|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  11th Meeting: Shanghai, China,10 -19 October, 2012 | Document: JCTVC-K0200  M26527 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG4: Sub-stream entry points SEI message** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Minhua Zhou Texas Instruments Inc., USA | Tel: Email:  : | +1-214-480-3816 [zhou@ti.com](mailto:zhou@ti.com) |
| *Source:* | Texas Instruments Inc; | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

In the current HM8.0 design, sub-stream entries are signaled in slice header only to facilitate sub-stream switching of decoding parallelized streams with single core decoders. No entry points are available in the bitstream to assist parallel decoding on multiple core platforms. This contribution advocates signaling of sub-stream entry points in a SEI message as well to facilitate parallel decoding. Each sub-stream entry point shall be a parallel decoding entry point and start with a slice header with dependent\_slice\_flag set equal to 0. It is also signaled whether sub-streams have approximately same size in terms of CTBs or compressed bitstream bytes. The number of sub-stream entry points signaled in the SEI message is not necessarily equal to the number of WPPs, tiles or slices in the picture associated with the SEI message. It is recommended to add this SEI message to the spec to facilitate parallel decoding in environments in which encoder and decoder negotiation might be possible, and provides extendibility of supporting parallel decoding in which parallel tools might need to be mandated.

# Introduction

In the current HM8.0 design, sub-stream entries are signaled in slice header to facilitate sub-stream switching of decoding parallelized streams with single core decoders. No entry points are available in the bitstream to assist parallel decoding on multiple core platforms. For example, in Fig. 1 if a picture is divided into tiles and each tile contains multiple slices, then there will be no sub-stream entries present in the bitstream based on the current design. However, in this particular example, tile entry points are needed for a multi-core decoder to be able to dispatch the bitstreams to cores in parallel for parallel processing purpose.

**Fig. 1 – an example of dividing a picture into four tiles, and each tile contains multiple slices**

In JCTVC-J0081 [3] it is proposed to add this picture-level sub-stream entry points signaling in APS. Since APS is removed from the Main Profile, it might be more appropriate to carry the sub-stream entry points information in a SEI message.

# Proposed solution

To facilitate the parallel decoding on multi-core platforms, we propose to add signaling of the sub-stream entry points at picture-level. Each picture is divided into a number of sub-pictures, and each sub-picture must start with a regular slice header to break dependency among sub-pictures. A sub-picture can be made up of a number of tiles, wavefronts, regular slices or combinations of those tools. After compression, each sub-picture becomes a sub-stream, and stitching sub-streams together in sub-picture raster-scan order forms the bitstream for the picture.

The sub-stream (i.e. sub-picture) entry points for a picture are signaled at picture level with a supplemental enhancement information (SEI) message. In addtion to signaling the entry points, the SEI message may also contain the information on how sub-pictures are divided. For example, sub-pictures may contain approximately same number of CTBs for a multi-core decoder to achieve sample-rate balancing among acores, or constain approximately same amount of bitstream bytes for a multi-core decoder to achieve bit-rate balancing among acores. The syntax specifiaction for the proposed SEI message is provided in Table 1.

|  |  |
| --- | --- |
| sub\_stream\_entry\_points( ) { | Descriptor |
| **sub\_stream\_entry\_point\_present\_flag** | u(1) |
| if (sub\_stream\_entry\_point\_present\_flag) { |  |
| **num\_sub\_stream\_entry\_point\_offsets\_minus1** | ue(v) |
| if(num\_sub\_stream\_entry\_point\_offsets\_minus1 > 0 ) { |  |
| **sub\_stream\_type** | ue(v) |
| **offset\_len\_minus1** | ue(v) |
| for( i = 0; i < num\_sub\_stream\_entry\_point\_minus1; i++ ) |  |
| **sub\_stream\_entry\_point\_offset**[ i +1] | u(v) |
| } |  |
| } |  |

**Table 1. Syntax specification for the proposed sub-stream entry points SEI message**

The **semantics** of the syntax elements for the proposed sub-stream entry points SEI message are defined as follows:

* **sub\_stream\_entry\_point\_present\_flag** equal to 1 specifies that sub-stream entry points information is present in the SEI message. Otherwise, sub-stream entry point information is not present in the SEI message.
* **num\_sub\_stream\_entry\_point\_offsets\_minus1** specifies the number of sub\_stream\_entry\_point\_offset[i+1] syntax elements in the SEI message. The value of num\_sub\_stream\_entry\_point\_offsets\_minus1 shall be equal to the number of sub-streams in the picture associated with the SEI message minus 1. When not present, the value of num\_sub\_stream\_entry\_point\_offsets\_minus1 is inferred to be equal to 0.
* **sub\_stream\_type** equal to 1 specifies that sub-streams are uniformly spaced in terms of number of CTBs, sub\_stream\_type equal to 2 specifies that sub-streams are uniformly spaced in terms of compressed bitstream bytes, sub\_stream\_type equal to 0 specifies that sub-streams are neither uniformly spaced in terms of number of CTBs nor uniformly spaced in terms of number of compressed bitstream bytes. sub\_stream\_type shall be in the range of 0 to 2, inclusive.
* **offset\_len\_minus1** plus 1 specifies the length, in bytes, of the sub\_stream\_entry\_point\_offset[ i + 1] syntax elements. The value of offset\_len\_minus1 shall be in the range of 0 to 31, inclusive.
* **sub\_stream\_entry\_point\_offset**[ i + 1] specifies the (i+1)-th entry point offset, in bytes, relative to the i-th sub-stream entry point. The 0-th offset sub\_stream\_entry\_point\_offset[0] is inferred to 0.

**Fig. 2 – Example of uniformly spaced 2x2 sub-picture partitioning with tiles**

Fig. 2 shows an example in which a picture is evenly divided into 2x2 sub-pictures by using the tile syntax. In this case, a picture is evenly divided into tiles by factor of 2 in vertical direction, while in the horizontal direction it is unevenly divided into 4 tiles, but there is one right-hand tile column boundary is coincident with the sub-picture boundary, which is right-hand column boundary of Tile02 and Tile21. Each sub-picture (sub-stream) must start with a regular slice header. In the bitstream, it is signaled that Tile00 is entry point for sub-picture 0, Tile10 is entry point for sub-picture 1, Tile20 is entry point for sub-picture 2 and Tile30 is entry point for sub-picture 3.

**Fig. 3 – Diagram of sub-stream entry point signaling for sub-picture partitioning given in Fig. 2**

Fig. 3 depicts the sub-stream entry points signaling for the sub-picture partitioning example illustrated in Fig. 2. In this example, a picture is divided into four sub-pictures of equal size, then

* **sub\_stream\_entry\_present\_flag** is set to 1
* **num\_sub\_stream\_entry\_point\_offsets\_minus1** is set to 3
* **sub\_stream\_type** is set to 1
* **sub\_stream\_entry\_point\_offset**[0] is inferred to 0
* **sub\_stream\_entry\_point\_offset**[1] is set equal to the bitstream size of sub-picture 0
* **sub\_stream\_entry\_point\_offset**[2] is set equal to the bitstream size of sub-picture 1
* **sub\_stream\_entry\_point\_offset**[3] is set equal to the bitstream size of sub-picture 2

# References

[1] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC Document, JCTVC-J1100, Stockholm, Sweden, July 2012.

[2] [B. Bross](mailto:benjamin.bross@hhi.fraunhofer.de), [W.-J. Han](mailto:wjhan.han@samsung.com), [J.-R. Ohm](mailto:ohm@ient.rwth-aachen.de), [G. J. Sullivan](mailto:garysull@microsoft.com), [T. Wiegand](mailto:thomas.wiegand@hhi.fraunhofer.de) “High Efficiency Video Coding (HEVC) Test Model 8 (HM 8) Encoder Description” JCT-VC Document, JCTVC-J1003, Stockholm, Sweden, July 2012.

[3] M. Zhou, “AHG4: Signaling sub-stream entries in APS for parallel decoding,” ” JCT-VC Document, JCTVC-J1081, Stockholm, Sweden, July 2012.

# Patent rights declaration(s)

**Texas Instruments, Inc. may have IPR relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation |ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

# CD text

1. Add sub\_stream\_entry\_points to Table D.1 SEI payload syntax

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | Descriptor |
| if( payloadType = = 0 ) |  |
| buffering\_period( payloadSize ) |  |
| else if( payloadType = = 1 ) |  |
| pic\_timing( payloadSize ) |  |
| else if( payloadType = = 2 ) |  |
| pan\_scan\_rect( payloadSize ) |  |
| else if( payloadType = = 3 ) |  |
| filler\_payload( payloadSize ) |  |
| else if( payloadType = = 4 ) |  |
| user\_data\_registered\_itu\_t\_t35( payloadSize ) |  |
| else if( payloadType = = 5 ) |  |
| user\_data\_unregistered( payloadSize ) |  |
| else if( payloadType = = 6 ) |  |
| recovery\_point( payloadSize ) |  |
| else if( payloadType = = 9 ) |  |
| scene\_info( payloadSize ) |  |
| else if( payloadType = = 15 ) |  |
| full\_frame\_snapshot( payloadSize ) |  |
| else if( payloadType = = 16 ) |  |
| progressive\_refinement\_segment\_start( payloadSize ) |  |
| else if( payloadType = = 17 ) |  |
| progressive\_refinement\_segment\_end( payloadSize ) |  |
| else if( payloadType = = 19 ) |  |
| film\_grain\_characteristics( payloadSize ) |  |
| else if( payloadType = = 20 ) |  |
| deblocking\_filter\_display\_preference( payloadSize ) |  |
| else if( payloadType = = 22 ) |  |
| post\_filter\_hint( payloadSize ) |  |
| else if( payloadType = = 23 ) |  |
| tone\_mapping\_info( payloadSize ) |  |
| else if( payloadType = = 45 ) |  |
| frame\_packing\_arrangement( payloadSize ) |  |
| else if( payloadType = = 47 ) [Ed. (GJS): Check numbering w.r.t. AVC.] |  |
| display\_orientation( payloadSize ) |  |
| else if( payloadType = = 128 ) |  |
| sop\_description( payloadSize ) |  |
| else if( payloadType = = 129 ) |  |
| field\_indication( payloadSize ) |  |
| else if( payloadType = = 130) |  |
| decoded\_picture\_hash( payloadSize ) |  |
| else if( payloadType = = 131) |  |
| active\_parameter\_sets( payloadSize ) |  |
| else if( payloadType = = 132 ) |  |
| sub\_pic\_timing( payloadSize ) |  |
| else if (payloadType == 133) |  |
| thread\_entry\_points(payloadSize) |  |
| else |  |
| reserved\_sei\_message( payloadSize ) |  |
| if( !byte\_aligned( ) ) { |  |
| **bit\_equal\_to\_one** /\* equal to 1 \*/ | f(1) |
| while( !byte\_aligned( ) ) |  |
| **bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| } |  |
| } |  |

1. Replace

D.1.23 Reserved SEI message syntax

|  |  |
| --- | --- |
| reserved\_sei\_message( payloadSize ) { | Descriptor |
| for( i = 0; i < payloadSize; i++ ) |  |
| **reserved\_sei\_message\_payload\_byte** | b(8) |
| } |  |

With

D.1.23 Thread entry points SEI message syntax

|  |  |
| --- | --- |
| thread\_entry\_points( payloadSize) { | Descriptor |
| **num\_thread\_entry\_point\_offsets\_minus1** | ue(v) |
| if(num\_thread\_entry\_point\_offsets\_minus1 > 0 ) { |  |
| **thread\_ uniform\_spacing\_flag** | u(1) |
| **offset\_len\_minus1** | ue(v) |
| for( i = 0; i < num\_thread\_entry\_point\_minus1; i++ ) |  |
| **thread\_entry\_point\_offset**[ i +1] | u(v) |
| } |  |

D.1.24 Reserved SEI message syntax

|  |  |
| --- | --- |
| reserved\_sei\_message( payloadSize ) { | Descriptor |
| for( i = 0; i < payloadSize; i++ ) |  |
| **reserved\_sei\_message\_payload\_byte** | b(8) |
| } |  |

1. Add sub-stream entry points SEI message semantics

D.2.23 Thread entry points SEI message semantics

The thread entry points SEI message provides picture-level thread entropy points information for an access unitassociated with the SEI message. Each thread shall contain an integer number of NAL units and shall start with a slice header with dependent\_slice\_flag set equal to 0.

**num\_thread\_entry\_point\_offsets\_minus1** specifies the number of thread\_entry\_point\_offset[i+1] syntax elements in the SEI message. The value of num\_thread\_entry\_point\_offsets\_minus1 shall be equal to the number of threads in the access unit associated with the SEI message minus 1. num\_thread\_entry\_point\_offsets\_minus1 equal to 0 specifies that the thread entry point information is not present in the SEI message.

**thread\_ uniform\_spacing\_flag** equal to 1 specifies that threads are uniformly spaced in terms of number of CTBs, thread\_uniform\_spacing\_flag equal to 0 specifies that threads are not uniformly spaced in terms of in terms of number of CTBs. When thread\_ uniform\_spacing\_flag is equal to 1, all the threads in the access unit shall contain number of CTBs less than or equal to

(PicHeightInCtbsY/(num\_thread\_entry\_point\_offsets\_minus1+1) + (PicHeightInCtbsY%( num\_thread\_entry\_point\_offsets\_minus1+1))? 1: 0 )\* PicWidthInCtbsY.

**offset\_len\_minus1** plus 1 specifies the length, in bytes, of the thread\_entry\_point\_offset[ i + 1] syntax elements. The value of offset\_len\_minus1 shall be in the range of 0 to 31, inclusive.

**thread\_entry\_point\_offset[ i + 1]** specifies the (i+1)-th thread entry point offset in bytes, and is represented by offset\_len\_minus1 plus 1 bits. thread\_entry\_point\_offset[0] is inferred to equal to 0 and points to the first byte of the access unit.The access unit consists of num\_thread\_entry\_point\_offsets\_minus1+ 1 subsets, with subset index values ranging from 0 to num\_thread\_entry\_point\_offsets\_minus1, inclusive. The first byte of the access unit is considered byte 0. When present, emulation prevention bytes that appear in the access unit are counted as part of the picture data for purposes of subset identification. Subset 0 consists of bytes 0 to thread\_entry\_point\_offset[ 1 ] − 1, inclusive, of the acess unit, subset k, with k in the range of 1 to num\_thread\_entry\_point\_offsets\_minus1 - 1, inclusive, consists of bytes firstByte[ k ] to lastByte[ k ], inclusive, of the acess unit with firstByte[ k ] and lastByte[ k ] defined as:

firstByte[ k ] = ∑n( thread\_entry\_point\_offset [ n-1] ) with n = 1..k+1

lastByte[ k ] = firstByte[ k ] – 1 + thread\_entry\_point\_offset [ k+1 ]

The last subset (with subset index equal to num\_thread\_entry\_point\_offsets\_minus1) consists of the remaining bytes of the access unit.