|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  9th Meeting: Geneva, Switzerland, 27 April – 07 May, 2012 | Document: JCTVC-I0118  M24357 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | AHG4: Enable parallel decoding with tiles | | |
| *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

Real-time UHD decoding can exceed the capability of a single core decoder. To enable parallel decoding on multi-core platforms, it is proposed to mandate evenly divided sub-pictures for high levels to guarantee pixel-rate balancing among cores when sub-pictures are processed in parallel. The key points of proposal are: 1) A picture is divided into a number of sub-pictures of equal size (in units of LCUs); 2) Sub-pictures are independent, only in-loop filters can be allowed cross the sub-picture boundaries; 3) Tiles, slices, entropy slices and WPP are contained in sub-pictures and cannot cross sub-picture boundaries; 4) The sub-picture partitioning information is signaled with tile syntax. If sub-pictures are mandated, tiles have to be uniformly spaced in vertical direction. 5) Sub-picture entries in bitstream are signaled in APS; 6) Sub-picture ID is signaled in slice header for low-latency applications. Finally, the limits for number of sub-pictures are also specified. The specification allows building a multi-core decoder by simply replicating the single core decoder without need of increasing the line buffer size.

# Introduction

Several parallel processing tools have been adopted into HEVC in addition to regular slices to support parallel processing for multi-core implementations. These tools include tiles, Wavefront Parallel Processing (WPP) and Entropy Slices (ES). So far the discussions have focused on the parallel processing on the encoder side, activities included the specification of rules that enables the usage of parallel tools on the encoder side without imposing significant burden on the single core decoder.

For UHD applications, multi-core devices are needed not only on the encoder side, but also on the decoder side. It is important that the HEVC has built-in features which enable the parallel processing capability on the decoder side as well.

For parallel processing on multiple core platforms, it is important that single core can be simply replicated to achieve high-performance for design simplification. Fig. 1 shows an example, in which a 4Kx2K@60 decoder can be built by simply replicating the 1080p@60 single core decoder four times, and adding a sub-picture boundary processing core to perform e.g. de-blocking filter, Sample Adaptive Offset (SAO), Adaptive Loop Filter (ALF) along sub-picture boundaries.

**Fig. 1 – From single core to multiple core**

To make such a decoder multi-core architecture feasible, a picture would need to be divided into sub-pictures of equal size, so that pixel-rate balancing can be guaranteed when individual cores are processing sub-pictures in parallel. However, based on the current specification, such a pixel balancing is not guaranteed (i.e. equal amount of LCUs dispatched to the cores) by parallel processing tools like tiles, WPP, entropy slice and slices.

Using tiles is a preferred solution to address the pixel rate balancing issue on multi-core platforms. Using the regular slices and WPP for the same purpose has the following drawbacks:

1. Regular slices: mandating regular slices of equal size (in terms of LCUs) could be a solution. However, there are use cases such as video conferencing in which a picture may create a large number of slices and each slice may contain a different number of LCUs. Therefore, it is hard to mandate slices of equal size in terms of LCUs. Also, slices are partitioned and coded in raster-scan order, which prevents from building a multi-core decoder by simply replicating single core decoders. For example, if a single core decoder is designed for e.g.1080p video, it would need to increase the single core line buffer size if cores are replicated to support e.g. 4Kx2K video.
2. WPP: for multi-core solutions, a shared MC cache is needed. Otherwise, it will incur huge memory bandwidth increase for motion compensation. Also, it needs frequent cross-core data communication for entropy decoding if sub-streams are processed on different cores in parallel. The line buffer of single core decoder needs to be increased too because WPP can only partitioned vertically. A shared MC cache and increased line buffer would require re-design of single core decoder. WPP is more appropriate for driving higher entropy decoding throughput by inserting parallel entropy decoding engines in a single core video decoder.

In the contribution, we propose to use tiles to enable such functionality. The design has considered the following requirements:

1. minimum changes to the current design.
2. minimum changes to the single core decoder architecture when cores are replicated to build up a multi-core decoder
3. and support of low-latency applications

# Algorithm description

To enable the multi-core architecture illustrated in Fig.1, constraints have to be imposed on the HEVC bitstreams. We propose to mandate evenly divided sub-pictures for high levels to support parallel decoding of UHD video.

## Sub-picture partitioning

If sub-pictures are mandated for a level, the sub-pictures share the following characteristics:

1. An incoming picture should be evenly divided into **num\_subPic\_columns x num\_subPic\_rows** sub-pictures so that pixel-rate balancing is guaranteed for multiple core platforms. A sub-picture contains an integer number of LCUs.
2. Sub-pictures should be independent, only de-blocking filter, Sample Adaptive Offset (SAO), Adaptive Loop Filter (ALF) can cross sub-picture boundary to minimize the data communication among the cores
3. Tiles, slices, entropy slices and WPP should be contained in sub-pictures and cannot cross sub-picture
4. The sub-picture partitioning information is signaled with tile syntax

## Signalling of the sub-picture partitioning with tile syntax

To signal the sub-picture partitioning using tile synatx, the following rules are applied:

1. Tiles have to be evenly divided (i.e. uniformly spaced) in vertical direction. If a picture is evenly divided into **num\_subPic\_columns** x **num\_subPic\_rows** sub-pictures, then num\_tile\_rows\_minus1 is constrained to be equal to num\_subPic\_rows -1. This constraint is important for guaranteeing pixel-rate balancing because tiles are processed in rater-scan order within a picture. Without this constraint, sub-picture data could be interleaved in the bitstream.
2. In horizontal direction tiles can either be evenly divided or non-uniformed spaced, **num\_tile\_columns\_minus1** can be larger or equal to **num\_subPic\_columns -1,** but it is constrained that there must be **num\_subPic\_columns -1** right-hand tile column boundaries are coincident with the vertical sub-picture boundaries. The horizontal location (in units of LCUs) of the i-th vertical sub-picture boundary is computed by
   * ( ( i + 1 )\*PicWidthInCtbs / num\_subPic\_columns, i = 0, 1, …, num\_subPic\_columns -1.
3. The sub-picture entries (start locations) in bitstream should be signaled in the APS to facilitate the parallel decoding.
4. For low-latency applications, a sub-picture ID (slice\_substream\_id) to which a slice belongs is signaled in the slice header.
5. Number of sub-picture **num\_subPic\_columns x num\_subPic\_rows** should be specified and mandated for levels based on the sample rates and picture resultions. If number of sub-picture **num\_subPic\_columns x num\_subPic\_rows** is set equal to 1 for a level, constraint 1 and 2, and signaling step 3 and 4 described above do not apply to that level.
6. Number of sub-picture **num\_subPic\_columns** x **num\_subPic\_rows** should be specified in a way in which line buffer size increase of single core decoder is not required when single cores are replicated to support higher resolution video.

The line buffer size used in a video core is proportional to the horizontal picture size. For example, if single core is designed for 1080p video, and it is replicated to process e.g. 4Kx2K video, then **num\_subPic\_columns** should be at least set to 2 for 4Kx2K level to avoid need of line buffer increase in the 1080p video core. This constraint simplifies the design of multi-core video codec by enabling simple replication of single cores to create multi-core platform.

Fig. 2 shows an example how to achieve an even sub-picture partitions of 2x2 sub-pictures by using the tile syntax. In this case, a picture is evenly divided into 2 tile rows in vertical direction, while in the horizontal direction it is unevenly divided into 4 columns, but there is one right-hand tile column boundary coincident with the sub-picture boundary, which is right-hand column boundary of Tile02 and Tile21. i.e. column\_width[0] + column\_width[1] has to be equal to PicWidthInCtbs/2 in this case.

In the bitstream, it should be signaled that Tile00 is start location for sub-picture 0, Tile10 is start location for sub-picture 1, Tile20 is start location for sub-picture 2 and Tile30 is start location for sub-picture 3. With this constrained tile partitions and signaling, sub-pictures can be dispatched and processed by multi-cores in parallel with equal pixel rate loading.

**Fig. 2 – Example of signaling the sub-picture boundaries for 2x2 sub-picture partitioning**

## Low-latency consideration

With the sub-picture entry signaling method, there is at least one picture decoding delay because the sub-picture entries can be sent only after the entire picture is encoded. This is unfriendly to low-latency applications. It is proposed to insert a sub-picture ID in the slice header to resolve the latency issue.

**Fig. 3 diagram of multi-core HEVC low-latency coding with signaling of sub-picture ID in slice header**

With the signaling of sub-picture ID “slice\_subtream\_id” in the slice header, the low latency HEVC video encoding and decoding can be realized on multi-core platforms. As shown in Fig. 3 for example, an input picture is divided into four sub-pictures of equal size, which are encoded by four encoder video cores in parallel. Each encoder video core inserts a “slice\_substreams\_id” into the slice header of the slices generated for the sub-pictures. On the encoder side, the sender transmits the slices to the receiver as long as a slice of a sub-picture becomes available. On the decoder side, the receiver receives bitstream and decodes “slice\_substreams\_id” from the slice header, then dispatches the slices to corresponding decoder cores based on the value of “slice\_substreams\_id”. In this way, the sub-pictures can be decoded in parallel by four video decoder cores.

While slices are still encoded, transmitted and decoded in raster scanning order within a sub-picture (i.e. data processing order is from left to right and from top to bottom in a sub-picture), the slices from different sub-pictures are transmitted in “interleaved” order to ensure low-latency, that is, a slice from a sub-picture is transmitted as long as it becomes available to the sender. This so-called interleaved transmission order is different from the traditional raster scan transmission order in which all the slices from sub-picture 0 are transmitted first, then all the slices from sib-picture 1 are transmitted, and so on so forth until all the slices from last sub-picture (it is sub-picture 3 in Fig. 3) are transmitted. On the decoder side, each video decoder cores still receives and decodes slices in raster scan order for each sub-picture, but the receiver receives slices for different sub-pictures in “interleaved” order.

## High-level syntax changes

The tile syntax in SPS (sequence parameter set) and PPS (picture parameter set) is modified as shown in Table 1. In the vertical direction the non-uniformed spaced tiles are allowed only if sub-pictures are not mandated.

|  |  |
| --- | --- |
| … |  |
| 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) |
| If (num\_subpic\_columns == 1 && num\_subpic\_rows ==1) |  |
| for( i = 0; i < num\_tile\_rows\_minus1; i++ ) |  |
| **row\_height[**i **]** | ue(v) |
| } |  |
| **loop\_filter\_across\_tiles\_enabled\_flag** | u(1) |
| } |  |
| … |  |

**Table 1 Modified Tile syntax for the SPS and PPS**

**I**n the current HEVC design the tile entries in the bitstream are signaled in slice header. With the information of tile entry, a decoder can eventually decode multiple tiles in parallel by start decoding at multiple tile entries in the bitstream. However, sub-picture entry information is picture-level information and must be signaled before decoding any slice, it is inadequate to carry the tile-like entry information in the slice header for parallel decoding purpose. Therefore, one solution of signaling the sub-picture entries in the bitstream is to put the similar slice-level tile entry information into the adaptation parameter set (APS) for sub-pictures as shown in Table 2 (marked in yellow).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| aps\_rbsp( ) { | | Descriptor | | |
| **aps\_id** | | ue(v) | | |
| **… …** | |  | | |
| **aps\_sub\_pic\_entry\_present\_flag** | | u(1) | | |
| if (aps\_sub\_pic\_entry\_present\_flag) { | |  | | |
| **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) |
|  | | |  | |
| } | | |  | |
| } | |  | | |
| **… …** | |  | | |

**Table 2 Modified APS syntax for the sub-picture entry signaling**

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| If ( tiles\_or\_entropy\_coding\_sync\_idc == 1 && (num\_subpic\_columns> 1 || num\_subpic\_rows > 1) ) { |  |
| **slice\_substream\_id** | ue(v) |
| } |  |
| **first\_slice\_in\_pic\_flag** | u(1) |
| if( first\_slice\_in\_pic\_flag = = 0 ) |  |
| **slice\_address** | u(v) |
| **slice\_type** | ue(v) |
| …. |  |
| if( seq\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_adaptive\_loop\_filter\_flag | | slice\_sample\_adaptive\_offset\_flag | |  !disable\_deblocking\_filter\_flag ) ) |  |
| slice\_loop\_filter\_across\_slices\_enabled\_flag | u(1) |
| if( tiles\_or\_entropy\_coding\_sync\_idc == 2 || (aps\_sub\_pic\_entry\_present\_flag==0 && tiles\_or\_entropy\_coding\_sync\_idc == 1)) { |  |
| 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) |
| } |  |
| } |  |
| } |  |

**Table 3 Modified slice header syntax for sub-picture ID signaling and tile entry signaling**

For low-latency applications, a slice\_substream\_id is inserted for each slice in the slice\_header as shown in Table 3. The slice\_substream\_id is in the range of 0 to num\_subpic\_columns\*num\_subpic\_rows -1, inclusive. If a slice belongs to sub-picture k (k= 0, 1, …, num\_subpic\_columns\*num\_subpic\_rows -1) in the picture, slice\_substream\_id is set equal to k. slice\_substream\_id is the **first** syntax element in the slice\_header() so that the decoder can easily decode this element without need of parsing through bunch of slice header data, and dispatch the slice to corresponding decoder cores based on the value of slice\_substream\_id.

## Specification for number of sub-pictures

The number of sub-pictures, num\_subpic\_columns \* num\_subpic\_row, and minimum value of sub-picture columns (i.e. **min\_num\_subpic\_columns**) mandated for each level are specified in Table 4 & 5. The actual number of sub-picture column **num\_subpic\_columns** should be larger than or equal to **min\_num\_subpic\_columns**. Table 4 assumes 1080p@60 single core decoder capability, and Table 5 assumes 4kx2k@30 single core decoder capability. The committee should decide on the minimum picture size and sample rate for which the sub-pictures should be mandated.

The specification of num\_subpic\_columns and num\_subpic\_rows in Table 4 and Table 5 enables building the multi-core real-time encoder/decoder by simply replicating single-core codecs without the need of increasing single core line buffer size to support higher-resolution video encoding/decoding. For example in Table 4 Level limits (assuming 1080p@60 single core decoder capability), level 5, 5.1 and 5.2 support 4Kx2K video, min\_num\_subpic\_columns is set equal to 2, meaning that a 4Kx2K picture has to be evenly divided at least by a factor 2 in horizontal direction for sub-pictures, this guarantees that a 1080p@60 single core which has horizontal line buffer size of 2K can encode/decode sub-pictures in real-time without need of enlarging its line buffer size.

This specification of sub-picture numbers provides more flexibility. For example, if the total number of sub-pictures is defined as 4 and min\_num\_subpic\_columns is defined as 2, a picture can either divided into 2x2 or 4x1 sub-pictures, in which 4x1 partitioning (i.e. 4 sub-picture columns and 1 sub-picture row) offers the lowest latency for coding on multiple cores.

Note that In Table 4 and Table 5, both number of sub-pictures (i.e. num\_subpic\_columns x num\_subpic\_rows) and min\_num\_subpic\_columns are constrained to be power of 2 to facilitate the pixel rate balancing on multi-core platforms. The actual number of sub-picture columns (num\_subpic\_columns) is also constrained to be power of 2.

Because num\_subpic\_rows is equal to num\_tile\_rows\_minus1 +1 when sub-pictures are present, num\_subpic\_colmns does not need to be explicitly signaled in the bitstream. Instead, it can be derived from the level ID and num\_tile\_rows\_minus1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Level** | **Max luma pixel rate MaxLumaPR**  **(samples/sec)** | **Max luma picture size MaxLumaFS (samples)** | **Max bit rate MaxBR**  **(1000 bits/s)** | **Min Compression Ratio MinCR** | **MaxDpbSize (picture storage buffers)** | **Max CPB size**  **(1000 bits)** | **Number of sub-pictures**  **(num\_subpic\_columns x num\_subpic\_rows)** | **min\_num\_subpic\_columns** |
| **1** | 552,960 | 36,864 | 128 | 2 | 6 | 350 | 1 | 1 |
| **2** | 3,686,400 | 122,880 | 1,000 | 2 | 6 | 1,000 | 1 | 1 |
| **3** | 13,762,560 | 458,752 | 5,000 | 2 | 6 | 5,000 | 1 | 1 |
| **3.1** | 33,177,600 | 983,040 | 9,000 | 2 | 6 | 9,000 | 1 | 1 |
| **4** | 62,668,800 | 2,088,960 | 15,000 | 4 | 6 | 15,000 | 1 | 1 |
| **4.1** | 62,668,800 | 2,088,960 | 30,000 | 4 | 6 | 30,000 | 1 | 1 |
| **4.2** | 133,693,440 | 2,228,224 | 30,000 | 4 | 6 | 30,000 | 1 | 1 |
| **4.3** | 133,693,440 | 2,228,224 | 50,000 | 4 | 6 | 50,000 | 1 | 1 |
| **5** | 267,386,880 | 8,912,896 | 50,000 | 6 | 6 | 50,000 | 2 | 2 |
| **5.1** | 267,386,880 | 8,912,896 | 100,000 | 8 | 6 | 100,000 | 2 | 2 |
| **5.2** | 534,773,760 | 8,912,896 | 150,000 | 8 | 6 | 150,000 | 4 | 2 |
| **6** | 1,002,700,800 | 33,423,360 | 300,000 | 8 | 6 | 300,000 | 8 | 4 |
| **6.1** | 2,005,401,600 | 33,423,360 | 500,000 | 8 | 6 | 500,000 | 16 | 4 |
| **6.2** | 4,010,803,200 | 33,423,360 | 800,000 | 6 | 6 | 800,000 | 32 | 4 |

Table 4 – Level limits (assuming 1080p@60 single core decoder capability)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Level*** | **Max luma pixel rate MaxLumaPR**  **(samples/sec)** | **Max luma picture size MaxLumaFS (samples)** | **Max bit rate MaxBR**  **(1000 bits/s)** | **Min Compression Ratio MinCR** | **MaxDpbSize (picture storage buffers)** | **Max CPB size**  **(1000 bits)** | **Number of sub-pictures**  **(num\_subpic\_columns x num\_subpic\_rows)** | **min\_num\_subpic\_columns** |
| ***1*** | 552,960 | 36,864 | 128 | 2 | 6 | 350 | 1 | 1 |
| ***2*** | 3,686,400 | 122,880 | 1,000 | 2 | 6 | 1,000 | 1 | 1 |
| ***3*** | 13,762,560 | 458,752 | 5,000 | 2 | 6 | 5,000 | 1 | 1 |
| ***3.1*** | 33,177,600 | 983,040 | 9,000 | 2 | 6 | 9,000 | 1 | 1 |
| ***4*** | 62,668,800 | 2,088,960 | 15,000 | 4 | 6 | 15,000 | 1 | 1 |
| ***4.1*** | 62,668,800 | 2,088,960 | 30,000 | 4 | 6 | 30,000 | 1 | 1 |
| ***4.2*** | 133,693,440 | 2,228,224 | 30,000 | 4 | 6 | 30,000 | 1 | 1 |
| ***4.3*** | 133,693,440 | 2,228,224 | 50,000 | 4 | 6 | 50,000 | 1 | 1 |
| ***5*** | 267,386,880 | 8,912,896 | 50,000 | 6 | 6 | 50,000 | 1 | 1 |
| ***5.1*** | 267,386,880 | 8,912,896 | 100,000 | 8 | 6 | 100,000 | 1 | 1 |
| ***5.2*** | 534,773,760 | 8,912,896 | 150,000 | 8 | 6 | 150,000 | 2 | 1 |
| ***6*** | 1,002,700,800 | 33,423,360 | 300,000 | 8 | 6 | 300,000 | 4 | 2 |
| ***6.1*** | 2,005,401,600 | 33,423,360 | 500,000 | 8 | 6 | 500,000 | 8 | 2 |
| ***6.2*** | 4,010,803,200 | 33,423,360 | 800,000 | 6 | 6 | 800,000 | 16 | 2 |

Table 5 – Level limits (assuming 4kx2k@30 single core decoder capability)

# Conclusion and recommendation

It is recommended to mandate tiles for high levels to enable parallel decoding on multi-core platforms.

# References

[1] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC Document, JCTVC-G1100, San Jose, CA, USA, February 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 6 (HM 6)” JCT-VC Document, JCTVC-G1003, San Jose, CA, USA, February 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).**