



# AHG4: ENABLING DECODER PARALLELISM WITH TILES

JCTVC-10233

RICKARD SJÖBERG

JONATAN SAMUELSSON

JACK ENHORN

ERICSSON AB

# SUMMARY

---

We propose the following changes:

- 1) *To make a separate `tile_info` syntax table that is shared between SPS and PPS*
- 2) *To merge the two PPS flags, `tile_info_present_flag` and `tile_control_present_flag` into one flag: `tile_info_present_in_pps_flag`*
- 3) *To add a flag in the slice header, `use_tile_info_from_pps_flag`, to control whether the tile info from the SPS or the PPS shall be used. The flag is only present if there is both SPS and PPS tile info.*
- 4) To add an SPS flag, `tiles_fixed_structure_flag`, to indicate that the tile info from the SPS is always used. If set to one, we do not parse `use_tile_info_from_pps_flag`.
- 5) To add two SPS flags to indicate that all tiles do have entry point offsets or entry point markers and to include tile id with entry point offsets and markers only if the corresponding flag is set equal to 0.
- 6) To only send `tile_idx_minus_1` for entry point markers if tiles are used (not send them in case of WPP) and change its name to `tile_id_marker_minus1`
- 7) To specify the length and value of `tile_idx_minus_1`
- 8) To add a tile id syntax element, `tile_id_offset_minus1`, for every tile entry point offset
- 9) To move tile parameters derivation text, currently in the semantics section, to a new subclause in the decoding process
- 10) To clarify the semantics for `entry_point_offset`

# PROPOSAL 1 AND 2 – CURRENT SYNTAX

## › Problems:

- Tile syntax is currently repeated in both SPS and PPS with same names
- There is no semantics for the PPS syntax elements
- It is not specified which one to use when tiles are specified in both SPS and PPS (there is a CD note on this)

pic_parameter_set_rbsp( ) {	
if( tiles_or_entropy_coding_sync_idc == 1 ) {	
<b>tile_info_present_flag</b>	u(1)
<b>tile_control_present_flag</b>	u(1)
if( tile_info_present_flag ) {	
<b>num_tile_columns_minus1</b>	ue(v)
<b>num_tile_rows_minus1</b>	ue(v)
<b>uniform_spacing_flag</b>	u(1)
if( !uniform_spacing_flag ) {	
for( i = 0; i < num_tile_columns_minus1; i++ )	
<b>column_width[ i ]</b>	ue(v)
for( i = 0; i < num_tile_rows_minus1; i++ )	
<b>row_height[ i ]</b>	ue(v)
}	
}	
if( tile_control_present_flag )	
<b>loop_filter_across_tiles_enabled_flag</b>	u(1)
}	

sps_parameter_set_rbsp( ) {	
<b>num_tile_columns_minus1</b>	ue(v)
<b>num_tile_rows_minus1</b>	ue(v)
<b>uniform_spacing_flag</b>	u(1)
if( !uniform_spacing_flag ) {	
for( i = 0; i < num_tile_columns_minus1; i++ )	
<b>column_width[ i ]</b>	ue(v)
for( i = 0; i < num_tile_rows_minus1; i++ )	
<b>row_height[ i ]</b>	ue(v)
}	
<b>loop_filter_across_tiles_enabled_flag</b>	u(1)
}	

# PROPOSAL 1 AND 2 – NEW SYNTAX

- › We propose a separate tile\_info syntax table that is shared between SPS and PPS
- › We propose to merge the two PPS flags, tile\_info\_present\_flag and tile\_control\_present\_flag into one flag: tile\_info\_present\_in\_pps\_flag
  - We see no real use case for applying loop\_filter\_across\_tiles\_enabled\_flag from the PPS while applying the other tile parameters from the SPS, or the other way around.

tile_info( InPpsFlag ) {	Descriptor
<b>num_tile_columns_minus1</b> [ InPpsFlag ]	ue(v)
<b>num_tile_rows_minus1</b> [ InPpsFlag ]	ue(v)
<b>uniform_spacing_flag</b> [ InPpsFlag ]	u(1)
if( !uniform_spacing_flag ) {	
for( i = 0; i < num_tile_columns_minus1; i++ )	
<b>column_width</b> [ InPpsFlag ][ i ]	ue(v)
for( i = 0; i < num_tile_rows_minus1; i++ )	
<b>row_height</b> [ InPpsFlag ][ i ]	ue(v)
}	
<b>loop_filter_across_tiles_enabled_flag</b> [ InPpsFlag ]	u(1)
}	

seq_parameter_set_rbsp( ) {
if( tiles_or_entropy_coding_sync_idc == 1 )
tile_info( 0 )

pic_parameter_set_rbsp( ) {
<b>tile_info_present_in_pps_flag</b>
if( tile_info_present_in_pps_flag )
tile_info( 1 )

**tile\_info\_present\_in\_pps\_flag** equal to 1 specifies that tile\_info() is present in the picture parameter set. tile\_info\_present\_in\_pps\_flag equal to 0 specifies that tile\_info() is not present in the picture parameter set.

## 7.4.2.3 Tile info semantics

Tile info may be present in a sequence parameter set or in a picture parameter set. If tile info is present in a sequence parameter set InPpsFlag shall be equal to 0, otherwise it shall be equal to 1.

# PROPOSAL 3

- › We propose to add a flag in the slice header, `use_tile_info_from_pps_flag`, to control whether the tile info from the SPS or the PPS shall be used. The flag is only present if there is both SPS and PPS tile info.

slice_header() {	Descriptor
...	
if( tiles_or_entropy_coding_sync_idc > 0 ) {	
if( tile_info_present_in_pps_flag )	
<b>use_tile_info_from_pps_flag</b>	u(1)

**use\_tile\_info\_from\_pps\_flag** equal to 1 specifies that tile info from the picture parameter set is used for the current picture. **use\_tile\_info\_from\_pps\_flag** equal to 0 specifies that tile info from the sequence parameter set is used for the current picture. When **use\_tile\_info\_from\_pps\_flag** is not present it shall be inferred to be equal to 0.

The variable `NumTilesInPic` is derived as follows.

$$\text{NumTilesInPic} = ( \text{num\_tile\_columns\_minus1}[ \text{use\_tile\_info\_from\_pps\_flag} ] + 1 ) * ( \text{num\_tile\_rows\_minus1}[ \text{use\_tile\_info\_from\_pps\_flag} ] + 1 )$$

# PROPOSAL 4

- › Problem:
  - A decoder does not know that an entire coded video sequence (CVS) is decodable with tiles since the tile structure may be changed or removed in-band.
- › Proposal:
  - We propose to add an SPS flag, `tiles_fixed_structure_flag`, to indicate that the tile info from the SPS is always used for all pictures in the CVS. If set to one, we do not parse `use_tile_info_from_pps_flag`.

<code>seq_parameter_set_rbsp() {</code>	<b>Descriptor</b>
<code>...</code>	
<code>if( tiles_or_entropy_coding_sync_idc == 1 ) {</code>	
<b><code>tiles_fixed_structure_flag</code></b>	<code>u(1)</code>

**`tiles_fixed_structure_flag`** equal to 1 specifies that `tile_info()` shall not be present in any picture parameter set with a `seq_parameter_set_id` that references this sequence parameter set. `tiles_fixed_structure_flag` equal to 0 indicates that `tile_info()` may be present in a picture parameter set with a `seq_parameter_set_id` that references this sequence parameter set.

<code>slice_header() {</code>	<b>Descriptor</b>
<code>...</code>	
<code>if( tiles_or_entropy_coding_sync_idc &gt; 0 ) {</code>	
<code>if( <b><code>!tiles_fixed_structure_flag</code></b> &amp;&amp; <code>tile_info_present_in_pps_flag</code> )</code>	
<b><code>use_tile_info_from_pps_flag</code></b>	<code>u(1)</code>

# PROPOSAL 5

- › Problem:
  - Parallel decoding with tiles can not be done if there are tiles in the bitstream that have no entry points. A decoder would not know the spatial location of those tiles.
- › Proposal:
  - 1) Add flags in the SPS to indicate that there are entry points offsets of markers for all tiles in the bitstream. Together with `tiles_fixed_structure_flag`, a decoder will know from the SPS whether it can decode the whole stream in parallel using tiles.
  - 2) Make presence of `tile_id_offset_minus1` conditioned on `entry_point_offsets_for_all_tiles_flag` and `tile_id_marker_minus1` conditioned on `entry_point_markers_for_all_tiles_flag`. If flag is set, there is no need to send `tile_id`.

<code>seq_parameter_set_rbsp() {</code>	<b>Descriptor</b>
<code>...</code>	
<code>if( tiles_or_entropy_coding_sync_idc == 1 ) {</code>	
<b><code>tiles_fixed_structure_flag</code></b>	<code>u(1)</code>
<b><code>entry_point_offsets_for_all_tiles_flag</code></b>	<code>u(1)</code>
<b><code>entry_point_markers_for_all_tiles_flag</code></b>	<code>u(1)</code>
<code>tile_info( 0 )</code>	

- › **`entry_point_offsets_for_all_tiles_flag`** equal to 1 indicates that the codeword `entry_point_offset[ i ]` is present for every tile in the coded video sequence except for tiles that are the first tile in a slice.
- › **`entry_point_markers_for_all_tiles_flag`** equal to 1 indicates that there is an `entry_point_marker_two_3bytes` codeword indicating the start of every tile in the coded video sequence except for tiles that are the first tile in a slice.

# PROPOSAL 6

- › Problem:
  - tile id entry point markers are sent when WPP is on
- › Proposal
  - 1) Only send tile id for entry point markers if tiles are used (not send them in case of WPP)
  - 2) Change the name of the syntax element from tile\_idx\_minus\_1 to tile\_id\_marker\_minus1.

slice_data( ) {	Descriptor
...	
rbsp_trailing_bits( )	
if( nextbits(24) == 0x000002 ) {	
<b>entry_point_marker_two_3bytes</b>	f(24)
if(tiles_or_entropy_coding_sync_idc == 1 && !entry_point_markers_for_all_tiles_flag )	
<b>tile_idx_</b> <del>marker</del> <b>_minus_1</b>	u(v)
}	
}	
} while( moreDataFlag )	
}	



# PROPOSAL 7

- › Problem:
  - tile\_id\_marker\_minus1 is u(v) but its length is unspecified
  - The value of tile\_id\_marker\_minus\_1 should be specified
- › Proposal: Correct the semantics of tile\_id\_minus1

slice_data( ) {	<b>Descriptor</b>
...	
rbsp_trailing_bits( )	
if( nextbits(24) == 0x000002 ) {	
<b>entry_point_marker_two_3bytes</b>	f(24)
if(tiles_or_entropy_coding_sync_idc == 1 && !entry_point_markers_for_all_tiles_flag )	
<b>tile_id_marker_minus1</b>	u(v)
}	
}	
} while( moreDataFlag )	
}	

## Current CD text:

**tile\_idx\_minus\_1** specifies the TileID in raster scan order. The first tile in the picture shall have a TileID of 0. The value of tile\_idx\_minus\_1 shall be in the range of 0 to  $(\text{num\_tile\_columns\_minus1} + 1) * (\text{num\_tile\_rows\_minus1} + 1) - 1$ .

## Proposed text:

**tile\_id\_marker\_minus1** specifies an ID of a tile in raster scan order and shall be represented by  $\text{Ceil}(\text{Log2}(\text{NumTilesInPic} - 1))$  bits. tile\_id\_marker\_minus1 shall be equal to  $\text{TileId}[\text{CtbAddrTS}] - 1$ .

# PROPOSAL 8

- › Problem:
  - There are no tile\_id syntax element for entry point offsets. If there are no tile\_id syntax elements and not all tiles have entry points, a decoder will not know the spatial location of the tiles. (Tile id is always sent for entry point markers)
- › Proposal: Add tile\_id\_offset\_minus1 to the specification

slice_header() {	Descriptor
...	
if( tiles_or_entropy_coding_sync_idc > 0 ) {	
if( !tiles_fixed_structure_flag && tile_info_present_in_pps_flag )	
<b>use_tile_info_from_pps_flag</b>	u(1)
<b>num_entry_point_offsets</b>	ue(v)
if( num_entry_point_offsets > 0 ) {	
<b>offset_len_minus1</b>	ue(v)
for( i = 0; i < num_entry_point_offsets; i++ ) {	
if( tiles_or_entropy_coding_sync_idc == 1 && !entry_point_offsets_for_all_tiles_flag )	
<b>tile_id_offset_minus1[ i ]</b>	u(v)
<b>entry_point_offset[ i ]</b>	u(v)
}	
}	
}	

**tile\_id\_offset\_minus1[ i ]** specifies an ID of a tile in raster scan order and shall be represented by  $\text{Ceil}(\text{Log}_2(\text{NumTilesInPic} - 1))$  bits. When tile\_id\_offset\_minus1 is not present it shall be inferred to be equal to i.

# PROPOSAL 9

---

- › Problem:
  - There is some text in the tile semantics that fits better in the process section of the specification (there is an editorial note in the CD about this).
- › Proposal:
  - We propose that the variables ColumnWidth, ColumnWidthInLumaSamples, ColBD, RowHeight and RowBd, currently specified in the semantics section, are moved to a new section "Decoding process for tiles".
- › Proposed text on the next slide, black text is copied from tile semantics section, red text is additions.

# PROPOSAL 9

## 8.3.4 Decoding process for tiles

This process is invoked once per picture, after decoding of a slice header but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction of the slice as specified in subclause 8.3.5.

*The variable  $X$  is set equal to `use_tile_info_from_pps_flag`.*

Values of `ColumnWidth[ i ]`, specifying the width of the  $i$ -th tile column in units of coding treeblocks, and the values of `ColumnWidthInLumaSamples[ i ]`, specifying the width of the  $i$ -th tile column in units of luma samples, are derived as follows:

```
for( i = 0; i <= num_tile_columns_minus1; i++ ) {
    if( uniform_spacing_flag )
        ColumnWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbs ) / ( num_tile_columns_minus1[ X ] + 1 ) -
            ( i * PicWidthInCtbs ) / ( num_tile_columns_minus1[ X ] + 1 )
    else
        ColumnWidth[ i ] = column_width[ X ][ i ]
    ColumnWidthInLumaSamples[ i ] = ColumnWidth[ i ] << Log2CtbSize
}
```

Values of `RowHeight[ i ]`, specifying the height of the  $i$ -th tile row in units of coding treeblocks, are derived as follows:

```
for( i = 0; i <= num_tile_rows_minus1; i++ )
    if( uniform_spacing_flag )
        RowHeight[ i ] = ( ( i + 1 ) * PicHeightInCtbs ) / ( num_tile_rows_minus1[ X ] + 1 ) -
            ( i * PicHeightInCtbs ) / ( num_tile_rows_minus1[ X ] + 1 )
    else
        RowHeight[ i ] = row_height[ X ][ i ]
```

Values of `ColBd[ i ]`, specifying the location of the left column boundary of the  $i$ -th tile column in units of coding treeblocks, are derived as follows:

```
for( ColBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1[ X ]; i++ )
    ColBd[ i + 1 ] = ColBd[ i ] + ColumnWidth[ i ]
```

Values of `RowBd[ i ]`, specifying the location of the top row boundary of the  $i$ -th tile row in units of coding treeblocks, are derived as follows:

```
for( RowBd[ 0 ] = 0, i = 0; i <= num_tile_rows_minus1[ X ]; i++ )
    RowBd[ i + 1 ] = RowBd[ i ] + RowHeight[ i ]
```

# PROPOSAL 10

---

- › Problem:
  - Semantics for entry\_point\_offset seems overly complicated
- › Proposal:
  - Replace the semantics with something clearer

## Current CD text:

**entry\_point\_offset**[ *i* ] specifies the *i*-th entry point offset, in bytes and shall be represented by offset\_len\_minus1 plus 1 bits. The coded slice NAL unit consists of num\_entry\_point\_offsets + 1 subsets, with subset index values ranging from 0 to num\_entry\_point\_offsets, inclusive. Subset 0 consists of bytes 0 to entry\_point\_offset[ 0 ] – 1, inclusive, of the coded slice NAL unit, subset *k*, with *k* in the range of 1 to num\_entry\_point\_offsets – 1, inclusive, consists of bytes entry\_point\_offset[ *k* – 1 ] to entry\_point\_offset[ *k* ] + entry\_point\_offset[ *k* – 1 ] – 1, inclusive, of the coded slice NAL unit, and the last subset (with subset index equal to num\_entry\_point\_offsets) consists of the remaining bytes of the coded slice NAL unit.

**NOTE** – The NAL unit header and the slice header of a coded slice NAL unit are always included in subset 0.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 1 and num\_entry\_point\_offsets is greater than 0, each subset shall contain all coded bits of one or multiple complete tiles, and the number of subsets shall be equal to or less than the number of tiles in the slice.

When tiles\_or\_entropy\_coding\_sync\_idc is equal to 2 and num\_entry\_point\_offsets is greater than 0, subset *k*, for each of all the possible *k* values, shall contain all bits to be used during the initialization process for the current bitstream pointer *k*.

# PROPOSAL 10

---

- › Problem:
  - Semantics for `entry_point_offset` seems over complicated
- › Proposal:
  - Replace the semantics with something clearer

## **Proposal:**

**`entry_point_offset[ i ]`** specifies the *i*-th entry point offset, in bytes and shall be represented by `offset_len_minus1` plus 1 bits.

When `tiles_or_entropy_coding_sync_idc` is equal to 1 **`entry_point_offset[ i ]` shall be equal to the offset in bytes, relative to the start of the slice header, for the first treeblock, with address `CtbAddrTS`, such that  $\text{TileId}[ \text{CtbAddrTS} ] = \text{tile\_id\_offset\_minus1}[ i ] + 1$ .**

When `tiles_or_entropy_coding_sync_idc` is equal to 2 and `num_entry_point_offsets` is greater than 0, subset *k*, for each of all the possible *k* values, shall contain all bits to be used during the initialization process for the current bitstream pointer *k*.

# SUMMARY

---

We propose the following changes:

- 1) *To make a separate tile\_info syntax table that is shared between SPS and PPS*
- 2) *To merge the two PPS flags, tile\_info\_present\_flag and tile\_control\_present\_flag into one flag: tile\_info\_present\_in\_pps\_flag*
- 3) *To add a flag in the slice header, use\_tile\_info\_from\_pps\_flag, to control whether the tile info from the SPS or the PPS shall be used. The flag is only present if there is both SPS and PPS tile info.*
- 4) To add an SPS flag, tiles\_fixed\_structure\_flag, to indicate that the tile info from the SPS is always used. If set to one, we do not parse use\_tile\_info\_from\_pps\_flag.
- 5) To add two SPS flags to indicate that all tiles do have entry point offsets or entry point markers and to include tile id with entry point offsets and markers only if the corresponding flag is set equal to 0.
- 6) To only send tile\_idx\_minus\_1 for entry point markers if tiles are used (not send them in case of WPP) and change its name to tile\_id\_marker\_minus1
- 7) To specify the length and value of tile\_idx\_minus\_1
- 8) To add a tile id syntax element, tile\_id\_offset\_minus1, for every tile entry point offset
- 9) To move tile parameters derivation text, currently in the semantics section, to a new subclause in the decoding process
- 10) To clarify the semantics for entry\_point\_offset



**ERICSSON**