to tile width

Problem Definition

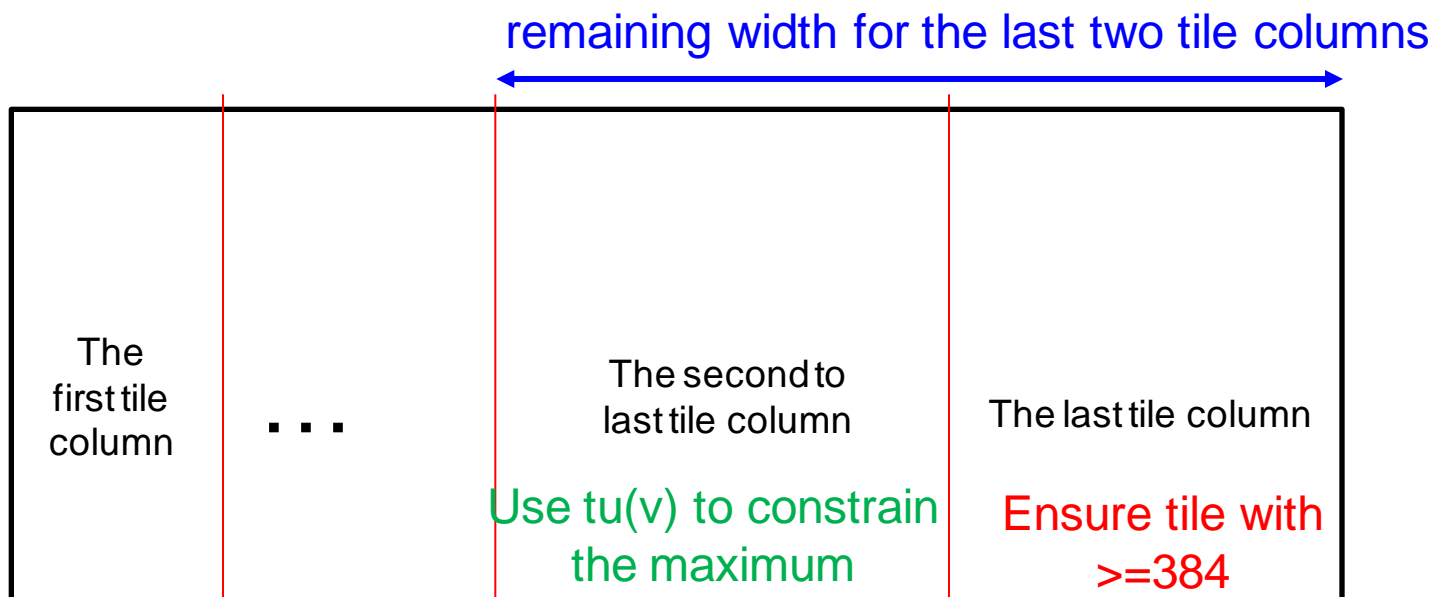
- Only constraint on encoder side
- Decoder is likely to receive illegal bitstreams
 - Decoder complexity is increased in order to handle illegal bitstreams in practical products
 - Worst case storage of tile partitions for an illegal bitstream
 - LCU size: 16x16. Resolution: 7680x4320
 - Total number of tiles: 129600
- Propose only legal tile widths are coded with corresponding codewords
 - Avoid inadvertent violation of the constraint by encoders
 - Relieve decoders from having to deal with this kind of illegal bitstreams.

Proposal for Non-Uniform Tile Partition -1

- Syntax modifications for **column_width**
- Original syntax
 - **column_width**
- Proposed syntax
 - **column_width_minus_min_ctb_num**
 - “min_ctb_num” is equal to “384/CtbSize”
- $\text{tile width} = (\text{column_width_minus_min_ctb_num} + (384/\text{CtbSize})) * \text{CtbSize}$
 - Is always equal to or greater than 384 pixels
- However, the width of the last tile column is not signaled, but is deduced.

Proposal for Non-Uniform Tile Partition -2

- To ensure the widths of the last 2 tile columns are legal, propose syntax for the second to last tile column
 - `second_to_last_column_width_minus_min_ctb_num`
 - Use truncated unary $\text{tu}(v)$ with maximum codeword as follow
 - “remaining width for the last two tile columns” - 384)/CtbSize
 - Ensure tile with of last column is equal to or greater than 384 pixels



Proposal for Uniform Tile Partition

- The number of tile columns determines the tile width.
- Use truncated unary to code number of tile columns
 - Ensure tile width of each uniform-partitioned tiles is equal to or greater than 384 pixels
- Original syntax
 - `num_tile_columns_minus1, ue(v)`
- Proposed syntax
 - `num_tile_columns_minus1, tu(v)`
 - The maximum codeword
 $\text{Floor}(\text{pic_width_in_luma_samples}/384)-1$

Proposed Syntax Table for Tiles

pic_parameter_set_rbsp() {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
.....	
tiles or entropy coding sync idc	u(2)
if(tiles or entropy coding sync idc == 1) {	
num tile columns minus1	tu(v)
num tile rows minus1	ue(v)
uniform spacing flag	u(1)
if(!uniform spacing flag) {	
for(i = 0; i < num tile columns minus1 - 1; i++)	
column width minus min ctb num[i]	ue(v)
if(num tile columns minus1 > 0)	
second last column width minus min ctb num	tu(v)
for(i = 0; i < num tile rows minus1; i++)	
row height[i]	ue(v)
}	
loop filter across tiles enabled flag	u(1)
} else if(tiles or entropy coding sync idc == 3)	
.....	
rbsp_trailing_bits()	
}	

Proposed Syntax Table for Tiles

	Descriptor
pic_parameter_set_rbsp() {	
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
.....	
tiles or entropy coding sync idc	u(2)
if(tiles or entropy coding sync idc == 1) {	
pic_width_in_luma_samples	ue(v)
num_tile_columns_minus1	tu(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
if(num_tile_columns_minus1 > 0) {	
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
}	
for(i = 0; i < num_tile_columns_minus1 - 1; i++)	
column_width_minus_min_ctb_num[i]	ue(v)
if(num_tile_columns_minus1 > 0)	
second_last_column_width_minus_min_ctb_num	tu(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
} else if(tiles or entropy coding sync idc == 3)	
.....	
rbsp_trailing_bits()	
}	

Re-sent (same as in SPS) to prevent from parsing issue

Conclusion

- In this contribution, syntax modifications are proposed
 - Only legal tile widths are coded with corresponding codewords
- Help encoders to follow the constraint which restricts the minimum tile width to 384 pixels
- Relieve decoders from having to deal with the illegal bitstreams that violate the constraint

Recommendation of the BoG

- Minimum tile width = 256
- Minimum tile height = 64
- Undesirable to use truncated unary codes
- J0042 syntax has to be modified accordingly

Proposal for Uniform Spacing

- **num_tile_columns_minus1**
 - Keep using **ue(v)** as the current HEVC text specification
 - Shall be in the range of $[0, \text{maxH}]$, where $\text{maxH} = (\text{PictureWidth}/256) - 1$

- **num_tile_rows_minus1**
 - Keep using **ue(v)** as the current HEVC text specification
 - Shall be in the range of $[0, \text{maxV}]$, where $\text{maxV} = (\text{PictureHeight}/64) - 1$

Proposal for Non-Uniform Spacing (1/2)

- Original syntax
 - **column_width, row_height**
- Proposed syntax
 - **column_width_minus_min_ctb_num_h**, coded by using ue(v)
 - “min_ctb_num_h” is equal to “**256**/CtbSize”
 - tile width =
 $(\text{column_width_minus_min_ctb_num_h} + (\text{256}/\text{CtbSize})) * \text{CtbSize}$
 - **row_height_minus_min_ctb_num_v**, coded by using ue(v)
 - “min_ctb_num_v” is equal to “**64**/CtbSize”
 - tile height = $(\text{row_height_minus_min_ctb_num_v} + (\text{64}/\text{CtbSize})) * \text{CtbSize}$
- The last tile column/row is not signaled

Proposal for Non-Uniform Spacing (2/2)

- Proposed syntax for the second to last tile column
 - **second_to_last_column_width_minus_min_ctb_num_h**
 - tile width =
 $(\text{second_to_last_column_width_minus_min_ctb_num_h} + (256/\text{CtbSize})) * \text{CtbSize}$
 - This syntax element shall be in the range of $[0, \text{maxH}/\text{CtbSize}-4]$, where $\text{maxH} = \text{PictureWidth} - \text{SumOfPreviousColumnWidths} - 256$
- Proposed syntax for the second to last tile row
 - **second_to_last_row_height_minus_min_ctb_num_v**
 - tile row =
 $(\text{second_to_last_row_height_minus_min_ctb_num_v} + (64/\text{CtbSize})) * \text{CtbSize}$
 - This syntax element shall be in the range of $[0, \text{maxV}/\text{CtbSize}-1]$, where $\text{maxV} = \text{PictureHeight} - \text{SumOfPreviousRowHeights} - 64$