

A DECODER AND CORRESPONDING METHODS TO SIGNAL PICTURE
PARTITIONING INFORMATION FOR SLICES

5 CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority from PCT/EP2020/055220 in the World Intellectual Property Organization. The disclosures of the aforementioned patent applications are hereby incorporated by reference in their entirety.

10 TECHNICAL FIELD

Embodiments of the present application generally relate to the field of picture processing and more particularly to signal picture partitioning information for slices.

BACKGROUND

- 15 Video coding (video encoding and decoding) is used in a wide range of digital video applications, for example broadcast digital TV, video transmission over internet and mobile networks, real-time conversational applications such as video chat, video conferencing, DVD and Blu-ray discs, video content acquisition and editing systems, and camcorders of security applications.
- 20 The amount of video data needed to depict even a relatively short video can be substantial, which may result in difficulties when the data is to be streamed or otherwise communicated across a communications network with limited bandwidth capacity. Thus, video data is generally compressed before being communicated across modern day telecommunications networks. The size of a video could also be an issue when the video is stored on a storage
- 25 device because memory resources may be limited. Video compression devices often use software and/or hardware at the source to code the video data prior to transmission or storage,

thereby decreasing the quantity of data needed to represent digital video images. The compressed data is then received at the destination by a video decompression device that decodes the video data. With limited network resources and ever increasing demands of higher video quality, improved compression and decompression techniques that improve
5 compression ratio with little to no sacrifice in picture quality are desirable.

SUMMARY

Embodiments of the present application provide apparatuses and methods for encoding and decoding according to the independent claims.

10 The foregoing and other objects are achieved by the subject matter of the independent claims. Further implementation forms are apparent from the dependent claims, the description and the figures.

The first aspect of the present invention provides a method of coding implemented by a decoding device, the method comprising: obtaining a bitstream for a current picture;
15 obtaining a quantity of tiles in a tile row of the current picture; obtaining a value of an address of a slice of the current picture; obtaining a value of tileX according to the value of the address of the slice; parsing an indication value for a slice width from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold.

20 According to embodiments of the present invention, a signaling scheme is disclosed, only there are at least two tile columns, then syntax element for slice width is signaled. Hence, the bitstream utilization and decoding efficiency have been improved.

The first threshold is an integer value, the first threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile row is same means as a number of
25 tile columns.

In one implementation, wherein the method further comprises: setting the indication value for the slice width to a default value, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is equal to the first threshold. The default value is an integer value, in an example, the default value may be 0, or 1.

- 5 In one implementation, wherein the default value indicates that the width of the slice comprises 1 tile column.

In one implementation, wherein the obtaining a value of tileX according to the value of the address of the slice comprises: obtaining the value of the tileX according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

- 10 In one implementation, wherein the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

- In one implementation, wherein the value of tileX is obtained according to a modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a first threshold”. Hence, the value of tileX is not calculated at first but is represented according to the result of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

- 20 In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.

- 25 In one implementation, wherein the value of tileX indicates a value of a horizontal coordinate. In an example, the value of tileX indicates a horizontal address in tile index.

In one implementation, wherein the method further comprises: obtaining a quantity of tiles in a tile column of the current picture; obtaining a value of tileY according to the value of the address of the slice; parsing an indication value for a slice height from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile column of the current picture and the value of the tileY is not equal to a second threshold. The second threshold is an integer value, the second threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile column is same means as a number of tile rows.

In one implementation, wherein the method further comprises: setting the indication value for the slice height to a preset value, when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is equal to the second threshold.

In one implementation, wherein the obtaining a value of tileY according to the value of the address of the slice comprises: obtaining the value of tileY according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of tileY is obtained according to an integer division operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. In one implementation, wherein the value of tileY indicates a value of a vertical coordinate. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is not equal to a second threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of the integer division operation between the value of the address of the slice and the quantity of tiles in the tile column of the current picture is not equal to a second threshold”. Hence, the value of tileY is not calculated at first but is represented according to the result of the integer division operation between the value of the address of the slice and the quantity of tiles in the

tile row of the current picture. In an example, the value of tileY indicates a vertical address in tile index.

The second aspect of the present invention provides a method of coding implemented by a decoding device, the method comprising: obtaining a bitstream for a current picture;
5 obtaining a quantity of tiles in a tile column of the current picture; obtaining a value of an address of a slice of the current picture; obtaining a value of tileY according to the value of the address of the slice; parsing an indication value for a slice height from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile column of the
10 current picture and the value of tileY is not equal to a second threshold.

The second threshold is an integer value, the second threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile column is same means as a number of tile rows.

In one implementation, wherein the method further comprises: setting the indication value for
15 the slice height to a preset value, when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is equal to the second threshold. The default value is an integer value, in an example, the default value may be 0, or 1.

In one implementation, wherein the default value indicates that the height of the slice comprises 1 tile row.

20 In one implementation, wherein the obtaining a value of tileY according to the value of the address of the slice comprises:

obtaining a quantity of tiles in a tile row of the current picture;

obtaining the value of tileY according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

25 In one implementation, wherein the value of tileY indicates a value of a vertical coordinate.

In an example, the value of tileY indicates a vertical address in tile index.

In one implementation, wherein the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

In one implementation, wherein the value of tileY is obtained according to an integer division operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.

In one implementation, wherein the method further comprises: obtaining a quantity of tiles in a tile row of the current picture; obtaining a value of tileX according to the value of the address of the slice; parsing an indication value for a slice width from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile row of the current picture and the value of the tileX is not equal to a first threshold. The first threshold is an integer value, the first threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile row is same means as a number of tile columns.

In one implementation, wherein the method further comprises: setting the indication value for the slice width to a default value, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is equal to the first threshold. The default value is an integer value, in an example, the default value may be 0, or 1.

In one implementation, wherein the default value indicates that the width of the slice comprises 1 tile column.

In one implementation, wherein the obtaining a value of tileX according to the value of the address of the slice comprises: obtaining the value of the tileX according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

In one implementation, wherein the value of tileX is obtained according to a modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of
5 modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a first threshold”. Hence, the value of tileX is not calculated at first but is represented according to the result of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.
10

In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.
15

In one implementation, wherein the value of tileX indicates a value of a horizontal coordinate.

The third aspect of the present invention provides a method of coding implemented by an encoding device, the method comprising: obtaining a quantity of tiles in a tile row of a current picture; obtaining a value of an address of a slice of the current picture; obtaining a value of tileX according to the value of the address of the slice; coding an indication value for a slice width into a bitstream for the current picture, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a
20 first threshold.
25

According to embodiments of the present invention, a signaling scheme is disclosed, only there are at least two tile columns, then syntax element for slice width is signaled. Hence, the bitstream utilization and decoding efficiency have been improved.

The first threshold is an integer value, the first threshold may be 0, 1, 2...

- 5 It could be understood that, the quantity of tiles in a tile row is same means as a number of tile columns.

In one implementation, wherein the obtaining a value of tileX according to the value of the address of the slice comprises: obtaining the value of the tileX according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

- 10 In one implementation, wherein the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

- In one implementation, wherein the value of tileX is obtained according to a modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a first threshold”. Hence, the value of tileX is not calculated at first but is represented according to the result of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

- 20 In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.
- 25 In one implementation, wherein the value of tileX indicates a value of a horizontal coordinate.

In one implementation, wherein the method further comprises: obtaining a quantity of tiles in a tile column of the current picture; obtaining a value of tileY according to the value of the address of the slice; coding an indication value for a slice height into the bitstream for the current picture, when a difference value between the quantity of tiles in the tile column of the current picture and the value of the tileY is not equal to a second threshold. The second threshold is an integer value, the second threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile column is same means as a number of tile rows.

In one implementation, wherein the obtaining a value of tileY according to the value of the address of the slice comprises: obtaining the value of tileY according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of tileY is obtained according to an integer division operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. In one implementation, wherein the value of tileY indicates a value of a vertical coordinate. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is not equal to a second threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of the integer division operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a second threshold”. Hence, the value of tileY is not calculated at first but is represented according to the result of the integer division operation between the value of the address of the slice and the quantity of tiles in the tile column of the current picture.

The fourth aspect of the present invention provides a method of coding implemented by an encoding device, the method comprising: obtaining a quantity of tiles in a tile column of the

current picture; obtaining a value of an address of a slice of the current picture; obtaining a value of tileY according to the value of the address of the slice; coding an indication value for a slice height into a bitstream for the current picture, when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is not equal to
5 a second threshold.

The second threshold is an integer value, the second threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile column is same means as a number of tile rows.

In one implementation, wherein the obtaining a value of tileY according to the value of the
10 address of the slice comprises:

obtaining the value of tileY according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of tileY indicates a value of a vertical coordinate.

In one implementation, wherein the address of the slice is an index of leftmost tile of the
15 topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

In one implementation, wherein the value of tileY is obtained according to an integer division operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

20 In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.

In one implementation, wherein the method further comprises: obtaining a quantity of tiles in a tile row of the current picture; obtaining a value of tileX according to the value of the address of the slice; coding an indication value for a slice width into the bitstream for the
25 current picture, when a difference value between the quantity of tiles in the tile row of the

current picture and the value of the tileX is not equal to a first threshold. The first threshold is an integer value, the first threshold may be 0, 1, 2...

It could be understood that, the quantity of tiles in a tile row is same means as a number of tile columns.

- 5 In one implementation, wherein the obtaining a value of tileX according to the value of the address of the slice comprises: obtaining the value of the tileX according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed
10 according to a raster scan order.

In one implementation, wherein the value of tileX is obtained according to a modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture. It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the
15 value of tileX is not equal to a first threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a first threshold”. Hence, the value of tileX is not calculated at first but is represented according to the result of modulus operation between the
20 value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.

In one implementation, wherein the value of tileX indicates a value of a horizontal coordinate.

25

The fifth aspect of the preset invention provides a video decoding apparatus, the apparatus comprise:

obtaining module, which is configured to obtain a bitstream for a current picture; and
obtaining module is configured to obtain a quantity of tiles in a tile row of the current picture

5 and obtain a value of an address of a slice of the current picture;

calculating module, which is configured to obtain a value of tileX according to the value of
the address of the slice;

parsing module, which is configured to parse an indication value for a slice width from the
bitstream for the current picture, when a difference value between the quantity of tiles in the
10 tile row of the current picture and the value of tileX is not equal to a first threshold.

The details for each module in the fifth aspect of the preset invention could refer to the
embodiments and implementations in the above first aspect of the preset invention.

The sixth aspect of the present invention provides a video decoding device, the decoding
15 device comprise:

obtaining module, which is configured to obtain a bitstream for a current picture; obtaining
module is configured to obtain a quantity of tiles in a tile column of the current picture, and
obtain a value of an address of a slice of the current picture;

calculating module, which is configured to obtain a value of tileY according to the value of
20 the address of the slice;

parsing module, which is configured to parse an indication value for a slice height from the
bitstream for the current picture, when a difference value between the quantity of tiles in the
tile column of the current picture and the value of tileY is not equal to a second threshold.

The details for each module in the sixth aspect of the preset invention could refer to the
25 embodiments and implementations in the above second aspect of the preset invention.

The seventh aspect of the present invention provides a video encoding device, the encoding device comprise:

obtaining module, which is configured to obtain a quantity of tiles in a tile row of a current picture; obtaining module is configured to obtaining a value of an address of a slice of the
5 current picture;

calculating module, which is configured to obtain a value of tileX according to the value of the address of the slice;

coding module, which is configured to code an indication value for a slice width into a bitsteam for the current picture, when a difference value between the quantity of tiles in the
10 tile row of the current picture and the value of tileX is not equal to a first threshold.

The details for each module in the seventh aspect of the preset invention could refer to the embodiments and implementations in the above third aspect of the preset invention.

The eighth aspect of the present invention provides a video encoding device, the video
15 encoding device comprise: obtaining module, which is configured to obtain a quantity of tiles in a tile column of the current picture; obtaining module is configured to obtain a value of an address of a slice of the current picture;

calculating module, which is configured to obtain a value of tileY according to the value of the address of the slice;

20 coding module, which is configured to code an indication value for a slice height into a bitsteam for the current picture, when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is not equal to a second threshold.

The details for each module in the eighth aspect of the preset invention could refer to the embodiments and implementations in the above fourth aspect of the preset invention.

25

The ninth aspect of the present invention provides a decoder comprising processing circuitry for carrying out the method according to the first aspect, the second aspect and any one of implementation of the first aspect, the second aspect.

- 5 The tenth aspect of the present invention provides a decoder comprising processing circuitry for carrying out the method according to the third aspect, the fourth aspect and any one of implementation of the third aspect, the fourth aspect.

- The tenth aspect of the present invention provides a computer program product comprising
10 program code for performing the method of the first aspect, the second aspect, the third aspect, the fourth aspect and any one of implementation of the first aspect, the second aspect, the third aspect and the fourth aspect.

- The eleventh aspect of the present invention provides a non-transitory computer-readable
15 medium carrying a program code which, when executed by a computer device, causes the computer device to perform the method of the first aspect, the second aspect, the third aspect, the fourth aspect and any one of implementation of the first aspect, the second aspect, the third aspect and the fourth aspect.

- 20 The twelfth aspect of the present invention provides a decoder, comprising:
one or more processors; and
a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the decoder to carry out the method according to the first aspect,
25 the second aspect and any one of implementation of the first aspect, the second aspect.

The thirteenth aspect of the present invention provides an encoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by

5 the processors, configures the decoder to carry out the method according to the third aspect, the fourth aspect and any one of implementation of the third aspect, the fourth aspect.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following embodiments of the invention are described in more detail with reference to

10 the attached figures and drawings, in which:

FIG. 1A is a block diagram showing an example of a video coding system configured to implement embodiments of the invention;

FIG. 1B is a block diagram showing another example of a video coding system configured to implement embodiments of the invention;

15 FIG. 2 is a block diagram showing an example of a video encoder configured to implement embodiments of the invention;

FIG. 3 is a block diagram showing an example structure of a video decoder configured to implement embodiments of the invention;

FIG. 4 is a block diagram illustrating an example of an encoding apparatus or a decoding apparatus;

20 FIG. 5 is a block diagram illustrating another example of an encoding apparatus or a decoding apparatus;

FIG. 6 is an example of a picture divided into CTUs;

FIG. 7 is an example of Tile-based raster scan order of CTUs;

25 FIG. 8 is an example of picture partitioning;

FIG. 9 is an example of CTU address information in a picture;

FIG. 10 is another example of a picture partitioning;

FIG. 11 is a block diagram showing an example structure of a content supply system 3100 which realizes a content delivery service;

FIG. 12 is a block diagram showing a structure of an example of a terminal device.

5 FIG. 13 is a flowchart showing a method embodiment.

FIG. 14 is a block diagram showing an apparatus embodiment.

In the following identical reference signs refer to identical or at least functionally equivalent features if not explicitly specified otherwise.

10 DETAILED DESCRIPTION OF THE EMBODIMENTS

In the following description, reference is made to the accompanying figures, which form part of the disclosure, and which show, by way of illustration, specific aspects of embodiments of the invention or specific aspects in which embodiments of the present invention may be used.

It is understood that embodiments of the invention may be used in other aspects and comprise

15 structural or logical changes not depicted in the figures. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

For instance, it is understood that a disclosure in connection with a described method may also hold true for a corresponding device or system configured to perform the method and

20 vice versa. For example, if one or a plurality of specific method steps are described, a corresponding device may include one or a plurality of units, e.g. functional units, to perform the described one or plurality of method steps (e.g. one unit performing the one or plurality of steps, or a plurality of units each performing one or more of the plurality of steps), even if such one or more units are not explicitly described or illustrated in the figures. On the other
25 hand, for example, if a specific apparatus is described based on one or a plurality of units, e.g. functional units, a corresponding method may include one step to perform the functionality of

the one or plurality of units (e.g. one step performing the functionality of the one or plurality of units, or a plurality of steps each performing the functionality of one or more of the plurality of units), even if such one or plurality of steps are not explicitly described or illustrated in the figures. Further, it is understood that the features of the various exemplary
5 embodiments and/or aspects described herein may be combined with each other, unless specifically noted otherwise.

Video coding typically refers to the processing of a sequence of pictures, which form the video or video sequence. Instead of the term “picture” the term “frame” or “image” may be used as synonyms in the field of video coding. Video coding (or coding in general) comprises
10 two parts video encoding and video decoding. Video encoding is performed at the source side, typically comprising processing (e.g. by compression) the original video pictures to reduce the amount of data required for representing the video pictures (for more efficient storage and/or transmission). Video decoding is performed at the destination side and typically comprises the inverse processing compared to the encoder to reconstruct the video pictures.
15 Embodiments referring to “coding” of video pictures (or pictures in general) shall be understood to relate to “encoding” or “decoding” of video pictures or respective video sequences. The combination of the encoding part and the decoding part is also referred to as CODEC (Coding and Decoding).

In case of lossless video coding, the original video pictures can be reconstructed, i.e. the
20 reconstructed video pictures have the same quality as the original video pictures (assuming no transmission loss or other data loss during storage or transmission). In case of lossy video coding, further compression, e.g. by quantization, is performed, to reduce the amount of data representing the video pictures, which cannot be completely reconstructed at the decoder, i.e. the quality of the reconstructed video pictures is lower or worse compared to the quality of
25 the original video pictures.

Several video coding standards belong to the group of “lossy hybrid video codecs” (i.e. combine spatial and temporal prediction in the sample domain and 2D transform coding for applying quantization in the transform domain). Each picture of a video sequence is typically partitioned into a set of non-overlapping blocks and the coding is typically performed on a block level. In other words, at the encoder the video is typically processed, i.e. encoded, on a block (video block) level, e.g. by using spatial (intra picture) prediction and/or temporal (inter picture) prediction to generate a prediction block, subtracting the prediction block from the current block (block currently processed/to be processed) to obtain a residual block, transforming the residual block and quantizing the residual block in the transform domain to reduce the amount of data to be transmitted (compression), whereas at the decoder the inverse processing compared to the encoder is applied to the encoded or compressed block to reconstruct the current block for representation. Furthermore, the encoder duplicates the decoder processing loop such that both will generate identical predictions (e.g. intra- and inter predictions) and/or re-constructions for processing, i.e. coding, the subsequent blocks.

In the following embodiments of a video coding system 10, a video encoder 20 and a video decoder 30 are described based on Figs. 1 to 3.

Fig. 1A is a schematic block diagram illustrating an example coding system 10, e.g. a video coding system 10 (or short coding system 10) that may utilize techniques of this present application. Video encoder 20 (or short encoder 20) and video decoder 30 (or short decoder 30) of video coding system 10 represent examples of devices that may be configured to perform techniques in accordance with various examples described in the present application. As shown in FIG. 1A, the coding system 10 comprises a source device 12 configured to provide encoded picture data 21 e.g. to a destination device 14 for decoding the encoded picture data 13.

The source device 12 comprises an encoder 20, and may additionally, i.e. optionally, comprise a picture source 16, a pre-processor (or pre-processing unit) 18, e.g. a picture pre-processor 18, and a communication interface or communication unit 22.

The picture source 16 may comprise or be any kind of picture capturing device, for example a camera for capturing a real-world picture, and/or any kind of a picture generating device, for example a computer-graphics processor for generating a computer animated picture, or any kind of other device for obtaining and/or providing a real-world picture, a computer generated picture (e.g. a screen content, a virtual reality (VR) picture) and/or any combination thereof (e.g. an augmented reality (AR) picture). The picture source may be any kind of memory or storage storing any of the aforementioned pictures.

In distinction to the pre-processor 18 and the processing performed by the pre-processing unit 18, the picture or picture data 17 may also be referred to as raw picture or raw picture data 17.

Pre-processor 18 is configured to receive the (raw) picture data 17 and to perform pre-processing on the picture data 17 to obtain a pre-processed picture 19 or pre-processed picture data 19. Pre-processing performed by the pre-processor 18 may, e.g., comprise trimming, color format conversion (e.g. from RGB to YCbCr), color correction, or de-noising. It can be understood that the pre-processing unit 18 may be optional component.

The video encoder 20 is configured to receive the pre-processed picture data 19 and provide encoded picture data 21 (further details will be described below, e.g., based on Fig. 2).

Communication interface 22 of the source device 12 may be configured to receive the encoded picture data 21 and to transmit the encoded picture data 21 (or any further processed version thereof) over communication channel 13 to another device, e.g. the destination device 14 or any other device, for storage or direct reconstruction.

The destination device 14 comprises a decoder 30 (e.g. a video decoder 30), and may additionally, i.e. optionally, comprise a communication interface or communication unit 28, a post-processor 32 (or post-processing unit 32) and a display device 34.

The communication interface 28 of the destination device 14 is configured receive the
5 encoded picture data 21 (or any further processed version thereof), e.g. directly from the source device 12 or from any other source, e.g. a storage device, e.g. an encoded picture data storage device, and provide the encoded picture data 21 to the decoder 30.

The communication interface 22 and the communication interface 28 may be configured to transmit or receive the encoded picture data 21 or encoded data 13 via a direct
10 communication link between the source device 12 and the destination device 14, e.g. a direct wired or wireless connection, or via any kind of network, e.g. a wired or wireless network or any combination thereof, or any kind of private and public network, or any kind of combination thereof.

The communication interface 22 may be, e.g., configured to package the encoded picture data
15 21 into an appropriate format, e.g. packets, and/or process the encoded picture data using any kind of transmission encoding or processing for transmission over a communication link or communication network.

The communication interface 28, forming the counterpart of the communication interface 22, may be, e.g., configured to receive the transmitted data and process the transmission data
20 using any kind of corresponding transmission decoding or processing and/or de-packaging to obtain the encoded picture data 21.

Both, communication interface 22 and communication interface 28 may be configured as unidirectional communication interfaces as indicated by the arrow for the communication channel 13 in Fig. 1A pointing from the source device 12 to the destination device 14, or
25 bi-directional communication interfaces, and may be configured, e.g. to send and receive messages, e.g. to set up a connection, to acknowledge and exchange any other information

related to the communication link and/or data transmission, e.g. encoded picture data transmission.

The decoder 30 is configured to receive the encoded picture data 21 and provide decoded picture data 31 or a decoded picture 31 (further details will be described below, e.g., based on Fig. 3 or Fig. 5).

The post-processor 32 of destination device 14 is configured to post-process the decoded picture data 31 (also called reconstructed picture data), e.g. the decoded picture 31, to obtain post-processed picture data 33, e.g. a post-processed picture 33. The post-processing performed by the post-processing unit 32 may comprise, e.g. color format conversion (e.g. from YCbCr to RGB), color correction, trimming, or re-sampling, or any other processing, e.g. for preparing the decoded picture data 31 for display, e.g. by display device 34.

The display device 34 of the destination device 14 is configured to receive the post-processed picture data 33 for displaying the picture, e.g. to a user or viewer. The display device 34 may be or comprise any kind of display for representing the reconstructed picture, e.g. an integrated or external display or monitor. The displays may, e.g. comprise liquid crystal displays (LCD), organic light emitting diodes (OLED) displays, plasma displays, projectors, micro LED displays, liquid crystal on silicon (LCoS), digital light processor (DLP) or any kind of other display.

Although Fig. 1A depicts the source device 12 and the destination device 14 as separate devices, embodiments of devices may also comprise both or both functionalities, the source device 12 or corresponding functionality and the destination device 14 or corresponding functionality. In such embodiments the source device 12 or corresponding functionality and the destination device 14 or corresponding functionality may be implemented using the same hardware and/or software or by separate hardware and/or software or any combination thereof.

As will be apparent for the skilled person based on the description, the existence and (exact) split of functionalities of the different units or functionalities within the source device 12 and/or destination device 14 as shown in Fig. 1A may vary depending on the actual device and application.

5 The encoder 20 (e.g. a video encoder 20) or the decoder 30 (e.g. a video decoder 30) or both encoder 20 and decoder 30 may be implemented via processing circuitry as shown in Fig. 1B, such as one or more microprocessors, digital signal processors (DSPs), application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), discrete logic, hardware, video coding dedicated or any combinations thereof. The encoder 20 may be
10 implemented via processing circuitry 46 to embody the various modules as discussed with respect to encoder 20 of FIG. 2 and/or any other encoder system or subsystem described herein. The decoder 30 may be implemented via processing circuitry 46 to embody the various modules as discussed with respect to decoder 30 of FIG. 3 and/or any other decoder system or subsystem described herein. The processing circuitry may be configured to perform
15 the various operations as discussed later. As shown in fig. 5, if the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable storage medium and may execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Either of video encoder 20 and video decoder 30 may be integrated as part of a combined
20 encoder/decoder (CODEC) in a single device, for example, as shown in Fig. 1B.

Source device 12 and destination device 14 may comprise any of a wide range of devices, including any kind of handheld or stationary devices, e.g. notebook or laptop computers, mobile phones, smart phones, tablets or tablet computers, cameras, desktop computers, set-top boxes, televisions, display devices, digital media players, video gaming consoles,
25 video streaming devices (such as content services servers or content delivery servers), broadcast receiver device, broadcast transmitter device, or the like and may use no or any

kind of operating system. In some cases, the source device 12 and the destination device 14 may be equipped for wireless communication. Thus, the source device 12 and the destination device 14 may be wireless communication devices.

In some cases, video coding system 10 illustrated in Fig. 1A is merely an example and the techniques of the present application may apply to video coding settings (e.g., video encoding or video decoding) that do not necessarily include any data communication between the encoding and decoding devices. In other examples, data is retrieved from a local memory, streamed over a network, or the like. A video encoding device may encode and store data to memory, and/or a video decoding device may retrieve and decode data from memory. In some examples, the encoding and decoding is performed by devices that do not communicate with one another, but simply encode data to memory and/or retrieve and decode data from memory.

For convenience of description, embodiments of the invention are described herein, for example, by reference to High-Efficiency Video Coding (HEVC) or to the reference software of Versatile Video coding (VVC), the next generation video coding standard developed by the Joint Collaboration Team on Video Coding (JCT-VC) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG). One of ordinary skill in the art will understand that embodiments of the invention are not limited to HEVC or VVC.

Encoder and Encoding Method

Fig. 2 shows a schematic block diagram of an example video encoder 20 that is configured to implement the techniques of the present application. In the example of Fig. 2, the video encoder 20 comprises an input 201 (or input interface 201), a residual calculation unit 204, a transform processing unit 206, a quantization unit 208, an inverse quantization unit 210, and inverse transform processing unit 212, a reconstruction unit 214, a loop filter unit 220, a decoded picture buffer (DPB) 230, a mode selection unit 260, an entropy encoding unit 270 and an output 272 (or output interface 272). The mode selection unit 260 may include an inter

prediction unit 244, an intra prediction unit 254 and a partitioning unit 262. Inter prediction unit 244 may include a motion estimation unit and a motion compensation unit (not shown). A video encoder 20 as shown in Fig. 2 may also be referred to as hybrid video encoder or a video encoder according to a hybrid video codec.

- 5 The residual calculation unit 204, the transform processing unit 206, the quantization unit 208, the mode selection unit 260 may be referred to as forming a forward signal path of the encoder 20, whereas the inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214, the buffer 216, the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 244 and the intra-prediction unit 254 may be referred to as forming a backward signal path of the video encoder 20, wherein the backward
10 signal path of the video encoder 20 corresponds to the signal path of the decoder (see video decoder 30 in Fig. 3). The inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214, the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 244 and the intra-prediction unit 254 are also referred to forming
15 the “built-in decoder” of video encoder 20.

Pictures & Picture Partitioning (Pictures & Blocks)

- The encoder 20 may be configured to receive, e.g. via input 201, a picture 17 (or picture data 17), e.g. picture of a sequence of pictures forming a video or video sequence. The received picture or picture data may also be a pre-processed picture 19 (or pre-processed picture data
20 19). For sake of simplicity the following description refers to the picture 17. The picture 17 may also be referred to as current picture or picture to be coded (in particular in video coding to distinguish the current picture from other pictures, e.g. previously encoded and/or decoded pictures of the same video sequence, i.e. the video sequence which also comprises the current picture).
- 25 A (digital) picture is or can be regarded as a two-dimensional array or matrix of samples with intensity values. A sample in the array may also be referred to as pixel (short form of picture

element) or a pel. The number of samples in horizontal and vertical direction (or axis) of the array or picture define the size and/or resolution of the picture. For representation of color, typically three color components are employed, i.e. the picture may be represented or include three sample arrays. In RGB format or color space a picture comprises a corresponding red, green and blue sample array. However, in video coding each pixel is typically represented in a luminance and chrominance format or color space, e.g. YCbCr, which comprises a luminance component indicated by Y (sometimes also L is used instead) and two chrominance components indicated by Cb and Cr. The luminance (or short luma) component Y represents the brightness or grey level intensity (e.g. like in a grey-scale picture), while the two chrominance (or short chroma) components Cb and Cr represent the chromaticity or color information components. Accordingly, a picture in YCbCr format comprises a luminance sample array of luminance sample values (Y), and two chrominance sample arrays of chrominance values (Cb and Cr). Pictures in RGB format may be converted or transformed into YCbCr format and vice versa, the process is also known as color transformation or conversion. If a picture is monochrome, the picture may comprise only a luminance sample array. Accordingly, a picture may be, for example, an array of luma samples in monochrome format or an array of luma samples and two corresponding arrays of chroma samples in 4:2:0, 4:2:2, and 4:4:4 colour format.

Embodiments of the video encoder 20 may comprise a picture partitioning unit (not depicted in Fig. 2) configured to partition the picture 17 into a plurality of (typically non-overlapping) picture blocks 203. These blocks may also be referred to as root blocks, macro blocks (H.264/AVC) or coding tree blocks (CTB) or coding tree units (CTU) (H.265/HEVC and VVC). The picture partitioning unit may be configured to use the same block size for all pictures of a video sequence and the corresponding grid defining the block size, or to change the block size between pictures or subsets or groups of pictures, and partition each picture into the corresponding blocks.

In further embodiments, the video encoder may be configured to receive directly a block 203 of the picture 17, e.g. one, several or all blocks forming the picture 17. The picture block 203 may also be referred to as current picture block or picture block to be coded.

Like the picture 17, the picture block 203 again is or can be regarded as a two-dimensional
5 array or matrix of samples with intensity values (sample values), although of smaller dimension than the picture 17. In other words, the block 203 may comprise, e.g., one sample array (e.g. a luma array in case of a monochrome picture 17, or a luma or chroma array in case of a color picture) or three sample arrays (e.g. a luma and two chroma arrays in case of a color picture 17) or any other number and/or kind of arrays depending on the color format
10 applied. The number of samples in horizontal and vertical direction (or axis) of the block 203 define the size of block 203. Accordingly, a block may, for example, an $M \times N$ (M -column by N -row) array of samples, or an $M \times N$ array of transform coefficients.

Embodiments of the video encoder 20 as shown in Fig. 2 may be configured to encode the picture 17 block by block, e.g. the encoding and prediction is performed per block 203.

15 Embodiments of the video encoder 20 as shown in Fig. 2 may be further configured to partition and/or encode the picture by using slices (also referred to as video slices), wherein a picture may be partitioned into or encoded using one or more slices (typically non-overlapping), and each slice may comprise one or more blocks (e.g. CTUs).

Embodiments of the video encoder 20 as shown in Fig. 2 may be further configured to
20 partition and/or encode the picture by using tile groups (also referred to as video tile groups) and/or tiles (also referred to as video tiles), wherein a picture may be partitioned into or encoded using one or more tile groups (typically non-overlapping), and each tile group may comprise, e.g. one or more blocks (e.g. CTUs) or one or more tiles, wherein each tile, e.g.
25 complete or fractional blocks.

Residual Calculation

The residual calculation unit 204 may be configured to calculate a residual block 205 (also referred to as residual 205) based on the picture block 203 and a prediction block 265 (further details about the prediction block 265 are provided later), e.g. by subtracting sample values of
5 the prediction block 265 from sample values of the picture block 203, sample by sample (pixel by pixel) to obtain the residual block 205 in the sample domain.

Transform

The transform processing unit 206 may be configured to apply a transform, e.g. a discrete cosine transform (DCT) or discrete sine transform (DST), on the sample values of the
10 residual block 205 to obtain transform coefficients 207 in a transform domain. The transform coefficients 207 may also be referred to as transform residual coefficients and represent the residual block 205 in the transform domain.

The transform processing unit 206 may be configured to apply integer approximations of DCT/DST, such as the transforms specified for H.265/HEVC. Compared to an orthogonal
15 DCT transform, such integer approximations are typically scaled by a certain factor. In order to preserve the norm of the residual block which is processed by forward and inverse transforms, additional scaling factors are applied as part of the transform process. The scaling factors are typically chosen based on certain constraints like scaling factors being a power of two for shift operations, bit depth of the transform coefficients, tradeoff between accuracy
20 and implementation costs, etc. Specific scaling factors are, for example, specified for the inverse transform, e.g. by inverse transform processing unit 212 (and the corresponding inverse transform, e.g. by inverse transform processing unit 312 at video decoder 30) and corresponding scaling factors for the forward transform, e.g. by transform processing unit 206, at an encoder 20 may be specified accordingly.

25 Embodiments of the video encoder 20 (respectively transform processing unit 206) may be configured to output transform parameters, e.g. a type of transform or transforms, e.g.

directly or encoded or compressed via the entropy encoding unit 270, so that, e.g., the video decoder 30 may receive and use the transform parameters for decoding.

Quantization

The quantization unit 208 may be configured to quantize the transform coefficients 207 to
5 obtain quantized coefficients 209, e.g. by applying scalar quantization or vector quantization.

The quantized coefficients 209 may also be referred to as quantized transform coefficients 209 or quantized residual coefficients 209.

The quantization process may reduce the bit depth associated with some or all of the transform coefficients 207. For example, an n-bit transform coefficient may be rounded down
10 to an m-bit Transform coefficient during quantization, where n is greater than m. The degree of quantization may be modified by adjusting a quantization parameter (QP). For example for scalar quantization, different scaling may be applied to achieve finer or coarser quantization. Smaller quantization step sizes correspond to finer quantization, whereas larger quantization step sizes correspond to coarser quantization. The applicable quantization step size may be
15 indicated by a quantization parameter (QP). The quantization parameter may for example be an index to a predefined set of applicable quantization step sizes. For example, small quantization parameters may correspond to fine quantization (small quantization step sizes) and large quantization parameters may correspond to coarse quantization (large quantization step sizes) or vice versa. The quantization may include division by a quantization step size
20 and a corresponding and/or the inverse dequantization, e.g. by inverse quantization unit 210, may include multiplication by the quantization step size. Embodiments according to some standards, e.g. HEVC, may be configured to use a quantization parameter to determine the quantization step size. Generally, the quantization step size may be calculated based on a quantization parameter using a fixed point approximation of an equation including division.
25 Additional scaling factors may be introduced for quantization and dequantization to restore the norm of the residual block, which might get modified because of the scaling used in the

fixed point approximation of the equation for quantization step size and quantization parameter. In one example implementation, the scaling of the inverse transform and dequantization might be combined. Alternatively, customized quantization tables may be used and signaled from an encoder to a decoder, e.g. in a bitstream. The quantization is a
5 lossy operation, wherein the loss increases with increasing quantization step sizes. Embodiments of the video encoder 20 (respectively quantization unit 208) may be configured to output quantization parameters (QP), e.g. directly or encoded via the entropy encoding unit 270, so that, e.g., the video decoder 30 may receive and apply the quantization parameters for decoding.

10 Inverse Quantization

The inverse quantization unit 210 is configured to apply the inverse quantization of the quantization unit 208 on the quantized coefficients to obtain dequantized coefficients 211, e.g. by applying the inverse of the quantization scheme applied by the quantization unit 208 based on or using the same quantization step size as the quantization unit 208. The dequantized
15 coefficients 211 may also be referred to as dequantized residual coefficients 211 and correspond - although typically not identical to the transform coefficients due to the loss by quantization - to the transform coefficients 207.

Inverse Transform

The inverse transform processing unit 212 is configured to apply the inverse transform of the
20 transform applied by the transform processing unit 206, e.g. an inverse discrete cosine transform (DCT) or inverse discrete sine transform (DST) or other inverse transforms, to obtain a reconstructed residual block 213 (or corresponding dequantized coefficients 213) in the sample domain. The reconstructed residual block 213 may also be referred to as transform block 213.

Reconstruction

The reconstruction unit 214 (e.g. adder or summer 214) is configured to add the transform block 213 (i.e. reconstructed residual block 213) to the prediction block 265 to obtain a reconstructed block 215 in the sample domain, e.g. by adding – sample by sample - the sample values of the reconstructed residual block 213 and the sample values of the prediction block 265.

Filtering

The loop filter unit 220 (or short “loop filter” 220), is configured to filter the reconstructed block 215 to obtain a filtered block 221, or in general, to filter reconstructed samples to obtain filtered samples. The loop filter unit is, e.g., configured to smooth pixel transitions, or otherwise improve the video quality. The loop filter unit 220 may comprise one or more loop filters such as a de-blocking filter, a sample-adaptive offset (SAO) filter or one or more other filters, e.g. a bilateral filter, an adaptive loop filter (ALF), a sharpening, a smoothing filters or a collaborative filters, or any combination thereof. Although the loop filter unit 220 is shown in FIG. 2 as being an in loop filter, in other configurations, the loop filter unit 220 may be implemented as a post loop filter. The filtered block 221 may also be referred to as filtered reconstructed block 221.

Embodiments of the video encoder 20 (respectively loop filter unit 220) may be configured to output loop filter parameters (such as sample adaptive offset information), e.g. directly or encoded via the entropy encoding unit 270, so that, e.g., a decoder 30 may receive and apply the same loop filter parameters or respective loop filters for decoding.

Decoded Picture Buffer

The decoded picture buffer (DPB) 230 may be a memory that stores reference pictures, or in general reference picture data, for encoding video data by video encoder 20. The DPB 230 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM

(MRAM), resistive RAM (RRAM), or other types of memory devices. The decoded picture buffer (DPB) 230 may be configured to store one or more filtered blocks 221. The decoded picture buffer 230 may be further configured to store other previously filtered blocks, e.g. previously reconstructed and filtered blocks 221, of the same current picture or of different pictures, e.g. previously reconstructed pictures, and may provide complete previously reconstructed, i.e. decoded, pictures (and corresponding reference blocks and samples) and/or a partially reconstructed current picture (and corresponding reference blocks and samples), for example for inter prediction. The decoded picture buffer (DPB) 230 may be also configured to store one or more unfiltered reconstructed blocks 215, or in general unfiltered reconstructed samples, e.g. if the reconstructed block 215 is not filtered by loop filter unit 220, or any other further processed version of the reconstructed blocks or samples.

Mode Selection (Partitioning & Prediction)

The mode selection unit 260 comprises partitioning unit 262, inter-prediction unit 244 and intra-prediction unit 254, and is configured to receive or obtain original picture data, e.g. an original block 203 (current block 203 of the current picture 17), and reconstructed picture data, e.g. filtered and/or unfiltered reconstructed samples or blocks of the same (current) picture and/or from one or a plurality of previously decoded pictures, e.g. from decoded picture buffer 230 or other buffers (e.g. line buffer, not shown).. The reconstructed picture data is used as reference picture data for prediction, e.g. inter-prediction or intra-prediction, to obtain a prediction block 265 or predictor 265.

Mode selection unit 260 may be configured to determine or select a partitioning for a current block prediction mode (including no partitioning) and a prediction mode (e.g. an intra or inter prediction mode) and generate a corresponding prediction block 265, which is used for the calculation of the residual block 205 and for the reconstruction of the reconstructed block 215.

Embodiments of the mode selection unit 260 may be configured to select the partitioning and the prediction mode (e.g. from those supported by or available for mode selection unit 260), which provide the best match or in other words the minimum residual (minimum residual means better compression for transmission or storage), or a minimum signaling overhead (minimum signaling overhead means better compression for transmission or storage), or
5 which considers or balances both. The mode selection unit 260 may be configured to determine the partitioning and prediction mode based on rate distortion optimization (RDO), i.e. select the prediction mode which provides a minimum rate distortion. Terms like “best”, “minimum”, “optimum” etc. in this context do not necessarily refer to an overall “best”,
10 “minimum”, “optimum”, etc. but may also refer to the fulfillment of a termination or selection criterion like a value exceeding or falling below a threshold or other constraints leading potentially to a “sub-optimum selection” but reducing complexity and processing time.

In other words, the partitioning unit 262 may be configured to partition the block 203 into
15 smaller block partitions or sub-blocks (which form again blocks), e.g. iteratively using quad-tree-partitioning (QT), binary partitioning (BT) or triple-tree-partitioning (TT) or any combination thereof, and to perform, e.g., the prediction for each of the block partitions or sub-blocks, wherein the mode selection comprises the selection of the tree-structure of the partitioned block 203 and the prediction modes are applied to each of the block partitions or
20 sub-blocks.

In the following the partitioning (e.g. by partitioning unit 260) and prediction processing (by inter-prediction unit 244 and intra-prediction unit 254) performed by an example video encoder 20 will be explained in more detail.

Partitioning

25 The partitioning unit 262 may partition (or split) a current block 203 into smaller partitions, e.g. smaller blocks of square or rectangular size. These smaller blocks (which may also be

referred to as sub-blocks) may be further partitioned into even smaller partitions. This is also referred to tree-partitioning or hierarchical tree-partitioning, wherein a root block, e.g. at root tree-level 0 (hierarchy-level 0, depth 0), may be recursively partitioned, e.g. partitioned into two or more blocks of a next lower tree-level, e.g. nodes at tree-level 1 (hierarchy-level 1, depth 1), wherein these blocks may be again partitioned into two or more blocks of a next lower level, e.g. tree-level 2 (hierarchy-level 2, depth 2), etc. until the partitioning is terminated, e.g. because a termination criterion is fulfilled, e.g. a maximum tree depth or minimum block size is reached. Blocks which are not further partitioned are also referred to as leaf-blocks or leaf nodes of the tree. A tree using partitioning into two partitions is referred to as binary-tree (BT), a tree using partitioning into three partitions is referred to as ternary-tree (TT), and a tree using partitioning into four partitions is referred to as quad-tree (QT).

As mentioned before, the term “block” as used herein may be a portion, in particular a square or rectangular portion, of a picture. With reference, for example, to HEVC and VVC, the block may be or correspond to a coding tree unit (CTU), a coding unit (CU), prediction unit (PU), and transform unit (TU) and/or to the corresponding blocks, e.g. a coding tree block (CTB), a coding block (CB), a transform block (TB) or prediction block (PB).

For example, a coding tree unit (CTU) may be or comprise a CTB of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples. Correspondingly, a coding tree block (CTB) may be an $N \times N$ block of samples for some value of N such that the division of a component into CTBs is a partitioning. A coding unit (CU) may be or comprise a coding block of luma samples, two corresponding coding blocks of chroma samples of a picture that has three sample arrays, or a coding block of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples.

Correspondingly a coding block (CB) may be an $M \times N$ block of samples for some values of M and N such that the division of a CTB into coding blocks is a partitioning.

In embodiments, e.g., according to HEVC, a coding tree unit (CTU) may be split into CUs by using a quad-tree structure denoted as coding tree. The decision whether to code a picture

5 area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the CU level.

Each CU can be further split into one, two or four PUs according to the PU splitting type.

Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. After obtaining the residual block by applying the prediction process based on the PU splitting type, a CU can be partitioned into transform

10 units (TUs) according to another quadtree structure similar to the coding tree for the CU.

In embodiments, e.g., according to the latest video coding standard currently in development, which is referred to as Versatile Video Coding (VVC), a combined Quad-tree and binary tree (QTBT) partitioning is for example used to partition a coding block. In the QTBT block structure, a CU can have either a square or rectangular shape. For example, a coding tree unit

15 (CTU) is first partitioned by a quadtree structure. The quadtree leaf nodes are further partitioned by a binary tree or ternary (or triple) tree structure. The partitioning tree leaf nodes are called coding units (CUs), and that segmentation is used for prediction and transform processing without any further partitioning. This means that the CU, PU and TU have the same block size in the QTBT coding block structure. In parallel, multiple partition, 20 for example, triple tree partition may be used together with the QTBT block structure.

In one example, the mode selection unit 260 of video encoder 20 may be configured to perform any combination of the partitioning techniques described herein.

As described above, the video encoder 20 is configured to determine or select the best or an optimum prediction mode from a set of (e.g. pre-determined) prediction modes. The set of

25 prediction modes may comprise, e.g., intra-prediction modes and/or inter-prediction modes.

Intra-Prediction

The set of intra-prediction modes may comprise 35 different intra-prediction modes, e.g. non-directional modes like DC (or mean) mode and planar mode, or directional modes, e.g. as defined in HEVC, or may comprise 67 different intra-prediction modes, e.g.

- 5 non-directional modes like DC (or mean) mode and planar mode, or directional modes, e.g. as defined for VVC.

The intra-prediction unit 254 is configured to use reconstructed samples of neighboring blocks of the same current picture to generate an intra-prediction block 265 according to an intra-prediction mode of the set of intra-prediction modes.

- 10 The intra prediction unit 254 (or in general the mode selection unit 260) is further configured to output intra-prediction parameters (or in general information indicative of the selected intra prediction mode for the block) to the entropy encoding unit 270 in form of syntax elements 266 for inclusion into the encoded picture data 21, so that, e.g., the video decoder 30 may receive and use the prediction parameters for decoding.

15 Inter-Prediction

The set of (or possible) inter-prediction modes depends on the available reference pictures (i.e. previous at least partially decoded pictures, e.g. stored in DBP 230) and other inter-prediction parameters, e.g. whether the whole reference picture or only a part, e.g. a search window area around the area of the current block, of the reference picture is used for
20 searching for a best matching reference block, and/or e.g. whether pixel interpolation is applied, e.g. half/semi-pel and/or quarter-pel interpolation, or not.

Additional to the above prediction modes, skip mode and/or direct mode may be applied.

- The inter prediction unit 244 may include a motion estimation (ME) unit and a motion compensation (MC) unit (both not shown in Fig 2). The motion estimation unit may be
25 configured to receive or obtain the picture block 203 (current picture block 203 of the current picture 17) and a decoded picture 231, or at least one or a plurality of previously

reconstructed blocks, e.g. reconstructed blocks of one or a plurality of other/different previously decoded pictures 231, for motion estimation. E.g. a video sequence may comprise the current picture and the previously decoded pictures 231, or in other words, the current picture and the previously decoded pictures 231 may be part of or form a sequence of pictures forming a video sequence.

The encoder 20 may, e.g., be configured to select a reference block from a plurality of reference blocks of the same or different pictures of the plurality of other pictures and provide a reference picture (or reference picture index) and/or an offset (spatial offset) between the position (x, y coordinates) of the reference block and the position of the current block as inter prediction parameters to the motion estimation unit. This offset is also called motion vector (MV).

The motion compensation unit is configured to obtain, e.g. receive, an inter prediction parameter and to perform inter prediction based on or using the inter prediction parameter to obtain an inter prediction block 265. Motion compensation, performed by the motion compensation unit, may involve fetching or generating the prediction block based on the motion/block vector determined by motion estimation, possibly performing interpolations to sub-pixel precision. Interpolation filtering may generate additional pixel samples from known pixel samples, thus potentially increasing the number of candidate prediction blocks that may be used to code a picture block. Upon receiving the motion vector for the PU of the current picture block, the motion compensation unit may locate the prediction block to which the motion vector points in one of the reference picture lists.

The motion compensation unit may also generate syntax elements associated with the blocks and video slices for use by video decoder 30 in decoding the picture blocks of the video slice. In addition or as an alternative to slices and respective syntax elements, tile groups and/or tiles and respective syntax elements may be generated or used.

Entropy Coding

The entropy encoding unit 270 is configured to apply, for example, an entropy encoding algorithm or scheme (e.g. a variable length coding (VLC) scheme, an context adaptive VLC scheme (CAVLC), an arithmetic coding scheme, a binarization, a context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy encoding methodology or technique) or bypass (no compression) on the quantized coefficients 209, inter prediction parameters, intra prediction parameters, loop filter parameters and/or other syntax elements to obtain encoded picture data 21 which can be output via the output 272, e.g. in the form of an encoded bitstream 21, so that, e.g., the video decoder 30 may receive and use the parameters for decoding. The encoded bitstream 21 may be transmitted to video decoder 30, or stored in a memory for later transmission or retrieval by video decoder 30. Other structural variations of the video encoder 20 can be used to encode the video stream. For example, a non-transform based encoder 20 can quantize the residual signal directly without the transform processing unit 206 for certain blocks or frames. In another implementation, an encoder 20 can have the quantization unit 208 and the inverse quantization unit 210 combined into a single unit.

Decoder and Decoding Method

Fig. 3 shows an example of a video decoder 30 that is configured to implement the techniques of this present application. The video decoder 30 is configured to receive encoded picture data 21 (e.g. encoded bitstream 21), e.g. encoded by encoder 20, to obtain a decoded picture 331. The encoded picture data or bitstream comprises information for decoding the encoded picture data, e.g. data that represents picture blocks of an encoded video slice (and/or tile groups or tiles) and associated syntax elements. In the example of Fig. 3, the decoder 30 comprises an entropy decoding unit 304, an inverse quantization unit 310, an inverse transform processing unit 312, a reconstruction unit 314 (e.g.

a summer 314), a loop filter 320, a decoded picture buffer (DBP) 330, a mode application unit 360, an inter prediction unit 344 and an intra prediction unit 354. Inter prediction unit 344 may be or include a motion compensation unit. Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to
5 video encoder 100 from FIG. 2.

As explained with regard to the encoder 20, the inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214 the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 344 and the intra prediction unit 354 are also referred to as forming the “built-in decoder” of video encoder 20. Accordingly, the
10 inverse quantization unit 310 may be identical in function to the inverse quantization unit 110, the inverse transform processing unit 312 may be identical in function to the inverse transform processing unit 212, the reconstruction unit 314 may be identical in function to reconstruction unit 214, the loop filter 320 may be identical in function to the loop filter 220, and the decoded picture buffer 330 may be identical in function to the decoded picture buffer
15 230. Therefore, the explanations provided for the respective units and functions of the video 20 encoder apply correspondingly to the respective units and functions of the video decoder 30.

Entropy Decoding

The entropy decoding unit 304 is configured to parse the bitstream 21 (or in general encoded
20 picture data 21) and perform, for example, entropy decoding to the encoded picture data 21 to obtain, e.g., quantized coefficients 309 and/or decoded coding parameters (not shown in Fig. 3), e.g. any or all of inter prediction parameters (e.g. reference picture index and motion vector), intra prediction parameter (e.g. intra prediction mode or index), transform parameters, quantization parameters, loop filter parameters, and/or other syntax elements. Entropy
25 decoding unit 304 maybe configured to apply the decoding algorithms or schemes corresponding to the encoding schemes as described with regard to the entropy encoding unit

270 of the encoder 20. Entropy decoding unit 304 may be further configured to provide inter prediction parameters, intra prediction parameter and/or other syntax elements to the mode application unit 360 and other parameters to other units of the decoder 30. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level. In addition or as an alternative to slices and respective syntax elements, tile groups and/or tiles and respective syntax elements may be received and/or used.

Inverse Quantization

The inverse quantization unit 310 may be configured to receive quantization parameters (QP) (or in general information related to the inverse quantization) and quantized coefficients from the encoded picture data 21 (e.g. by parsing and/or decoding, e.g. by entropy decoding unit 304) and to apply based on the quantization parameters an inverse quantization on the decoded quantized coefficients 309 to obtain dequantized coefficients 311, which may also be referred to as transform coefficients 311. The inverse quantization process may include use of a quantization parameter determined by video encoder 20 for each video block in the video slice (or tile or tile group) to determine a degree of quantization and, likewise, a degree of inverse quantization that should be applied.

Inverse Transform

Inverse transform processing unit 312 may be configured to receive dequantized coefficients 311, also referred to as transform coefficients 311, and to apply a transform to the dequantized coefficients 311 in order to obtain reconstructed residual blocks 213 in the sample domain. The reconstructed residual blocks 213 may also be referred to as transform blocks 313. The transform may be an inverse transform, e.g., an inverse DCT, an inverse DST, an inverse integer transform, or a conceptually similar inverse transform process. The inverse transform processing unit 312 may be further configured to receive transform parameters or corresponding information from the encoded picture data 21 (e.g. by parsing

and/or decoding, e.g. by entropy decoding unit 304) to determine the transform to be applied to the dequantized coefficients 311.

Reconstruction

The reconstruction unit 314 (e.g. adder or summer 314) may be configured to add the
5 reconstructed residual block 313, to the prediction block 365 to obtain a reconstructed block 315 in the sample domain, e.g. by adding the sample values of the reconstructed residual block 313 and the sample values of the prediction block 365.

Filtering

The loop filter unit 320 (either in the coding loop or after the coding loop) is configured to
10 filter the reconstructed block 315 to obtain a filtered block 321, e.g. to smooth pixel transitions, or otherwise improve the video quality. The loop filter unit 320 may comprise one or more loop filters such as a de-blocking filter, a sample-adaptive offset (SAO) filter or one or more other filters, e.g. a bilateral filter, an adaptive loop filter (ALF), a sharpening, a smoothing filters or a collaborative filters, or any combination thereof. Although the loop
15 filter unit 320 is shown in FIG. 3 as being an in loop filter, in other configurations, the loop filter unit 320 may be implemented as a post loop filter.

Decoded Picture Buffer

The decoded video blocks 321 of a picture are then stored in decoded picture buffer 330, which stores the decoded pictures 331 as reference pictures for subsequent motion
20 compensation for other pictures and/or for output respectively display.

The decoder 30 is configured to output the decoded picture 311, e.g. via output 312, for presentation or viewing to a user.

Prediction

The inter prediction unit 344 may be identical to the inter prediction unit 244 (in particular to
25 the motion compensation unit) and the intra prediction unit 354 may be identical to the inter prediction unit 254 in function, and performs split or partitioning decisions and prediction

based on the partitioning and/or prediction parameters or respective information received from the encoded picture data 21 (e.g. by parsing and/or decoding, e.g. by entropy decoding unit 304). Mode application unit 360 may be configured to perform the prediction (intra or inter prediction) per block based on reconstructed pictures, blocks or respective samples
5 (filtered or unfiltered) to obtain the prediction block 365.

When the video slice is coded as an intra coded (I) slice, intra prediction unit 354 of mode application unit 360 is configured to generate prediction block 365 for a picture block of the current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current picture. When the video picture is coded as an inter coded (i.e.,
10 B, or P) slice, inter prediction unit 344 (e.g. motion compensation unit) of mode application unit 360 is configured to produce prediction blocks 365 for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 304. For inter prediction, the prediction blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may construct
15 the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in DPB 330. The same or similar may be applied for or by embodiments using tile groups (e.g. video tile groups) and/or tiles (e.g. video tiles) in addition or alternatively to slices (e.g. video slices), e.g. a video may be coded using I, P or B tile groups and /or tiles.

20 Mode application unit 360 is configured to determine the prediction information for a video block of the current video slice by parsing the motion vectors or related information and other syntax elements, and uses the prediction information to produce the prediction blocks for the current video block being decoded. For example, the mode application unit 360 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction)
25 used to code the video blocks of the video slice, an inter prediction slice type (e.g., B slice, P slice, or GPB slice), construction information for one or more of the reference picture lists for

the slice, motion vectors for each inter encoded video block of the slice, inter prediction status for each inter coded video block of the slice, and other information to decode the video blocks in the current video slice. The same or similar may be applied for or by embodiments using tile groups (e.g. video tile groups) and/or tiles (e.g. video tiles) in addition or
5 alternatively to slices (e.g. video slices), e.g. a video may be coded using I, P or B tile groups and/or tiles.

Embodiments of the video decoder 30 as shown in Fig. 3 may be configured to partition and/or decode the picture by using slices (also referred to as video slices), wherein a picture may be partitioned into or decoded using one or more slices (typically non-overlapping), and
10 each slice may comprise one or more blocks (e.g. CTUs).

Embodiments of the video decoder 30 as shown in Fig. 3 may be configured to partition and/or decode the picture by using tile groups (also referred to as video tile groups) and/or tiles (also referred to as video tiles), wherein a picture may be partitioned into or decoded using one or more tile groups (typically non-overlapping), and each tile group may comprise,
15 e.g. one or more blocks (e.g. CTUs) or one or more tiles, wherein each tile, e.g. may be of rectangular shape and may comprise one or more blocks (e.g. CTUs), e.g. complete or fractional blocks.

Other variations of the video decoder 30 can be used to decode the encoded picture data 21. For example, the decoder 30 can produce the output video stream without the loop filtering unit 320. For example, a non-transform based decoder 30 can inverse-quantize the residual
20 signal directly without the inverse-transform processing unit 312 for certain blocks or frames. In another implementation, the video decoder 30 can have the inverse-quantization unit 310 and the inverse-transform processing unit 312 combined into a single unit.

It should be understood that, in the encoder 20 and the decoder 30, a processing result of a
25 current step may be further processed and then output to the next step. For example, after interpolation filtering, motion vector derivation or loop filtering, a further operation, such as

Clip or shift, may be performed on the processing result of the interpolation filtering, motion vector derivation or loop filtering.

- It should be noted that further operations may be applied to the derived motion vectors of current block (including but not limit to control point motion vectors of affine mode, sub-block motion vectors in affine, planar, ATMVP modes, temporal motion vectors, and so on). For example, the value of motion vector is constrained to a predefined range according to its representing bit. If the representing bit of motion vector is bitDepth, then the range is $-2^{(\text{bitDepth}-1)} \sim 2^{(\text{bitDepth}-1)-1}$, where “ $\wedge$ ” means exponentiation. For example, if bitDepth is set equal to 16, the range is $-32768 \sim 32767$; if bitDepth is set equal to 18, the range is $-131072 \sim 131071$. For example, the value of the derived motion vector (e.g. the MVs of four 4x4 sub-blocks within one 8x8 block) is constrained such that the max difference between integer parts of the four 4x4 sub-block MVs is no more than N pixels, such as no more than 1 pixel. Here provides two methods for constraining the motion vector according to the bitDepth.
- Method 1: remove the overflow MSB (most significant bit) by flowing operations

$$ux = (mvx + 2^{\text{bitDepth}}) \% 2^{\text{bitDepth}} \quad (1)$$

$$mvx = (ux \geq 2^{\text{bitDepth}-1}) ? (ux - 2^{\text{bitDepth}}) : ux \quad (2)$$

$$uy = (mvy + 2^{\text{bitDepth}}) \% 2^{\text{bitDepth}} \quad (3)$$

$$mvy = (uy \geq 2^{\text{bitDepth}-1}) ? (uy - 2^{\text{bitDepth}}) : uy \quad (4)$$

- where mvx is a horizontal component of a motion vector of an image block or a sub-block, mvy is a vertical component of a motion vector of an image block or a sub-block, and ux and uy indicates an intermediate value;

- For example, if the value of mvx is -32769, after applying formula (1) and (2), the resulting value is 32767. In computer system, decimal numbers are stored as two's complement. The two's complement of -32769 is 1,0111,1111,1111,1111 (17 bits), then the MSB is discarded, so the resulting two's complement is 0111,1111,1111,1111 (decimal number is 32767), which is same as the output by applying formula (1) and (2).

$$ux = (mvp_x + mvd_x + 2^{\text{bitDepth}}) \% 2^{\text{bitDepth}} \quad (5)$$

$$mvx = (ux \geq 2^{\text{bitDepth}-1}) ? (ux - 2^{\text{bitDepth}}) : ux \quad (6)$$

$$uy = (mvp_y + mvd_y + 2^{\text{bitDepth}}) \% 2^{\text{bitDepth}} \quad (7)$$

$$mvy = (uy \geq 2^{\text{bitDepth}-1}) ? (uy - 2^{\text{bitDepth}}) : uy \quad (8)$$

5 The operations may be applied during the sum of mvp and mvd, as shown in formula (5) to (8).

Method 2: remove the overflow MSB by clipping the value

$$vx = \text{Clip3}(-2^{\text{bitDepth}-1}, 2^{\text{bitDepth}-1} - 1, vx)$$

$$vy = \text{Clip3}(-2^{\text{bitDepth}-1}, 2^{\text{bitDepth}-1} - 1, vy)$$

10 where vx is a horizontal component of a motion vector of an image block or a sub-block, vy is a vertical component of a motion vector of an image block or a sub-block; x, y and z respectively correspond to three input value of the MV clipping process, and the definition of function Clip3 is as follow:

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; & z < x \\ y & ; & z > y \\ z & ; & \text{otherwise} \end{cases}$$

15 FIG. 4 is a schematic diagram of a video coding device 400 according to an embodiment of the disclosure. The video coding device 400 is suitable for implementing the disclosed embodiments as described herein. In an embodiment, the video coding device 400 may be a decoder such as video decoder 30 of FIG. 1A or an encoder such as video encoder 20 of FIG. 1A.

20 The video coding device 400 comprises ingress ports 410 (or input ports 410) and receiver units (Rx) 420 for receiving data; a processor, logic unit, or central processing unit (CPU) 430 to process the data; transmitter units (Tx) 440 and egress ports 450 (or output ports 450) for transmitting the data; and a memory 460 for storing the data. The video coding device 400 may also comprise optical-to-electrical (OE) components and electrical-to-optical (EO)

components coupled to the ingress ports 410, the receiver units 420, the transmitter units 440, and the egress ports 450 for egress or ingress of optical or electrical signals.

The processor 430 is implemented by hardware and software. The processor 430 may be implemented as one or more CPU chips, cores (e.g., as a multi-core processor), FPGAs,

5 ASICs, and DSPs. The processor 430 is in communication with the ingress ports 410, receiver units 420, transmitter units 440, egress ports 450, and memory 460. The processor 430 comprises a coding module 470. The coding module 470 implements the disclosed embodiments described above. For instance, the coding module 470 implements, processes, prepares, or provides the various coding operations. The inclusion of the coding module 470
10 therefore provides a substantial improvement to the functionality of the video coding device 400 and effects a transformation of the video coding device 400 to a different state.

Alternatively, the coding module 470 is implemented as instructions stored in the memory 460 and executed by the processor 430.

The memory 460 may comprise one or more disks, tape drives, and solid-state drives and
15 may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution. The memory 460 may be, for example, volatile and/or non-volatile and may be a read-only memory (ROM), random access memory (RAM), ternary content-addressable memory (TCAM), and/or static random-access memory (SRAM).

20 Fig. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from Fig. 1 according to an exemplary embodiment.

A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating
25 or processing information now-existing or hereafter developed. Although the disclosed

implementations can be practiced with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage

5 device can be used as the memory 504. The memory 504 can include code and data 506 that is accessed by the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here.

For example, the application programs 510 can include applications 1 through N, which
10 further include a video coding application that performs the methods described here.

The apparatus 500 can also include one or more output devices, such as a display 518. The display 518 may be, in one example, a touch sensitive display that combines a display with a touch sensitive element that is operable to sense touch inputs. The display 518 can be coupled to the processor 502 via the bus 512.

15 Although depicted here as a single bus, the bus 512 of the apparatus 500 can be composed of multiple buses. Further, the secondary storage 514 can be directly coupled to the other components of the apparatus 500 or can be accessed via a network and can comprise a single integrated unit such as a memory card or multiple units such as multiple memory cards. The apparatus 500 can thus be implemented in a wide variety of configurations.

20 Parameter Sets

Parameter sets are fundamentally similar and share the same basic design goals—namely bit rate efficiency, error resiliency, and providing systems layer interfaces. There is a hierarchy of parameter sets in HEVC (ITU-T H.265), including the Video Parameter Set (VPS), Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) which are similar to their
25 counterparts in AVC and VVC. Each slice references a single active PPS, SPS and VPS to access information used for decoding the slice. The PPS contains information which applies

to all slices in a picture, and hence all slices in a picture must refer to the same PPS. The slices in different pictures are also allowed to refer to the same PPS. Similarly, the SPS contains information which applies to all pictures in the same coded video sequence.

While the PPS may differ for separate pictures, it is common for many or all pictures in a coded video sequence to refer to the same PPS. Reusing parameter sets is bit rate efficient because it avoids the necessity to send shared information multiple times. It is also loss robust because it allows parameter set content to be carried by some more reliable external communication link or to be repeated frequently within the bitstream to ensure that it will not get lost.

10

Sequence Parameter Set (SPS)

The SPS contains parameters that apply to one or more layers of a coded video sequence and do not change from picture to picture within a coded video sequence. In particular, the SPS including information that signaling of subpicture.

15 Some parts of the following table shows a snapshot of part of the subpicture signaling in SPS in ITU JVET-Q2001-v11, with the download link as follows:

http://phenix.it-sudparis.eu/jvet/doc_end_user/documents/17_Brussels/wg11/JVET-Q2001-v11.zip. In the remaining part of the application this document will be named VVC draft 8 for simplicity.

<code>seq_parameter_set_rbsp() {</code>	Descriptor
<code>sps_seq_parameter_set_id</code>	<code>u(4)</code>
<code>sps_video_parameter_set_id</code>	<code>u(4)</code>
<code>sps_max_sublayers_minus1</code>	<code>u(3)</code>
<code>sps_reserved_zero_4bits</code>	<code>u(4)</code>
<code>sps_ptl_dpb_hrd_params_present_flag</code>	<code>u(1)</code>
<code>if(sps_ptl_dpb_hrd_params_present_flag)</code>	
<code>profile_tier_level(1, sps_max_sublayers_minus1)</code>	

gdr_enabled_flag	u(1)
chroma_format_idc	u(2)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
res_change_in_clvs_allowed_flag	u(1)
pic_width_max_in_luma_samples	ue(v)
pic_height_max_in_luma_samples	ue(v)
sps_conformance_window_flag	u(1)
if(sps_conformance_window_flag) {	
sps_conf_win_left_offset	ue(v)
sps_conf_win_right_offset	ue(v)
sps_conf_win_top_offset	ue(v)
sps_conf_win_bottom_offset	ue(v)
}	
sps_log2_ctu_size_minus5	u(2)
subpic_info_present_flag	u(1)
if(subpic_info_present_flag) {	
sps_num_subpics_minus1	ue(v)
sps_independent_subpics_flag	u(1)
for(i = 0; sps_num_subpics_minus1 > 0 && i <= sps_num_subpics_minus1; i++)	
{	
if(i > 0 && pic_width_max_in_luma_samples > CtbSizeY)	
subpic_ctu_top_left_x[i]	u(v)
if(i > 0 && pic_height_max_in_luma_samples > CtbSizeY) {	
subpic_ctu_top_left_y[i]	u(v)
if(i < sps_num_subpics_minus1 && pic_width_max_in_luma_samples > CtbSizeY)	
subpic_width_minus1[i]	u(v)
if(i < sps_num_subpics_minus1 && pic_height_max_in_luma_samples > CtbSizeY)	
subpic_height_minus1[i]	u(v)
if(!sps_independent_subpics_flag) {	

subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	
...	
...	

Some syntax elements in SPS signal position information and control flags of each subpicture.

The position information for the i-th subpicture includes:

- subpic_ctu_top_left_x[i], indicating the horizontal component of the top-left coordinate of the subpicture i in the picture; or
 - subpic_ctu_top_left_y[i] indicating the vertical component of the top-left coordinate of the subpicture i in the picture; or
 - subpic_width_minus1[i] indicating width of the subpicture i in the picture; or
 - subpic_height_minus1[i] indicating height of the subpicture i in the picture.
- Some syntax elements indicate the number of subpictures in the picture, e.g. sps_num_subpics_minus1.

Partitioning of a picture into CTUs, slices, tiles and subpictures

Partitioning of the picture into CTUs

- Picture is divided into a sequence of coding tree units (CTUs). The term CTU is sometimes used interchangeably as CTB (coding tree block). In an example, the term CTU is same as the CTU definition in the ITU-T H.265. For a picture that has three sample arrays, a CTU comprises an N×N block of luma samples, and two corresponding blocks of chroma samples. Figure 6 shows an example of a picture divided into CTUs. The size of CTUs may be same except for the CTUs located at the picture boundaries (where incomplete CTUs can be present).

Partitioning of the picture into tiles

In some examples, when tiles are enabled, picture is divided into rectangular-shaped groups of CTUs separated by vertical and/or horizontal boundaries. The vertical and horizontal tile
5 boundaries intersect the picture from top to bottom and from left picture boundary to right picture boundary respectively. Indication information related to the position of the horizontal tile boundaries and vertical tile boundaries are coded in bitstream.

Fig. 7 exemplifies partitioning of a picture into 9 tiles. In this example, the tile boundaries are marked with bold dashed lines. Tile-based raster scan order of CTUs with nine tiles of
10 different sizes in the picture.

When there are more than 1 tile dividing a picture vertically, the scan order of the CTUs is changed with respect to the raster scan order of CTUs in the picture. The CTUs are scanned according to the following rule:

1. Tiles are scanned from left to right and from top to bottom in raster scan order, which is
15 called the tile scan order. This means that starting from the top-left tile, all tiles in the same tile row are scanned from left to right. Then starting with the first tile in the second tile row (the tile row that is one below), all tiles in the second tile row are scanned from left to right. The process is repeated till all tiles are scanned.
2. For a tile, CTUs in this tile are scanned in raster scan order. For each CTU row, CTUs are
20 scanned from left to right and CTU rows are scanned from top to bottom. Fig. 7 exemplifies the scanning order of CTUs in the tiles, the numbers correspond to the CTUs indicate the scanning order.

The tile provides a partitioning of a picture in such a way that each tile is independently decodable from other tiles of the same picture, wherein decoding refers to entropy, residual,
25 and predictive decoding. Moreover with tiles, it is possible to partition the picture into similar sized regions. Therefore it is possible to process the tiles of a picture in parallel to each other,

which is preferable for multi-core processing environments where each processing core is identical to each other.

The terms processing order and scanning order are used as follows in the application:

Processing refers to encoding or decoding of CTUs in the encoder or decoder. Scanning order
5 relates to the indexing of the particular partition in a picture. The CTUs are indexed in increasing order in a picture according to a specified scan order. CTU scan order in tile means how the CTUs inside a tile is indexed, which might not be the same order in which they are processed (i.e. processing order).

10 Partitioning of the picture into slices

The slice concept provides a partitioning of a picture in such a way that each slice is independently decodable from other slices of the same picture, wherein decoding refers to entropy, residual, and predictive decoding. The difference to tiles is that slices may have arbitrary shapes that are not necessarily rectangular (more flexible in partitioning
15 possibilities), and the purpose of slice partitioning is not for parallel processing but is for packet size matching in transmission environments and error resilience.

A slice may comprises a complete picture or comprises a part of a picture. In ITU-T H.265, a slice comprises consecutive CTUs of a picture in processing order. A slice is identified according to a starting CTU address, the starting CTU address is signalled in a slice header or
20 in a picture parameter set or in other unit. In an example, the slice might be identified according to a starting tile address when the slices are required to comprise integer number of tiles.

In the draft 8 of VVC, a slice comprises an integer number of tiles or an integer number of consecutive CTU rows within a tile of a picture. Consequently, a vertical slice boundary is
25 also a vertical tile boundary. It is possible that a horizontal boundary of a slice is not a tile boundary, horizontal CTU boundaries may be comprised within a tile. In an example, when a

tile is split into multiple rectangular slices, each slice comprises an integer number of consecutive complete CTU rows within the tile.

- In some examples, there are two slice modes, the raster-scan slice mode and the rectangular slice mode. In the raster-scan slice mode, a slice comprises a sequence of tiles in a tile raster scan of a picture. In the rectangular slice mode, a slice comprises a number of tiles that collectively form a rectangular region of the picture, or a slice comprises a number of consecutive CTU rows of one tile that collectively form a rectangular region of the picture. Tiles within a rectangular slice are scanned in tile raster scan order within the rectangular region corresponding to that slice.
- 10 All slices of a picture collectively form the entire picture, all CTUs of a picture are comprised in one of the slices of a picture. Similar rules apply for tiles and subpictures.

Partitioning of the picture into subpictures

- A subpicture may be a rectangular partition of a picture. A subpicture may be the whole picture or a part of the picture. A subpicture is partitioning of a picture in such a way that each subpicture is independently decodable from other subpictures of the entire video sequence. In VVC draft 8, when `subpic_treated_as_pic_flag[i]` indication is true (e.g. a value of `subpic_treated_as_pic_flag[i]` is 1) for subpicture *i*, the subpicture *i* is independently decodable from other subpictures of the entire video sequence.
- 20 The difference between the subpicture and tiles or slices is that, subpictures refer to an independently decodable video sequence. For tiles and slices, independent decoding is performed in a single picture of a video sequence.

- In VVC draft 8, a subpicture comprises one or more slices that collectively cover a rectangular region of a picture. Consequently, each subpicture boundary is always a slice boundary, and each vertical subpicture boundary is always a vertical tile boundary.
- 25

Figure 8 provides an example of tiles, slices and subpictures. In figure 8, an example of picture containing 4 tiles (2 tile column and 2 tile rows), 4 rectangular slices, and 3 subpictures. The subpicture 2 contains two slices, subpicture 1 and 3 contain 1 slice each. Subpicture 1 contains 2 tiles, subpicture 2 and 3 contain 1 tile each. Subpicture, slice and tile boundaries all coincide with CTU boundaries. In the boundaries are shown with small offset to each other in order to make them distinguishable.

In one example as shown in Figure 8, a picture is partitioned into 216 CTUs, 4 tiles, 4 slices and 3 subpictures. A value of `sps_num_subpics_minus1` is 2, and the position-related syntax elements have the following values:

For subpicture 1

- `subpic_ctu_top_left_x[ 0 ]`, is not signaled but inferred as 0;
- `subpic_ctu_top_left_y[ 0 ]`, is not signaled but inferred as 0;
- `subpic_width_minus1[ 0 ]`, value is 8;
- `subpic_height_minus1[ 0 ]`, value is 11.

For subpicture 2

- `subpic_ctu_top_left_x[ 1 ]`, value is 9;
- `subpic_ctu_top_left_y[ 1 ]`, value is 0;
- `subpic_width_minus1[ 1 ]`, value is 8;
- `subpic_height_minus1[ 1 ]`, value is 5.

For subpicture 3

- `subpic_ctu_top_left_x[ 2 ]`, value is 9;
- `subpic_ctu_top_left_y[ 2 ]`, value is 6;
- `subpic_width_minus1[ 2 ]`, is not signaled but inferred as 8;
- `subpic_height_minus1[ 2 ]`, is not signaled but inferred as 5.

Signalling of tiles

The following table exemplifies the signalling of tile sizes and coordinates of tiles in a picture, which is disclosed in the Picture Parameter Set RBSP syntax table of VVC draft 8.

no_pic_partition_flag	u(1)
if(!no_pic_partition_flag) {	
pps_log2_ctu_size_minus5	u(2)
num_exp_tile_columns_minus1	ue(v)
num_exp_tile_rows_minus1	ue(v)
for(i = 0; i <= num_exp_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i <= num_exp_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
if(NumTilesInPic > 1)	

- 5 The partitioning information (e.g. tile address and tile dimensions) of tile is usually coded in a parameter set. In the above example, a value of an indication no_pic_partition_flag is coded in the bitstream, the value of no_pic_partition_flag is used to indicate whether a picture is partitioned into slices and tiles or not. If no_pic_partition_flag is true (meaning the picture is not partitioned into slices or tiles), it is inferred that the picture is partitioned into only one
- 10 slice and only one tile, whose boundaries are aligned with picture boundaries. Otherwise (no_pic_partition_flag is false), tile partitioning information is coded in the bitstream.

A value of a syntax element tile_column_width_minus1[i] indicates the width of an ith tile columns. A value of a syntax element tile_row_height_minus1[i] indicates the height of the ith tile row.

- 15 The tile row height and tile column widths can be either explicitly signalled in the bitstream or can be inferred according to the bitstream. Syntax elements num_exp_tile_columns_minus1 and num_exp_tile_rows_minus1 indicate the number of tile columns and tile rows respectively, widths and heights of these tiles are explicitly signaled in

bitstream. The widths and heights of other tile columns and rows are inferred according to a function.

The indexing of tiles are represent the “tile scan order in the picture”. The tiles in a picture are ordered (scanned) according to raster scan order, the first tile which is at the top-left corner of the picture is the 0th tile, and the index increases from left to right in each tile row and after the last tile in a tile row is scanned, one continues with the leftmost tile of the next tile row (one below current tile row).

The CTUs in a tile (CTU scan order in a tile) are scanned in raster scan order.

Tile-based scan order of CTUs in a picture indicates that the tile scan order in a picture and CTU scan order in each tile are followed simultaneously. In other words,

- tiles of a picture are scanned according to tile scan order in picture;
- the CTUs of one tile are scanned completely before any of the CTUs in a different tile;
- in each tile, CTUs are scanned in raster scan order (CTU scan order in tile).

15 Signaling of slices

The following table exemplifies signalling of sizes and coordinates of rectangular shaped slices in a picture, which is disclosed in the Picture Parameter Set RBSP syntax table of VVC draft 8.

if(NumTilesInPic > 1)	
rect_slice_flag	u(1)
if(rect_slice_flag)	
single_slice_per_subpic_flag	u(1)
if(rect_slice_flag && !single_slice_per_subpic_flag) {	
num_slices_in_pic_minus1	ue(v)
if(num_slices_in_pic_minus1 > 0)	
tile_idx_delta_present_flag	u(1)
for(i = 0; i < num_slices_in_pic_minus1; i++) {	

if(NumTileColumns > 1)	
slice_width_in_tiles_minus1[i]	ue(v)
if(NumTileRows > 1 && (tile_idx_delta_present_flag tileIdx % NumTileColumns == 0))	
slice_height_in_tiles_minus1[i]	ue(v)
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0 && RowHeight[SliceTopLeftTileIdx[i] / NumTileRows] > 1) {	
num_exp_slices_in_tile[i]	ue(v)
numExpSlicesInTile = num_exp_slices_in_tile[i]	
for(j = 0; j < numExpSlicesInTile; j++)	
exp_slice_height_in_ctus_minus1[j]	ue(v)
i += NumSlicesInTile[i]	
}	
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)	
tile_idx_delta[i]	se(v)
}	
}	

In VVC draft 8, a slice contains one or more complete tiles, or a tile contains one or more complete slices. Therefore the slice coordinates and sizes are indicated with respect to tile partitioning. In VVC draft 8, the tile partitioning is signalled in the picture parameter set. The slice partitioning information is signalled using the tile mapping information.

In the above table, a value of the syntax element num_slices_in_pic_minus1 indicates a number of slices in a picture. A value of syntax element Tile_idx_delta[i] indicates a difference between the tile indexes of first tile of the (i+1)th slice and ith slice. For example, a tile index of the first tile of the first slice in a picture is 0. If the tile index of the first tile of the second slice is 5, a value of Tile_idx_delta[0] is equal to 5. In this context, the tile index may be used as an address of the slice, e.g. an index of the first tile of a slice is the starting address of the slice.

A value of syntax element `slice_width_in_tiles_minus1[i]` indicates the width the i^{th} slice in the picture in number of tiles, and a value of syntax element `slice_height_in_tiles_minus1[i]` indicates the height of the i^{th} slice in the picture, in number of tiles (in an example, as disclosed in VVC draft 8, `slice_width_in_tiles_minus1[i]` plus 1 specifies the width of the i^{th} rectangular slice in units of tile columns. `slice_height_in_tiles_minus1[i]` plus 1 specifies the height of the i^{th} rectangular slice in units of tile rows).

In the above table, if a value of `slice_width_in_tiles_minus1[i]` and a value of `slice_height_in_tiles_minus1[i]` are both equal to 0 (indicating that i^{th} slice has a dimension of at most one tile in height and one tile in width), a value of syntax element `num_exp_slices_in_tile[i]` is coded in the bitstream. The value the syntax element `num_exp_slices_in_tile[i]` indicates a number of slices in a tile.

According to VVC draft 8, a slice may contain multiple complete tiles or a tile may contain multiple complete slices. As shown in the above syntax table, a number of tiles in a slice is indicated by `slice_width_in_tiles_minus1[i]` and `slice_height_in_tiles_minus1[i]`. And if the number of tiles in a slice is equal to 1, a number of slices in the tile is indicated by `num_exp_slices_in_tile[i]`. So if the value of `slice_width_in_tiles_minus1[i]` and the value of `slice_height_in_tiles_minus1[i]` are both equal to 1, the size of the slice might be equal to or smaller than a size of a tile.

The syntax element `single_slice_per_subpic_flag` is true (e.g. a value of `single_slice_per_subpic_flag` is equal to 1) indicates that there is one and only one slice per subpicture for all subpictures of a picture (in this case, a subpicture cannot be divided into more than one slice).

According to one alternative signaling method, the slice starting addresses and slice sizes are indicated in VVC draft 8 is according to following steps:

1. First the tile partitioning information is indicated in the bitstream, where an index (tileIdx) is used to index tiles in a picture (According to the tile scan order in picture). After this step, values of the indices, coordinates and sizes of each tile are obtained.
2. A number of slices in the picture is signalled. In one example, the number of slices
5 can be indicated by a value of syntax element num_slices_in_pic_minus1.
3. For the first slice in the picture, width and height of the slice in number of tiles are indicated. The starting address of the first slice is not explicitly signalled, and is inferred to be tileIdx 0, first tile in the picture is the first tile in the first slice of the picture.
- 10 4. If the size of the first slice is equal to one tile in width and one tile in height, and if there are more than one CTU rows in the tile, a value of syntax element num_exp_slices_in_tile[0] is signalled, the value of syntax element num_exp_slices_in_tile[0] indicates a quantity of slices that are comprised in the said tile. The value of num_exp_slices_in_tile[0] might indicate a number of slices
15 in a tile whose sizes are explicitly indicated in the bitstream, whereas there might be slices in the said tile whose quantity and sizes are inferred based on the size of the tile, and sizes of the slices whose sizes are explicitly signalled. The total number of slices in a tile might therefore be determined according to num_exp_slices_in_tile[0] and the size of the tile. In one example, the number of slices in a tile can be represented by
20 the parameter numSlicesInTile[0].
5. For each of the second slice till the last slice (including second slice but excluding the last slice) in the picture, width and height of the slice in number of tiles is explicitly indicated. The starting address of the slice might be explicitly indicated by a value of syntax element tile_idx_delta[i], where i is the index of the slice. If the starting
25 address is not explicitly signalled (e.g. if slices are signalled in an order that makes it possible to infer the starting position of a next slice, by using the starting position and

width and height of the current slice), then the starting address of the slice is inferred via a function. As an example, if slices are signalled in the bitstream in left to right order in a picture, if the horizontal coordinate of the starting address of a slice and the width of the slice are known, the horizontal coordinate of the starting coordinate of the next slice is inferred as sum of width of the slices plus starting address of the slice.

6. If a size of the n^{th} slice (n is between 2 and value of slices in picture minus 1, both inclusive) is equal to 1 tile in width and 1 tile in height, and if there is more than 1 CTU rows in the tile that is comprised within the n^{th} slice, a value of syntax element `num_exp_slices_in_tile[ n ]` is signalled, the value of syntax element `num_exp_slices_in_tile[ n ]` indicates a quantity of slices that are comprised in the said tile.
7. For the last slice in the picture, the width and height of the slice is not explicitly signalled, but inferred according to the number of tiles in picture width, number of tiles in picture height and the starting address of the last slice. The starting address of the last slice can be explicitly indicated or inferred. The inference of the width and height of the last slice in picture can be performed according to following two equations, which are from the section 6.5.1 of VVC Draft 8.

$$\begin{aligned}\text{slice_width_in_tiles_minus1}[i] &= \text{NumTileColumns} - 1 - \text{tileX} \\ \text{slice_height_in_tiles_minus1}[i] &= \text{NumTileRows} - 1 - \text{tileY}\end{aligned}$$

- 20 `slice_width_in_tiles_minus1[ i ]` plus 1 specifies the width of the i^{th} rectangular slice in units of tile columns.

`slice_height_in_tiles_minus1[ i ]` plus 1 specifies the height of the i^{th} rectangular slice in units of tile rows.

- In the above steps, the width and height of the last slice is not signaled, as it can be inferred if the starting address of the slice is known.

The variables tileX, tileY, NumTileColumns, NumTileRows in the above equation are explained later.

Section 6.5.1 of VVC draft 8

6.5.1 CTB raster scanning, tile scanning, and subpicture scanning processes

- 5 For rectangular slices, the list NumCtusInSlice[i] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, specifying the number of CTU in the i-th slice, the list SliceTopLeftTileIdx[i] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, specifying the index of the top-left tile of the slice, and the matrix CtbAddrInSlice[i][j] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, and j ranging from 0 to
- 10 NumCtusInSlice[i] - 1, inclusive, specifying the picture raster scan address of the j-th CTB within the i-th slice, are derived as follows:

```

    if( single_slice_per_subpic_flag ) {
        for( i = 0; i <= sps_num_subpics_minus1; i++ )
            NumCtusInSlice[ i ] = 0
15    for( i = 0; i < PicSizeInCtbsY; i++ ) {
        sliceIdx = subpic_info_present_flag ? CtbToSubpicIdx[ i ] : 0
        CtbAddrInSlice[ sliceIdx ][ NumCtusInSlice[ sliceIdx ] ] = i
        NumCtusInSlice[ sliceIdx ]++
    }
20    } else {
        tileIdx = 0
        for( i = 0; i <= num_slices_in_pic_minus1; i++ )
            NumCtusInSlice[ i ] = 0
        for( i = 0; i <= num_slices_in_pic_minus1; i++ ) {
25            SliceTopLeftTileIdx[ i ] = tileIdx
            tileX = tileIdx % NumTileColumns
            tileY = tileIdx / NumTileColumns
            if( i == num_slices_in_pic_minus1 ) {
                slice_width_in_tiles_minus1[ i ] = NumTileColumns - 1 - tileX
30                slice_height_in_tiles_minus1[ i ] = NumTileRows - 1 - tileY
                NumSlicesInTile[ i ] = 1
            }
            if( slice_width_in_tiles_minus1[ i ] == 0 &&
                slice_height_in_tiles_minus1[ i ] == 0 ) {
35                ctbY = tileRowBd[ tileY ]
                for( j = 0; j < NumSlicesInTile[ i ] - 1; j++ ) {

```

```

        AddCtbsToSlice( i, tileColBd[ tileX ], tileColBd[ tileX + 1 ],
            ctbY, ctbY + SliceHeightInCtusMinus1[ i ] + 1 )
        ctbY += SliceHeightInCtusMinus1[ i ] + 1
        i++
5      }
      AddCtbsToSlice( i, tileColBd[ tileX ], tileColBd[ tileX + 1 ], ctbY,
        tileRowBd[ tileY + 1 ] )
      } else
        for( j = 0; j <= slice_height_in_tiles_minus1[ i ]; j++ )
10      for( k = 0; k <= slice_width_in_tiles_minus1[ i ]; k++ )
        AddCtbsToSlice( i, tileColBd[ tileX + k ], tileColBd[ tileX + k + 1 ],
            tileRowBd[ tileY + j ], tileRowBd[ tileY + j + 1 ] )
        if( tile_idx_delta_present_flag )
            tileIdx += tile_idx_delta[ i ]
15      else {
        tileIdx += slice_width_in_tiles_minus1[ i ] + 1
        if( tileIdx % NumTileColumns == 0 )
            tileIdx += slice_height_in_tiles_minus1[ i ] * NumTileColumns
        }
20    }
  }
}

```

Where the function AddCtbsToSlice(sliceIdx, startX, stopX, startY, stopY) is specified as follows:

```

    for( ctbY = startY; ctbY < stopY; ctbY++ )
25    for( ctbX = startX; ctbX < stopX; ctbX++ ) {
      CtbAddrInSlice[ sliceIdx ][ NumCtusInSlice[ sliceIdx ] ] =
        ctbY * PicWidthInCtbsY + ctbX      (30)
      NumCtusInSlice[ sliceIdx ]++
    }

```

30 The above description of the signalling of slice starting addresses and slice sizes in a picture is one example disclosed in VVC draft8. Specifically, the description describes the case where rectangular shaped slices are used, a number of slices in each subpicture is not indicated to be equal to 1 (a value of single_slice_per_subpic_flag syntax element is not equal to 1; when a value of single_slice_per_subpic_flag syntax element is equal to 1, each

subpicture of a picture is composed of one and only one slice), more than 1 tile is present in the picture, and number of CTU rows in a tile is greater than 1. If some of the said parameters are changed, other modes of signalling of slice starting addresses and slice sizes can be used. For example, if there is only 1 slice in each subpicture, width and heights of the slices are not explicitly signalled in the bitstream, but inferred to be equal to corresponding subpicture width and height, respectively.

The subclause 6.5.1 of VVC draft 8 also discloses a scan order of CTUs in a slice i , where i is the slice index. The matrix $CtbAddrInSlice[i][n]$, which is the output of this subclause, specifies the CTU scan order in slice i , where n is the CTU index that is from 0 to number of CTUs of slice i . The value of $CtbAddrInSlice[i][n]$ specifies the address of the n^{th} CTU in slice i .

Normally, the CTU scan order in a slice specifies an order in which the CTUs of a slice are scanned. The steps above of Section 6.5.1 of VVC draft 8 exemplifies an example about how the CTU scan order in slice is determined, e.g. how the matrix $CtbAddrInSlice[i][n]$ is constructed.

As disclosed in VVC Draft 8, $CtbAddrInSlice[i][n]$ (which specifies CTU scan order in slice i) also specifies a processing order of CTUs in a slice. Specifically, the CTUs of a picture are processed in the following order:

Slices are indexed by i and processed in increasing i , starting from i which is equal 0 till i which is equal to the number of slices in picture minus 1. Each CTU n in a slice i is processed in increasing order, starting from n which is equal 0 till n which is equal to the number of CTUs in slice i minus 1. The value of $CtbAddrInSlice[i][n]$ specifies the address of the n^{th} CTU in slice i .

As disclosed in VVC Draft 8, a signalling method is used to indicate a number of slices in each subpicture of a picture. A value of `single_slice_per_subpic_flag` is used to indicate whether all subpictures of a picture comprises one and only one slice or not. If a value of

single_slice_per_subpic_flag equal to 1, all subpictures of a picture comprises one and only one slice. Otherwise (a value of single_slice_per_subpic_flag is not equal to 1), a subpicture of a picture might comprises more than 1 slice.

- 5 Figure 9 shows CTU addresses in a picture and one slice (slice 5, the 5th slice in the picture) in the picture as example. According to this example, values of the CtbAddrInSlice is as follows:

CtbAddrInSlice[4][0] = 27

CtbAddrInSlice[4][1] = 28

- 10 CtbAddrInSlice[4][2] = 29

CtbAddrInSlice[4][3] = 30

CtbAddrInSlice[4][4] = 37

CtbAddrInSlice[4][5] = 38

CtbAddrInSlice[4][6] = 39

- 15 CtbAddrInSlice[4][7] = 40.

Scanning order refers to the indexing, according to the order in which the index is incremented, of X in Y, such as CTU scanning order in tile, where X is CTU and Y is tile.

Processing means decoding or encoding in the encoder or decoder. Therefore the processing order means the order in which X (e.g. a CTU) is processed in in the encoder or decoder.

- 20 In the VVC draft 8, when there are more than one tile comprised in a picture, a slice in the picture is signaled as follows:

1. Value of syntax element (explicit indication in a bitstream or inference from a bitstream) is used to determine the starting tile address of a slice, in number of tiles.
 2. For each slice except for the last slice, a quantity of tiles comprised in the slice is
- 25 signaled.

- a. If the slice only comprise 1 tile, a quantity of slices comprised in the tile is indicated.
3. For the last slice in the picture, a quantity of tiles comprised in the slice is inferred if the slice comprises at least 1 complete tile.
- 5 In VVC Draft 8, if a size of the last slice in a picture is greater than or equal to 1 tile (in width and height), then a value of the size of the last slice is inferred from a bitstream and not directly coded in the bitstream.
- As showed in the above, a value of the `slice_width_in_tiles_minus1[ i ]` and a value of `slice_height_in_tiles_minus1[ i ]` (which indicate the width and height of the i-th slice in number of tiles, respectively) are coded in a bitstream if i is smaller than `num_slices_in_pic_minus1` (due to the for-loop “for(i = 0; i < num_slices_in_pic_minus1; i++)”). A value of the width and a value of height of the slice are not signalled when i is equal to `num_slices_in_pic_minus1`, e.g. for the last slice.

15 Embodiment 1

- Determine whether a picture is partitioned into more than one tile.
- When it is determined that the picture is partitioned into more than one tile (e.g. when the `NumTilesInPic > 1` condition is true), obtain a quantity of tiles in a picture width (e.g. the number of tiles in a tile row of a picture), `NumTileColumns`.
- 20 • Obtain a quantity of tile rows in the picture height (e.g. the number of tiles in a tile column of a picture), `NumTileRows`.
- Obtain the top-left coordinate of a slice in tile index (`SliceTopLeftTileIdx[ i ]`). A value of tile index specifies an index value of a tile in the picture, where tiles are ordered (indexed) in raster scan order in a picture.

- Obtain the horizontal coordinate of the tile according to the value of the tile index, indicated by parameter tileX.
 - Obtain the vertical coordinate of the tile according to the value of the tile index, indicated by parameter tileY.
- 5 • For each slice in the picture, a value of an indication for the width of the slice is coded in the bitstream, if a difference value between NumTileColumns and tileX is greater than a first threshold.
- 10 • For each slice in the picture, a value of an indication for the height of the slice is coded in the bitstream, if a difference value between NumTileRows and tileY is greater than a second threshold.

The first and second thresholds might be integer numbers that are greater than 0. In one example both threshold values might be equal to 1. The thresholds indicate a size threshold in number of tiles along horizontal (first threshold) and vertical (second threshold) directions. When the first threshold indicates that the size threshold is equal to 1 tile, the indication for slice width is included in the bitstream if the difference between tileX and number of tiles in picture width is greater than 1 tile. It is noted that depending on the starting value convention of tileX (the value of tileX for the first tile in the picture), the threshold might be different. For example if the starting value of tileX is 0, then first threshold might be 1. If the starting value of tileX is 1, then first threshold might be 0.

- 20 In an embodiment, a value of the width of the last slice and a value of the height of the last slice are also not coded in the bitstream, if the last slice comprises at least 1 complete tile.

In an embodiment, a value of tileX is obtained according to modulo operation and according to a value of NumTileColumns. For example, $\text{tileX} = \text{tileIdx} \% \text{NumTileColumns}$.

- 25 In an embodiment, a value of tileY is obtained according to division operation and according to a value of NumTileRows. For example, $\text{tileY} = \text{tileIdx} / \text{NumTileRows}$.

In an embodiment, the condition “greater than threshold a ” may be implemented as “not equal to a threshold b ”. It is also possible to implement the condition as “greater than or equal to a threshold c ”. Normally, the threshold a is equal to the threshold b , and the third threshold c is equal to threshold $a + 1$.

- 5 One possible implementation of the invention is depicted in table 3. The table 1, which was part of VVC draft 8 is modified.

In table 3, row number 10 and 11 describe how to obtain the tileX and tileY. A value of SliceTopLeftTileIdx[i] specifies a value of the index (tile index) of the top-left tile in the i -th slice. A value of the tile index is the index of a tile in a picture, where tiles are ordered in
 10 the picture in raster scan order. For example showed in Fig. 7, the tile comprises CTU 0 has the tile index 0. The tile comprises CTU 16 has tile index 1, the tile comprises CTU 24 has tile index 2, the tile comprises CTU 44 has tile index 3, the tile comprises CTU 52 has tile index 4, the tile comprises CTU 56 has tile index 5, and so on.

A value of tileX (horizontal coordinate of tile in the picture) and a value of tileY (vertical
 15 coordinate of tile in the picture) are obtained using the equations:

$$\text{tileX} = \text{SliceTopLeftTileIdx}[i] \% \text{NumTileColumns};$$

$$\text{tileY} = \text{SliceTopLeftTileIdx}[i] / \text{NumTileColumns}.$$

In an embodiment, in line 12, the following condition “NumTileColumns – tileX > 1” is added to other conditions. If the condition is not satisfied (NumTileColumns – tileX is not
 20 greater than 1), a value of syntax element slice_width_in_tiles_minus1[i] is not directly coded in a bitstream. Therefore bitrate saving and hence compression efficiency can be achieved.

Similarly, in line 14, the following condition “NumTileRows – tileY > 1” is added to other conditions. If the condition is not satisfied (NumTileRows – tileY is not greater than 1), a
 25 value of syntax element slice_height_in_tiles_minus1[i] is not directly coded in the bitstream. Therefore bitrate saving and hence compression efficiency can be achieved.

In an example, the threshold value is 1, which can be different in different implementations.

Table 3

if(NumTilesInPic > 1)		1
rect_slice_flag	u(1)	2
if(rect_slice_flag)		3
single_slice_per_subpic_flag	u(1)	4
if(rect_slice_flag && !single_slice_per_subpic_flag) {		5
num_slices_in_pic_minus1	ue(v)	6
if(num_slices_in_pic_minus1 > 0)		7
tile_idx_delta_present_flag	u(1)	8
for(i = 0; i < num_slices_in_pic_minus1; i++) {		9
tileX = SliceTopLeftTileIdx[i] % NumTileColumns		10
tileY = SliceTopLeftTileIdx[i] / NumTileRows		11
if(NumTileColumns > 1 && NumTileColumns - tileX > 1)		12
slice_width_in_tiles_minus1[i]	ue(v)	13
if(NumTileRows > 1 && NumTileRows - tileY > 1 && (tile_idx_delta_present_flag tileX == 0))		14
slice_height_in_tiles_minus1[i]	ue(v)	15
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0 && RowHeight[SliceTopLeftTileIdx[i] / NumTileColumns] > 1) {		16
num_exp_slices_in_tile[i]	ue(v)	17
numExpSlicesInTile = num_exp_slices_in_tile[i]		18
for(j = 0; j < numExpSlicesInTile; j++)		19
exp_slice_height_in_ctus_minus1[j]	ue(v)	20
i += NumSlicesInTile[i]		21
}		22
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)		23
tile_idx_delta[i]	se(v)	24
}		25
}		26

In one embodiment, the values of `slice_width_in_tiles_minus1[ i ]` and `slice_height_in_tiles_minus1[ i ]` are inferred when they are not present in the bitstream.

Below, an implementation of the invention in Section 6.5.1 of VVC draft 8 is presented as example.

- 5 According to one example, the values are inferred to be equal to zero. This can for instance be implemented in the Section 6.5.1 of VVC draft 8 as follows:

```

        if(NumTileColumns – tileX =< 1) {
            slice_width_in_tiles_minus1[ i ] = 0
        }
10    if(NumTileRows – tileY =< 1) {
        slice_height_in_tiles_minus1[ i ] = 0
    }

```

- In the above assignments, the values of `slice_width_in_tiles_minus1[ i ]` is set equal to 0 when the difference between `NumTileColumns` and `tileX` is greater than 1, which is the same
- 15 threshold that determines the presense of `slice_width_in_tiles_minus1[ i ]` in the bitstream. In other words, the condition “`NumTileColumns – tileX <= thr`” determines whether or not `slice_width_in_tiles_minus1[ i ]` is present in the bitstream, and when `slice_width_in_tiles_minus1[ i ]` is not present in the bitstream, the value of `slice_width_in_tiles_minus1[ i ]` is inferred (set equal to 0). Similar assignment is done for
- 20 `slice_height_in_tiles_minus1[ i ]` when `slice_height_in_tiles_minus1[ i ]` is not present in the bitstream.

- The conditions “`NumTileRows – tileY =< 1`” and “`NumTileColumns – tileX =< 1`” in Section 6.5.1 of VVC draft 8 below are for checking the presence of `slice_width_in_tiles_minus1[ i ]` and `slice_height_in_tiles_minus1[ i ]` in the bitstream. They
- 25 might be represented in other ways as well, the key point is when `slice_width_in_tiles_minus1` and `slice_height_in_tiles_minus1` are not present in rows 13 and

15 of table 3 (the condition for determining presence in the bitstream evaluates to false), then the values of slice_width_in_tiles_minus1 and slice_height_in_tiles_minus1 should be inferred as 0.

The value of slice_height_in_tiles_minus1[i] is inferred to be zero indicates the height of the slice i is equal to 1 tile.

When the value of slice_width_in_tiles_minus1[i] is inferred to be zero, it indicates the width of the slice i is equal to 1 tile.

slice_height_in_tiles_minus1 and slice_width_in_tiles_minus1 are syntax elements that indicate the size of the slice in each direction. According to embodiments of the invention, if the syntax element related to the width or height of the slice in number of tiles is not present in the bitstream, the width or height of the slice is inferred to be 1 tile.

Implementation of the invention in Section 6.5.1 of VVC draft 8

6.5.1 CTB raster scanning, tile scanning, and subpicture scanning processes

For rectangular slices, the list NumCtusInSlice[i] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, specifying the number of CTU in the i-th slice, the list SliceTopLeftTileIdx[i] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, specifying the index of the top-left tile of the slice, and the matrix CtbAddrInSlice[i][j] for i ranging from 0 to num_slices_in_pic_minus1, inclusive, and j ranging from 0 to NumCtusInSlice[i] - 1, inclusive, specifying the picture raster scan address of the j-th CTB within the i-th slice, are derived as follows:

```

    if( single_slice_per_subpic_flag ) {
        for( i = 0; i <= sps_num_subpics_minus1; i++ )
            NumCtusInSlice[ i ] = 0
        for( i = 0; i < PicSizeInCtbsY; i++ ) {
            sliceIdx = subpic_info_present_flag ? CtbToSubpicIdx[ i ] : 0
            CtbAddrInSlice[ sliceIdx ][ NumCtusInSlice[ sliceIdx ] ] = i
            NumCtusInSlice[ sliceIdx ]++
        }
    } else {
        tileIdx = 0
        for( i = 0; i <= num_slices_in_pic_minus1; i++ )

```

```

    NumCtusInSlice[ i ] = 0
    for( i = 0; i <= num_slices_in_pic_minus1; i++ ) {
        SliceTopLeftTileIdx[ i ] = tileIdx
        tileX = tileIdx % NumTileColumns
5       tileY = tileIdx / NumTileColumns
        if( i == num_slices_in_pic_minus1 ) {
            slice_width_in_tiles_minus1[ i ] = NumTileColumns - 1 - tileX
            slice_height_in_tiles_minus1[ i ] = NumTileRows - 1 - tileY
            NumSlicesInTile[ i ] = 1
10      }

        if( NumTileColumns - tileX <= 1 ) {
            slice_width_in_tiles_minus1[ i ] = 0
        }
        if( NumTileRows - tileY <= 1 ) {
15      slice_height_in_tiles_minus1[ i ] = 0
        }

        if( slice_width_in_tiles_minus1[ i ] == 0 &&
slice_height_in_tiles_minus1[ i ] == 0 ) {
            ctbY = tileRowBd[ tileY ]
20      for( j = 0; j < NumSlicesInTile[ i ] - 1; j++ ) {
                AddCtbsToSlice( i, tileColBd[ tileX ], tileColBd[ tileX + 1 ],
                    ctbY, ctbY + SliceHeightInCtusMinus1[ i ] + 1 )
                ctbY += SliceHeightInCtusMinus1[ i ] + 1
                i++
25      }

            AddCtbsToSlice( i, tileColBd[ tileX ], tileColBd[ tileX + 1 ], ctbY,
tileRowBd[ tileY + 1 ] )
        } else
            for( j = 0; j <= slice_height_in_tiles_minus1[ i ]; j++ )
30      for( k = 0; k <= slice_width_in_tiles_minus1[ i ]; k++ )
                AddCtbsToSlice( i, tileColBd[ tileX + k ], tileColBd[ tileX + k + 1 ],
                    tileRowBd[ tileY + j ], tileRowBd[ tileY + j + 1 ] )
            if( tile_idx_delta_present_flag )
                tileIdx += tile_idx_delta[ i ]
35      else {
                tileIdx += slice_width_in_tiles_minus1[ i ] + 1
                if( tileIdx % NumTileColumns == 0 )
                    tileIdx += slice_height_in_tiles_minus1[ i ] * NumTileColumns

```

```

    }
  }
}

```

Where the function AddCtbsToSlice(sliceIdx, startX, stopX, startY, stopY) is specified as follows:

```

    for( ctbY = startY; ctbY < stopY; ctbY++ )
      for( ctbX = startX; ctbX < stopX; ctbX++ ) {
        CtbAddrInSlice[ sliceIdx ][ NumCtusInSlice[ sliceIdx ] ] =
10      ctbY * PicWidthInCtbsY + ctbX      (30)
        NumCtusInSlice[ sliceIdx ]++
      }

```

As shown in Fig.13, a method of coding implemented by a decoding device is disclosed, the method comprising:

S1301: obtaining a bitstream for a current picture.

15 The bitstream may be obtained according to wireless network or wired network. The bitstream may be transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, microwave, WIFI, Bluetooth, LTE or 5G.

In an embodiment, a bitstream are a sequence of bits, in the form of a network abstraction layer (NAL) unit stream or a byte stream, that forms the representation of a sequence of access units (AUs) forming one or more coded video sequences (CVSs).

In some embodiments, for a decoding process, decoder side reads a bitstream and derives decoded pictures from the bitstream; for an encoding process, encoder side produces a bitstream.

25 Normally, a bitstream will comprise syntax elements that are formed by a syntax structure.

syntax element: An element of data represented in the bitstream.

syntax structure: Zero or more syntax elements present together in the bitstream in a specified order.

In a specific example, bitstream formats specifies the relationship between the network abstraction layer (NAL) unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. The NAL unit stream format comprises a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes.

This clause specifies the relationship between source and decoded pictures that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of pictures in decoding order.

The source and decoded pictures are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome).
- Luma and two chroma (YCbCr or YCgCo).
- Green, blue, and red (GBR, also known as RGB).
- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ).

The variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be

indicated in syntax that is specified in VUI parameters as specified in ITU-T H.266 | ISO/IEC 23002-7.

S1302: obtaining a quantity of tiles in a tile row of the current picture.

- 5 It could be understood that, the quantity of tiles in a tile row is same means as a number of tile columns.

In an example, a quantity of tiles in a tile row of the current picture is represented according to a variable NumTileColumns, specifying the number of tile columns.

- In an example, a quantity of tiles in a tile row of the current picture is obtained according to
 10 parameters coded in the picture parameter set of the bitstream. The quantity of tiles in a tile row of the current picture may be obtained according to directly parse parameter in the bitstream or obtained according to derive a value from parameter.

S1303: obtaining a value of an address of a slice of the current picture.

- 15 In an example, the value of the address of the slice specify an address of a top-left coordinate of the slice in tile index.

In an example, the address of the slice is an index of leftmost tile of the topmost tile row of the slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

- 20 In an example, a value of an address of a slice is represented according to a variable SliceTopLeftTileIdx[i], “i” is a slice index.

SliceTopLeftTileIdx[i] for i ranging from 0 to pps_num_slices_in_pic_minus1, inclusive, specifying the tile index of the tile containing the first CTU in the slice.
 pps_num_slices_in_pic_minus1 plus 1 specifies the number of rectangular slices in each

- 25 picture referring to the picture parameter set, PPS.

In an example, a value of an address of a slice is obtained according to parameter coded in the picture parameter set of the bitstream. The quantity of tiles in a tile row of the current picture may be obtained according to directly parse parameter in the bitstream or obtained according to derive a value from parameter.

5

S1304: obtaining a value of tileX according to the value of the address of the slice.

In an example, the obtaining a value of tileX according to the value of the address of the slice comprises: obtaining the value of the tileX according to the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

10 In an example, the value of tileX is obtained according to a modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

In an example, $\text{tileX} = \text{SliceTopLeftTileIdx}[i] \% \text{NumTileColumns}$.

In an example, the value of tileX indicates a value of a horizontal coordinate.

15 S1305: parsing an indication value for a slice width from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold.

The first threshold is an integer value, the first threshold may be 0, 1, 2...

In an example, the indication value for a slice width is represented as
 20 $\text{pps_slice_width_in_tiles_minus1}[i]$. $\text{pps_slice_width_in_tiles_minus1}[i]$ plus 1 specifies the width of the i-th rectangular slice in units of tile columns. The value of $\text{pps_slice_width_in_tiles_minus1}[i]$ may be in the range of 0 to $\text{NumTileColumns} - 1$, inclusive. When not present, the value of $\text{pps_slice_width_in_tiles_minus1}[i]$ is inferred to be equal to 0.

25 It could be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not

equal to a first threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile row of the current picture and the value of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture is not equal to a first threshold”. Hence, the value of tileX is not calculated at first but is represented according to the result of modulus operation between the value of the address of the slice and the quantity of tiles in the tile row of the current picture.

According to embodiments of the present invention, a signaling scheme is disclosed, only there are at least two tile columns, then syntax element for slice width is signaled. Hence, the bitstream utilization and decoding efficiency have been improved.

10

In an example, wherein the method further comprises: setting the indication value for the slice width to a default value, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is equal to the first threshold. The default value is an integer value, in an example, the default value may be 0, or 1.

15 In an example, wherein the default value indicates that the width of the slice comprises 1 tile column.

In an example, the method further comprises: obtaining a quantity of tiles in a tile column of the current picture; obtaining a value of tileY according to the value of the address of the slice; parsing an indication value for a slice height from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile column of the current picture and the value of the tileY is not equal to a second threshold. The second threshold is an integer value, the second threshold may be 0, 1, 2...

20

It could be understood that, the quantity of tiles in a tile column is same means as a number of tile rows.

In an example, the quantity of tiles in a tile column of the current picture is represented according to a variable NumTileRows. The variable NumTileRows, specifying the number of tile row.

In an example, an indication value for a slice height is represented according to
5 pps_slice_height_in_tiles_minus1[i]. pps_slice_height_in_tiles_minus1[i] plus 1 specifies the height of the i-th rectangular slice in units of tile rows when pps_num_exp_slices_in_tile[i] is equal to 0. The value of pps_slice_height_in_tiles_minus1[i] shall be in the range of 0 to NumTileRows - 1, inclusive.

10 In an example, the method further comprises: setting the indication value for the slice height to a preset value, when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is equal to the second threshold.

In an example, wherein the obtaining a value of tileY according to the value of the address of the slice comprises: obtaining the value of tileY according to the value of the address of the
15 slice and the quantity of tiles in the tile row of the current picture.

In one implementation, wherein the value of tileY is obtained according to an integer division operation between the value of the address of the slice and the quantity of tiles in the tile column of the current picture. E.g. $\text{tileY} = \text{SliceTopLeftTileIdx}[i] / \text{NumTileColumns}$.

In an example, wherein the value of tileY indicates a value of a vertical coordinate. It could
20 be understood that, in some examples, the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of tileY is not equal to a second threshold” is equal to the condition “when a difference value between the quantity of tiles in the tile column of the current picture and the value of the integer division operation between the value of the address of the slice and the quantity of tiles in the tile column of the
25 current picture is not equal to a second threshold”. Hence, the value of tileY is not calculated at first but is represented according to the result of the integer division operation between the

value of the address of the slice and the quantity of tiles in the tile column of the current picture.

In an example, a syntax table for the above embodiments is

for(i = 0; i < pps_num_slices_in_pic_minus1; i++) {	
if(SliceTopLeftTileIdx[i] % NumTileColumns != NumTileColumns - 1)	
pps_slice_width_in_tiles_minus1[i]	ue(v)
if(SliceTopLeftTileIdx[i] / NumTileColumns != NumTileRows - 1 && (pps_tile_idx_delta_present_flag SliceTopLeftTileIdx[i] % NumTileColumns == 0))	
pps_slice_height_in_tiles_minus1[i]	ue(v)

- 5 As shown in Fig.14, a video decoding apparatus 1400 is disclosed, the apparatus 1400 comprise:

obtaining module 1401, which is configured to obtain a bitstream for a current picture; and
obtaining module 1401 is configured to obtain a quantity of tiles in a tile row of the current picture and obtain a value of an address of a slice of the current picture;

- 10 calculating module 1402, which is configured to obtain a value of tileX according to the value of the address of the slice;

parsing module 1403, which is configured to parse an indication value for a slice width from the bitstream for the current picture, when a difference value between the quantity of tiles in the tile row of the current picture and the value of tileX is not equal to a first threshold.

- 15 The details for each module could refer to the embodiments and implementations in the above method embodiments.

Benefits of the invention

Fig. 10 exemplifies a partitioning of a picture into 96 CTUs, 4 tiles and 8 slices.

According to the solution disclosed in VVC draft 8 document, the following syntax elements need to be coded in the bitstream according to table 1.

- num_slices_in_pic_minus1, whose value is 0.
- tile_idx_delta_present_flag, whose value is equal to 1
- 5 • for the first slice (meaning that i index is equal to 0)
 - slice_width_in_tiles_minus1[0], whose value is equal to 0
 - slice_height_in_tiles_minus1[0], whose value is equal to 0
 - num_exp_slices_in_tile[0], whose value is equal to 2
 - exp_slice_height_in_ctus_minus1[0] whose value is equal to 0, which
 - 10 indicates the height of the first slice in number of CTU rows.
- for the second slice (i=1)
 - exp_slice_height_in_ctus_minus1[1] whose value is equal to 1,
 - tile_idx_delta[1], whose value is equal to 1.
- for the third slice (i=2)
 - 15 ○ slice_width_in_tiles_minus1[2], whose value is equal to 0
 - slice_height_in_tiles_minus1[2], whose value is equal to 0
 - num_exp_slices_in_tile[2], whose value is equal to 2
 - exp_slice_height_in_ctus_minus1[2] whose value is equal to 0
- for the forth slice (i=3)
 - 20 ○ exp_slice_height_in_ctus_minus1[3] whose value is equal to 1,
 - tile_idx_delta[3], whose value is equal to 1.
- for the fifth slice (i=4)
 - slice_width_in_tiles_minus1[4], whose value is equal to 0
 - slice_height_in_tiles_minus1[4], whose value is equal to 0
 - 25 ○ num_exp_slices_in_tile[4], whose value is equal to 0
 - tile_idx_delta[4], whose value is equal to 1.

- for the sixth slice (i=5)
 - slice_width_in_tiles_minus1[5], whose value is equal to 0
 - slice_height_in_tiles_minus1[5], whose value is equal to 0
 - num_exp_slices_in_tile[5], whose value is equal to 3
 - 5 ○ exp_slice_height_in_ctus_minus1[5] whose value is equal to 1
 - for the seventh slice (i=6)
 - exp_slice_height_in_ctus_minus1[6] whose value is equal to 0,
 - for the eighth slice (i=7)
 - exp_slice_height_in_ctus_minus1[7] whose value is equal to 2,
- 10 According an embodiment of the invention, as showed in Table 3, the following syntax elements need to be coded in the bitstream, in order to describe the same picture partitioning in Fig. 10.
- num_slices_in_pic_minus1, whose value is 0.
 - tile_idx_delta_present_flag, whose value is equal to 1
 - 15 • for the first slice (meaning that i index is equal to 0)
 - slice_width_in_tiles_minus1[0], whose value is equal to 0
 - slice_height_in_tiles_minus1[0], whose value is equal to 0
 - num_exp_slices_in_tile[0], whose value is equal to 2
 - exp_slice_height_in_ctus_minus1[0] whose value is equal to 0, which
 - 20 indicates the height of the first slice in number of CTU rows.
 - for the second slice (i=1)
 - exp_slice_height_in_ctus_minus1[1] whose value is equal to 1,
 - tile_idx_delta[1], whose value is equal to 1.
 - for the third slice (i=2)
 - 25 ○ slice_height_in_tiles_minus1[2], whose value is equal to 0

- num_exp_slices_in_tile[2], whose value is equal to 2
- exp_slice_height_in_ctus_minus1[2] whose value is equal to 0
- for the forth slice (i=3)
 - exp_slice_height_in_ctus_minus1[3] whose value is equal to 1,
 - 5 ○ tile_idx_delta[3], whose value is equal to 1.
- for the fifth slice (i=4)
 - slice_width_in_tiles_minus1[4], whose value is equal to 0
 - num_exp_slices_in_tile[4], whose value is equal to 0
 - tile_idx_delta[4], whose value is equal to 1.
- 10 • for the sixth slice (i=5)
 - num_exp_slices_in_tile[5], whose value is equal to 3
 - exp_slice_height_in_ctus_minus1[5] whose value is equal to 1
- for the seventh slice (i=6)
 - exp_slice_height_in_ctus_minus1[6] whose value is equal to 0,
- 15 • for the eighth slice (i=7)
 - exp_slice_height_in_ctus_minus1[7] whose value is equal to 2,

In summary the following syntax elements do not need to be coded in the bitstream:

- slice_width_in_tiles_minus1[2], whose value is equal to 0
- slice_height_in_tiles_minus1[4], whose value is equal to 0
- 20 • slice_width_in_tiles_minus1[5], whose value is equal to 0
- slice_height_in_tiles_minus1[5], whose value is equal to 0

The values of those syntax elements are inferred to be equal to zero according to the following conditions that are added to the Section 6.5.1 of VVC draft 8.

```

25         if( NumTileColumns - 1 - tileX <= 1 ) {
           slice_width_in_tiles_minus1[ i ] = 0

```

```

    }

    if( NumTileRows - 1 - tileY <= 1 ) {

        slice_height_in_tiles_minus1[ i ] = 0

    }

```

if(NumTilesInPic > 1)		1
rect_slice_flag	u(1)	2
if(rect_slice_flag)		3
single_slice_per_subpic_flag	u(1)	4
if(rect_slice_flag && !single_slice_per_subpic_flag) {		5
num_slices_in_pic_minus1	ue(v)	6
if(num_slices_in_pic_minus1 > 0)		7
tile_idx_delta_present_flag	u(1)	8
for(i = 0; i < num_slices_in_pic_minus1; i++) {		9
tileX = SliceTopLeftTileIdx[i] % NumTileColumns		10
tileY = SliceTopLeftTileIdx[i] / NumTileColumns		11
if(NumTileColumns > 1 && NumTileColumns - tileX > 1)		12
slice_width_in_tiles_minus1[i]	ue(v)	13
if(NumTileRows > 1 && NumTileRows - tileY > 1 && (tile_idx_delta_present_flag tileX == 0))		14
slice_height_in_tiles_minus1[i]	ue(v)	15
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0 && RowHeight[SliceTopLeftTileIdx[i] / NumTileColumns] > 1) {		16
num_exp_slices_in_tile[i]	ue(v)	17
numExpSlicesInTile = num_exp_slices_in_tile[i]		18
for(j = 0; j < numExpSlicesInTile; j++)		19
exp_slice_height_in_ctus_minus1[j]	ue(v)	20
i += NumSlicesInTile[i]		21
}		22
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)		23
tile_idx_delta[i]	se(v)	24

}		25
}		26

Following is an explanation of the applications of the encoding method as well as the decoding method as shown in the above-mentioned embodiments, and a system using them.

1. A method of coding implemented by a decoding device, the method comprising:

obtaining a bitstream for a current picture (or frame);

5 obtaining a quantity of tiles in a tile row of the current picture (e.g. a value of NumTileColumns);

obtaining a value of an address of a current slice (e.g. SliceTopLeftTileIdx[i]), the current slice is comprised in the current picture;

obtaining a value of a horizontal coordinate (in an example, a horizontal coordinate means a
10 coordinate located in a horizontal coordinate axis) (e.g. tileX) according to the value of the address of the current slice (e.g. SliceTopLeftTileIdx[i]);

parsing an indication value for a slice width (i.e. width of a slice) from the bitstream for the current picture (i.e. the indication value for the slice width is coded in the bitstream), when a difference value between the quantity of tiles in the tile row of the current picture (e.g. a
15 value of NumTileColumns) and the value of the horizontal coordinate (e.g. tileX) is greater than a first threshold (e.g. the first threshold is an integer number, may be 0, 1, 2... and so on).

Example 2. The method of example 1, wherein the method further comprises:

setting the indication value for the slice width to a default value (e.g. the default value may be equal to 0), when a difference value between the quantity of tiles in the tile row of the current
20 picture (e.g. a value of NumTileColumns) and the value of the horizontal coordinate (e.g. tileX) is smaller than or equal to the first threshold.

Example 3. The method of example 2, wherein the default value indicates that the width of the current slice comprises 1 tile columns.

Example 4. The method of any one of examples 1 to 3, wherein the obtaining a value of a horizontal coordinate according to the value of the address of the current slice comprises:

obtaining the value of the horizontal coordinate according to the value of the address of the current slice and the quantity of tiles in the tile row of the current picture.

- 5 Example 5. The method of any one of examples 1 to 4, wherein the address of the current slice is an index of leftmost tile of the topmost tile row of the current slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

Example 6. The method of example 4 or 5, wherein the value of the horizontal coordinate is obtained according to a modulus operation between the value of the address of the current

- 10 slice and the quantity of tiles in the tile row of the current picture (e.g. $\text{SliceTopLeftTileIdx}[i] \% \text{NumTileColumns}$).

Example 7. The method of any one of examples 1 to 6, wherein the value of the address of the current slice (e.g. $\text{SliceTopLeftTileIdx}[i]$) specify an address of a top-left coordinate of the current slice in tile index.

- 15 Example 8. The method of any one of examples 1 to 7, wherein the method further comprises:

obtaining a quantity of tiles in a tile column of the current picture (e.g. a value of NumTileRows);

- 20 obtaining a value of a vertical coordinate (in an example, an vertical coordinate means a coordinate located in an vertical coordinate axis) (e.g. tileY) according to the value of the address of the current slice (e.g. $\text{SliceTopLeftTileIdx}[i]$);

- parsing an indication value for a slice height (i.e. height of a slice) from the bitstream for the current picture (i.e. the indication value for the slice height is coded in the bitstream), when a difference value between the quantity of tiles in the tile column of the current picture (e.g. a value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is greater than a second threshold (e.g. the second threshold is an integer number, may be 0, 1, 2... and so on).
- 25

Example 9. The method of example 8, wherein the method further comprises:

setting the indication value for the slice height to a preset value (e.g. the preset value may be equal to 0), when a difference value between the quantity of tiles in the tile column of the current picture (e.g. a value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is smaller than or equal to the second threshold.

Example 10. The method of example 8 or 9, wherein the obtaining a value of a vertical coordinate according to the value of the address of the current slice comprises:

obtaining the value of the vertical coordinate according to the value of the address of the current slice and the quantity of tiles in the tile column of the current picture.

Example 11. The method of example 10, wherein the value of the vertical coordinate is obtained according to an integer division operation between the value of the address of the current slice and the quantity of tiles in the tile column of the current picture (e.g. $\text{SliceTopLeftTileIdx}[i] / \text{NumTileRows}$).

Example 12. A method of coding implemented by a decoding device, the method comprising:

obtaining a bitstream for a current picture (or frame);

obtaining a quantity of tiles in a tile column of the current picture (e.g. a value of NumTileRows);

obtaining a value of an address of a current slice (e.g. $\text{SliceTopLeftTileIdx}[i]$), the current slice is comprised in the current picture;

obtaining a value of a vertical coordinate (in an example, an vertical coordinate means a coordinate located in an vertical coordinate axis) (e.g. tileY) according to the value of the address of the current slice (e.g. $\text{SliceTopLeftTileIdx}[i]$);

parsing an indication value for a slice height (i.e. height of a slice) from the bitstream for the current picture (i.e. the indication value for the slice height is coded in the bitstream), when a difference value between the quantity of tiles in the tile column of the current picture (e.g. a

value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is greater than a second threshold (e.g. the second threshold is an integer number, may be 0, 1, 2... and so on).

Example 13. The method of example 12, wherein the method further comprises:

5 setting the indication value for the slice height to a preset value (e.g. the preset value may be equal to 0), when a difference value between the quantity of tiles in the tile column of the current picture (e.g. a value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is smaller than or equal to the second threshold.

Example 14. The method of example 13, wherein

the default value indicates that the height of the current slice comprises 1 tile rows.

10 Example 15. The method of any one of examples 12 to 14, wherein the obtaining a value of a vertical coordinate according to the value of the address of the current slice comprises:

obtaining the value of the vertical coordinate according to the value of the address of the current slice and the quantity of tiles in the tile column of the current picture.

Example 16. The method of any one of examples 12 to 15, wherein the address of the current
15 slice is an index of leftmost tile of the topmost tile row of the current slice, wherein the tiles comprised in the current picture are indexed according to a raster scan order.

Example 17. The method of example 15 or 16, wherein the value of the vertical coordinate is obtained according to an integer division operation between the value of the address of the current slice and the quantity of tiles in the tile column of the current picture (e.g.
20 $\text{SliceTopLeftTileIdx}[i] / \text{NumTileRows}$).

Example 18. The method of any one of examples 12 to 17, wherein the value of the address of the current slice (e.g. $\text{SliceTopLeftTileIdx}[i]$) specify an address of a top-left coordinate of the current slice in tile index.

Example 19. The method of any one of examples 12 to 18, wherein the method further
25 comprises:

obtaining a quantity of tiles in a tile row of the current picture (e.g. a value of NumTileColumns);

obtaining a value of a horizontal coordinate (in an example, a horizontal coordinate means a coordinate located in a horizontal coordinate axis) (e.g. tileX) according to the value of the

5 address of the current slice (e.g. SliceTopLeftTileIdx[i]);

parsing an indication value for a slice width (i.e. width of a slice) from the bitstream for the current picture (i.e. the indication value for the slice width is coded in the bitstream), when a difference value between the quantity of tiles in the tile row of the current picture (e.g. a value of NumTileColumns) and the value of the horizontal coordinate (e.g. tileX) is greater

10 than a first threshold (e.g. the first threshold is an integer number, may be 0, 1, 2... and so on).

Example 20. The method of example 19, wherein the method further comprises:

setting the indication value for the slice width to a default value (e.g. the default value may be equal to 0), when a difference value between the quantity of tiles in the tile row of the current picture (e.g. a value of NumTileColumns) and the value of the horizontal coordinate (e.g.

15 tileX) is smaller than or equal to the first threshold.

Example 21. The method of example 19 or 20, wherein the obtaining a value of a horizontal coordinate according to the value of the address of the current slice comprises:

obtaining the value of the horizontal coordinate according to the value of the address of the current slice and the quantity of tiles in the tile row of the current picture.

20 Example 22. The method of example 21, wherein the value of the horizontal coordinate is obtained according to a modulus operation between the value of the address of the current slice and the quantity of tiles in the tile row of the current picture (e.g. SliceTopLeftTileIdx[i]% NumTileColumns).

Example 23. A method of coding implemented by an encoding device, the method
25 comprising:

obtaining a quantity of tiles in a tile row of the current picture (e.g. a value of NumTileColumns);

obtaining a value of an address of a current slice (e.g. SliceTopLeftTileIdx[i]), the current slice is comprised in the current picture;

- 5 obtaining a value of a horizontal coordinate (in an example, a horizontal coordinate means a coordinate located in a horizontal coordinate axis) (e.g. tileX) according to the value of the address of the current slice (e.g. SliceTopLeftTileIdx[i]);

coding an indication value for a slice width (i.e. width of a slice) for the current picture (i.e. the indication value for the slice width is coded in the bitstream) to obtain a bitstream, when a

- 10 difference value between the quantity of tiles in the tile row of the current picture (e.g. a value of NumTileColumns) and the value of the horizontal coordinate (e.g. tileX) is greater than a first threshold (e.g. the first threshold is an integer number, may be 0, 1, 2... and so on).

Example 24. The method of example 23, wherein the method further comprises:

obtaining a quantity of tiles in a tile column of the current picture (e.g. a value of

- 15 NumTileRows);

obtaining a value of a vertical coordinate (in an example, an vertical coordinate means a coordinate located in an vertical coordinate axis) (e.g. tileY) according to the value of the address of the current slice (e.g. SliceTopLeftTileIdx[i]);

coding an indication value for a slice height (i.e. height of a slice) for the current picture (i.e.

- 20 the indication value for the slice height is coded in the bitstream) to obtain a bitstream, when a difference value between the quantity of tiles in the tile column of the current picture (e.g. a value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is greater than a second threshold (e.g. the second threshold is an integer number, may be 0, 1, 2... and so on).

Example 25. A method of coding implemented by an encoding device, the method

- 25 comprising:

obtaining a quantity of tiles in a tile column of the current picture (e.g. a value of NumTileRows);

obtaining a value of an address of a current slice (e.g. SliceTopLeftTileIdx[i]), the current slice is comprised in the current picture;

- 5 obtaining a value of a vertical coordinate (in an example, an vertical coordinate means a coordinate located in an vertical coordinate axis) (e.g. tileY) according to the value of the address of the current slice (e.g. SliceTopLeftTileIdx[i]);

coding an indication value for a slice height (i.e. height of a slice) for the current picture (i.e. the indication value for the slice height is coded in the bitstream) to obtain a bitstream, when

- 10 a difference value between the quantity of tiles in the tile column of the current picture (e.g. a value of NumTileRows) and the value of the vertical coordinate (e.g. tileY) is greater than a second threshold (e.g. the second threshold is an integer number, may be 0, 1, 2... and so on).

Example 26. The method of example 25, wherein the method further comprises:

obtaining a quantity of tiles in a tile row of the current picture (e.g. a value of

- 15 NumTileColumns);

obtaining a value of a horizontal coordinate (in an example, a horizontal coordinate means a coordinate located in a horizontal coordinate axis) (e.g. tileX) according to the value of the address of the current slice (e.g. SliceTopLeftTileIdx[i]);

coding an indication value for a slice width (i.e. width of a slice) for the current picture (i.e.

- 20 the indication value for the slice width is coded in the bitstream) to obtain a bitstream, when a difference value between the quantity of tiles in the tile row of the current picture (e.g. a value of NumTileColumns) and the value of the horizontal coordinate (e.g. tileX) is greater than a first threshold (e.g. the first threshold is an integer number, may be 0, 1, 2... and so on).

Example 27. A decoder (30) comprising processing circuitry for carrying out the method

- 25 according to any one of examples 1 to 22.

Example 28. An encoder comprising processing circuitry for carrying out the method according to any one of examples 23 to 26.

Example 29. A computer program product comprising program code for performing the method according to any one of the preceding examples when executed on a computer or a
5 processor.

Example 30. A decoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by
10 the processors, configures the decoder to carry out the method according to any one of the preceding examples.

Example 31. An encoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by
15 the processors, configures the decoder to carry out the method according to any one of the preceding examples.

Example 32. A non-transitory computer-readable medium carrying a program code which, when executed by a computer device, causes the computer device to perform the method of
20 any one of the preceding examples.

FIG. 8 is a block diagram showing a content supply system 3100 for realizing content distribution service. This content supply system 3100 includes capture device 3102, terminal device 3106, and optionally includes display 3126. The capture device 3102 communicates with the terminal device 3106 over communication link 3104. The communication link may
25 include the communication channel 13 described above. The communication link 3104

includes but not limited to WIFI, Ethernet, Cable, wireless (3G/4G/5G), USB, or any kind of combination thereof, or the like.

The capture device 3102 generates data, and may encode the data by the encoding method as shown in the above embodiments. Alternatively, the capture device 3102 may distribute the data to a streaming server (not shown in the Figures), and the server encodes the data and transmits the encoded data to the terminal device 3106. The capture device 3102 includes but not limited to camera, smart phone or Pad, computer or laptop, video conference system, PDA, vehicle mounted device, or a combination of any of them, or the like. For example, the capture device 3102 may include the source device 12 as described above. When the data includes video, the video encoder 20 included in the capture device 3102 may actually perform video encoding processing. When the data includes audio (i.e., voice), an audio encoder included in the capture device 3102 may actually perform audio encoding processing. For some practical scenarios, the capture device 3102 distributes the encoded video and audio data by multiplexing them together. For other practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. Capture device 3102 distributes the encoded audio data and the encoded video data to the terminal device 3106 separately.

In the content supply system 3100, the terminal device 310 receives and reproduces the encoded data. The terminal device 3106 could be a device with data receiving and recovering capability, such as smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114, set top box (STB) 3116, video conference system 3118, video surveillance system 3120, personal digital assistant (PDA) 3122, vehicle mounted device 3124, or a combination of any of them, or the like capable of decoding the above-mentioned encoded data. For example, the terminal device 3106 may include the destination device 14 as described above. When the encoded data includes video, the video decoder 30 included in the terminal device is prioritized to perform video decoding.

When the encoded data includes audio, an audio decoder included in the terminal device is prioritized to perform audio decoding processing.

For a terminal device with its display, for example, smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114,
5 personal digital assistant (PDA) 3122, or vehicle mounted device 3124, the terminal device can feed the decoded data to its display. For a terminal device equipped with no display, for example, STB 3116, video conference system 3118, or video surveillance system 3120, an external display 3126 is contacted therein to receive and show the decoded data.

When each device in this system performs encoding or decoding, the picture encoding device
10 or the picture decoding device, as shown in the above-mentioned embodiments, can be used.

FIG. 9 is a diagram showing a structure of an example of the terminal device 3106. After the terminal device 3106 receives stream from the capture device 3102, the protocol proceeding unit 3202 analyzes the transmission protocol of the stream. The protocol includes but not limited to Real Time Streaming Protocol (RTSP), Hyper Text Transfer Protocol (HTTP),
15 HTTP Live streaming protocol (HLS), MPEG-DASH, Real-time Transport protocol (RTP), Real Time Messaging Protocol (RTMP), or any kind of combination thereof, or the like.

After the protocol proceeding unit 3202 processes the stream, stream file is generated. The file is outputted to a demultiplexing unit 3204. The demultiplexing unit 3204 can separate the multiplexed data into the encoded audio data and the encoded video data. As described above,
20 for some practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. In this situation, the encoded data is transmitted to video decoder 3206 and audio decoder 3208 without through the demultiplexing unit 3204.

Via the demultiplexing processing, video elementary stream (ES), audio ES, and optionally
25 subtitle are generated. The video decoder 3206, which includes the video decoder 30 as explained in the above mentioned embodiments, decodes the video ES by the decoding

method as shown in the above-mentioned embodiments to generate video frame, and feeds this data to the synchronous unit 3212. The audio decoder 3208, decodes the audio ES to generate audio frame, and feeds this data to the synchronous unit 3212. Alternatively, the video frame may store in a buffer (not shown in FIG. 9) before feeding it to the synchronous unit 3212. Similarly, the audio frame may store in a buffer (not shown in FIG. 9) before feeding it to the synchronous unit 3212.

The synchronous unit 3212 synchronizes the video frame and the audio frame, and supplies the video/audio to a video/audio display 3214. For example, the synchronous unit 3212 synchronizes the presentation of the video and audio information. Information may code in the syntax using time stamps concerning the presentation of coded audio and visual data and time stamps concerning the delivery of the data stream itself.

If subtitle is included in the stream, the subtitle decoder 3210 decodes the subtitle, and synchronizes it with the video frame and the audio frame, and supplies the video/audio/subtitle to a video/audio/subtitle display 3216.

The present invention is not limited to the above-mentioned system, and either the picture encoding device or the picture decoding device in the above-mentioned embodiments can be incorporated into other system, for example, a car system.

Mathematical Operators

The mathematical operators used in this application are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, $7 / 4$ and $-7 / -4$ are truncated to 1 and $-7 / 4$ and $7 / -4$ are truncated to -1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^y f(i)$	The summation of $f(i)$ with i taking all integer values from x up to and including y.
$x \% y$	Modulus. Remainder of x divided by y, defined only for integers x and y with $x \geq 0$ and $y > 0$.

Logical operators

The following logical operators are defined as follows:

	$x \&\& y$	Boolean logical "and" of x and y
	$x y$	Boolean logical "or" of x and y
5	!	Boolean logical "not"
	$x ? y : z$	If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

Relational operators

The following relational operators are defined as follows:

10	>	Greater than
	>=	Greater than or equal to
	<	Less than
	<=	Less than or equal to
	= =	Equal to
15	!=	Not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

Bit-wise operators

5 The following bit-wise operators are defined as follows:

- & Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- 10 | Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- 15 ^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of x prior to the shift operation.
- 20 x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.
- 25

Assignment operators

The following arithmetic operators are defined as follows:

- = Assignment operator
- 30 ++ Increment, i.e., $x++$ is equivalent to $x = x + 1$; when used in an array index, evaluates to the value of the variable prior to the increment operation.
- Decrement, i.e., $x--$ is equivalent to $x = x - 1$; when used in an array index, evaluates to the value of the variable prior to the decrement operation.
- += Increment by amount specified, i.e., $x += 3$ is equivalent to $x = x + 3$, and $x += (-3)$ is equivalent to $x = x + (-3)$.
- 35 -= Decrement by amount specified, i.e., $x -= 3$ is equivalent to $x = x - 3$, and $x -= (-3)$ is equivalent to $x = x - (-3)$.

Range notation

The following notation is used to specify a range of values:

$x = y..z$ x takes on integer values starting from y to z , inclusive, with x , y , and z being integer numbers and z being greater than y .

5 Mathematical functions

The following mathematical functions are defined:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases}$$

10 $\text{Asin}(x)$ the trigonometric inverse sine function, operating on an argument x that is in the range of -1.0 to 1.0 , inclusive, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians

$\text{Atan}(x)$ the trigonometric inverse tangent function, operating on an argument x , with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians

$$\text{Atan2}(y, x) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \quad x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \quad x < 0 \ \&\& \ y \geq 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \quad x < 0 \ \&\& \ y < 0 \\ +\frac{\pi}{2} & ; \quad x = 0 \ \&\& \ y \geq 0 \\ -\frac{\pi}{2} & ; \quad \text{otherwise} \end{cases}$$

$\text{Ceil}(x)$ the smallest integer greater than or equal to x .

15 $\text{Clip1}_Y(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_Y) - 1, x)$

$\text{Clip1}_C(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_C) - 1, x)$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases}$$

$\text{Cos}(x)$ the trigonometric cosine function operating on an argument x in units of radians.

$\text{Floor}(x)$ the largest integer less than or equal to x .

20 $\text{GetCurrMsb}(a, b, c, d) = \begin{cases} c + d & ; \quad b - a \geq d/2 \\ c - d & ; \quad a - b > d/2 \\ c & ; \quad \text{otherwise} \end{cases}$

$\text{Ln}(x)$ the natural logarithm of x (the base- e logarithm, where e is the natural logarithm base constant 2.718 281 828...).

$\text{Log}_2(x)$ the base-2 logarithm of x .

$\text{Log}_{10}(x)$ the base-10 logarithm of x .

$$5 \quad \text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases}$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases}$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5)$$

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x = 0 \\ -1 & ; \quad x < 0 \end{cases}$$

$\text{Sin}(x)$ the trigonometric sine function operating on an argument x in units of radians

$$10 \quad \text{Sqrt}(x) = \sqrt{x}$$

$$\text{Swap}(x, y) = (y, x)$$

$\text{Tan}(x)$ the trigonometric tangent function operating on an argument x in units of radians

Order of operation precedence

When an order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- Operations of a higher precedence are evaluated before any operation of a lower precedence.
- Operations of the same precedence are evaluated sequentially from left to right.

The table below specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table: Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x--"
"!x", "-x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", " $\frac{x}{y}$ ", "x % y"
"x + y", "x - y" (as a two-argument operator), " $\sum_{i=x}^y f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x == y", "x != y"
"x & y"
"x y"
"x && y"
"x y"
"x ? y : z"
"x.y"
"x = y", "x += y", "x -= y"

Text description of logical operations

- In the text, a statement of logical operations as would be described mathematically in the following form:

```

5      if( condition 0 )
          statement 0
      else if( condition 1 )
          statement 1
10      ...
      else /* informative remark on remaining condition */
          statement n

```

may be described in the following manner:

... as follows / ... the following applies:

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1

5

– ...

- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

10

In the text, a statement of logical operations as would be described mathematically in the following form:

15

```
if( condition 0a  &&  condition 0b )
    statement 0
else if( condition 1a  ||  condition 1b )
    statement 1
```

...

20

```
else
    statement n
```

may be described in the following manner:

... as follows / ... the following applies:

- If all of the following conditions are true, statement 0:
 - condition 0a
 - condition 0b
- Otherwise, if one or more of the following conditions are true, statement 1:
 - condition 1a
 - condition 1b
- ...
- Otherwise, statement n

25

30

In the text, a statement of logical operations as would be described mathematically in the following form:

```

    if( condition 0 )
        statement 0
    if( condition 1 )
        statement 1

```

5 may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1.

Although embodiments of the invention have been primarily described based on video coding, it should be noted that embodiments of the coding system 10, encoder 20 and decoder 30

10 (and correspondingly the system 10) and the other embodiments described herein may also be configured for still picture processing or coding, i.e. the processing or coding of an individual picture independent of any preceding or consecutive picture as in video coding. In general only inter-prediction units 244 (encoder) and 344 (decoder) may not be available in case the picture processing coding is limited to a single picture 17. All other functionalities (also

15 referred to as tools or technologies) of the video encoder 20 and video decoder 30 may equally be used for still picture processing, e.g. residual calculation 204/304, transform 206, quantization 208, inverse quantization 210/310, (inverse) transform 212/312, partitioning 262/362, intra-prediction 254/354, and/or loop filtering 220, 320, and entropy coding 270 and entropy decoding 304.

20 Embodiments, e.g. of the encoder 20 and the decoder 30, and functions described herein, e.g. with reference to the encoder 20 and the decoder 30, may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on a computer-readable medium or transmitted over communication media as one or more instructions or code and executed by a hardware-based processing unit.

25 Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g.,

according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to
5 retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

By way of example, and not limiting, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or
10 other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless
15 technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible
20 storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

Instructions may be executed by one or more processors, such as one or more digital signal
25 processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete

logic circuitry. Accordingly, the term “processor,” as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and
5 decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to
10 emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

15