

Specifying entry points to facilitate different decoder implementations (JCTVC-I0237)

Wade Wan and Peisong Chen

Broadcom Corporation



- Harmonized method for signaling entry point of tiles and wavefront substreams
 - Presented in H0556 (Hendry et al., “AHG4: Harmonized Method for Signalling Entry Points of Tiles and WPP Substreams”)
 - Adopted after minor refinements in H0737 (Wang et al., “Text for tiles, WPP and entropy slices”)
- Current definition of entry points (num_entry_point_offsets) in CD allows an encoder to choose to transmit **none, some or all** possible entry points.
 - Idea was attempt to support a scenario where an encoder using N cores knows that it could provide (only) M entry points to a decoder (where $M < N$)

- Problematic:
 - For multicast environments
 - When encoder does not have a priori knowledge of a decoder's architecture
- Significant possibility of complicating interoperability by allowing this flexibility/ambiguity
- Examples:
 - Single core decoder attempting to support tiles with raster scan decoding
 - A four code decoder attempting to decode a stream generated with four wavefronts but given only two entry points

- Mandate that the entry point of every tile and every wavefront substream in a bitstream be explicitly signaled
- Can be implemented in three parts:
 1. Changes to num_entry_point_offsets semantics
 2. Changes to entry_point_offset semantics
 3. Clarification and changes to slice header syntax

Changes to num_entry_point_offsets semantics



- Current text:

`num_entry_point_offsets` specifies the number of `entry_point_offset[i]` syntax elements in the slice header. When `tiles_or_entropy_coding_sync_idc` is equal to 1, the value of `num_entry_point_offsets` shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive. When `tiles_or_entropy_coding_sync_idc` is equal to 2, the value of `num_entry_point_offsets` shall be in the range of 0 to `num_substreams_minus1`, inclusive. When not present, the value of `num_entry_point_offsets` is inferred to be equal to 0.

- Proposed text:

`num_entry_point_offsets` specifies the number of `entry_point_offset[i]` syntax elements in the slice header. When `tiles_or_entropy_coding_sync_idc` is equal to 1, the value of `num_entry_point_offsets` shall be ~~in the range of 0 to~~ $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, ~~inclusive~~. When `tiles_or_entropy_coding_sync_idc` is equal to 2, the value of `num_entry_point_offsets` shall be ~~in the range of 0 to~~ `num_substreams_minus1`, ~~inclusive~~. When not present, the value of `num_entry_point_offsets` is inferred to be equal to 0.

- Reason: Require every possible entry point to be explicitly signaled

Changes to entry_point_offset semantics (1/2)

- Current 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.

- Proposed 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.

- Reason: Believe to be typo on bytes corresponding to an entry point

Changes to entry_point_offset semantics (2/2)

- Current text:

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.

- Proposed text:

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 **a one or multiple complete tiles**, and the number of subsets shall be equal to or less than the number of tiles in the slice.

- Reason: Each tile must have an associated entry point independent of whether it is part of a slice consisting of multiple complete tiles.

Clarification and changes to slice header syntax (1/2)



- Current syntax for entry points appear to be sufficient when there is only one slice in a picture
- Unclear if there are multiple slices in a picture?
- Assumption that all entry points should be transmitted at beginning of picture to facilitate a decoder quickly determining entry points
 - Enable transmission of all entry points with very first slice header
 - Not necessary to repeat this information for subsequent slice headers

Clarification and changes to slice header syntax (2/2)

slice_header() {	Descriptor
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag == 0)	
slice_address	u(v)
slice_type	ue(v)
...	
...	
if ((tiles_or_entropy_coding_sync_idc > 0) && (first_slice_in_pic_flag == 0)) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset[i]	u(v)
}	
}	
}	

Thank you!

Wade Wan and Peisong Chen

Broadcom Corporation

