CODING TREE UNIT SIZE SIGNALING

[0001] This application claims priority from U.S. Provisional Patent Application No. 62/905,339, filed September 24, 2019, and U.S. Patent Application No. 17/024,246, filed on September 17, 2020.

FIELD

[0002] This disclosure relates generally to field of data processing, and more particularly to video encoding and decoding.

BACKGROUND

[0003] ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) published the H.265/HEVC (High Efficiency Video Coding) standard in 2013 (version 1), 2014 (version 2), 2015 (version 3), and 2016 (version 4). In 2015, these two standard organizations jointly formed the JVET (Joint Video Exploration Team) to explore the potential of developing the next video coding standard beyond HEVC. In October 2017, they issued the Joint Call for Proposals on Video Compression with Capability beyond HEVC (CfP). By February 15, 2018, total 22 CfP responses on standard dynamic range (SDR), 12 CfP responses on high dynamic range (HDR), and 12 CfP responses on 360 video categories were submitted, respectively. In April 2018, all received CfP responses were evaluated in the 122 MPEG / 10th JVET meeting. As a result of this meeting, JVET formally launched the standardization process of next-generation video coding beyond HEVC. The new standard was named Versatile Video Coding (VVC), and JVET was renamed as Joint Video Expert Team. The current version of VTM (VVC Test Model), i.e., VTM 6.

SUMMARY

[0004] Embodiments relate to a method, system, and computer readable medium for coding video data. According to one aspect, a method for coding video data is provided. The method may include receiving video data having a coding tree unit size. The coding tree unit size associated with the video data is signaled by setting two or more flags. The video data is encoded/decoded based on the flags corresponding to the signaled coding tree unit size.

[0011] FIG. 4 is an operational flowchart illustrating the steps carried out by a program that codes video data, according to at least one embodiment;

[0012] FIG. 5 is a block diagram of internal and external components of computers and servers depicted in FIG. 1 according to at least one embodiment;

[0013] FIG. 6 is a block diagram of an illustrative cloud computing environment including the computer system depicted in FIG. 1, according to at least one embodiment; and

[0014] FIG. 7 is a block diagram of functional layers of the illustrative cloud computing environment of FIG. 6, according to at least one embodiment.

DETAILED DESCRIPTION

[0015] Detailed embodiments of the claimed structures and methods are disclosed herein; however, it can be understood that the disclosed embodiments are merely illustrative of the claimed structures and methods that may be embodied in various forms. Those structures and methods may, however, be embodied in many different forms and should not be construed as limited to the exemplary embodiments set forth herein. Rather, these exemplary embodiments are provided so that this disclosure will be thorough and complete and will fully convey the scope to those skilled in the art. In the description, details of well-known features and techniques may be omitted to avoid unnecessarily obscuring the presented embodiments.

[0016] Embodiments relate generally to the field of data processing, and more particularly to video encoding and decoding. The following described exemplary embodiments provide a system, method and computer program to, among other things, code video data using separate flags to replace the coding tree unit size syntax. Therefore, some embodiments have the capacity to improve the field of computing by allowing for less memory use by saving bits through signaling a coding tree unit size through flags.

[0017] As previously described, ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) published the H.265/HEVC (High Efficiency Video Coding) standard in 2013 (version 1), 2014 (version 2), 2015 (version 3), and 2016 (version 4). In 2015, these two standard organizations jointly formed the JVET (Joint Video Exploration Team) to explore the potential of developing the next video coding standard beyond HEVC. In October 2017, they issued the Joint

another set .of devices of system 100.

[0027] Referring now to FIG 2, an exemplary QTBT block structure 200 is depicted. The QTBT block structure 200 may include block partitioning by using QTBT. An corresponding tree representation may also be depicted. The solid lines may indicate quadtree splitting and the dotted lines may indicate binary tree splitting. In each splitting (i.e., non-leaf) node of the binary tree, one flag may be signalled to indicate which splitting type (i.e., horizontal or vertical) may be used, where 0 may indicate horizontal splitting and 1 may indicate vertical splitting. For the quadtree splitting, there may be no need to indicate the splitting type since quadtree splitting always splits a block both horizontally and vertically to produce 4 sub-blocks with an equal size.

[0028] QTBT may remove the concepts of multiple partition types. For example, QTBT may remove the separation of the coding unit, prediction unit, and transform unit concepts, and supports more flexibility for coding unit partition shapes. In the QTBT block structure 200, a coding unit can have either a square or rectangular shape. A coding tree unit (CTU) may be partitioned by a quadtree structure. The quadtree leaf nodes may be further partitioned by a binary tree structure. There may be two splitting types in the binary tree splitting: symmetric horizontal splitting and symmetric vertical splitting.

[0029] The binary tree leaf nodes may be called coding units (CUs), and that segmentation may be used for prediction and transform processing without any further partitioning. This means that the coding unit, prediction unit, and transform unit may have the same block size in the QTBT coding block structure 200. A coding unit may include coding blocks (CBs) of different colour components (e.g. one coding unit may contain one luma CB and two chroma CBs in the case of P and B slices of the 4:2:0 chroma format) or may include a CB of a single component (e.g., one coding unit may contain either one luma CB or two chroma CBs in the case of I slices).

[0030] The following parameters may be defined for the QTBT partitioning scheme:
coding tree unit size may be the root node size of a quadtree, the same concept as in HEVC;
MinQTSize may be the minimum allowed quadtree leaf node size;
MaxBTSize may be the maximum allowed binary tree root node size;
MaxBTDepth may be the maximum allowed binary tree depth; and
MinBTSize may be the minimum allowed binary tree leaf node size

[0039] **log2_min_luma_coding_block_size_minus2** plus 2 may specify the minimum luma coding block size.

[0040] The variables CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, MinTbLog2SizeY, MaxTbLog2SizeY, MinTbSizeY, MaxTbSizeY, PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC may be derived as:

```
if ( use_smallest_ctu_size_flag )
        CtbLog2SizeY = 0
elseif ( use_largest_ctu_size_flag )
        CtbLog2SizeY = 2
else
        CtbLog2SizeY = 1
CtbSizeY = 1 << CtbLog2SizeY
```

[0041] Referring now to FIG. 3C, according to one or more embodiments, sps_max_luma_transform_size_64_flag may be signaled only if coding tree unit size may be greater than or equal to 64×64. In one embodiment, if use_32_ctu_size_flag may be signaled first and may be equal to 1, sps_max_luma_transform_size_64_flag may not be signaled. In one embodiment, if use_64_ctu_size_flag may be signaled first and may be equal to 0, use_32_ctu_size_flag may be then signaled and may be equal to 1, and sps_max_luma_transform_size_64_flag may not be signaled. In one embodiment, if use_128_ctu_size_flag may be signaled first and may be equal to 0, use_32_ctu_size_flag may be then signaled and may be equal to 1, and sps_max_luma_transform_size_64_flag may not be signaled.

[0042] **use_32_ctu_size_flag** equal to 1 may specify that the luma coding tree block size of each coding tree unit may be equal to 32x32. use_32_ctu_size_flag equal to 0 may specify that use_128_ctu_size_flag may be present.

use_128_ctu_size_flag equal to 1 may specify that the luma coding tree block size of each coding tree unit may be equal to 128x128. use_128_ctu_size_flag equal to 0 may specify that the

[0049] **log2_min_luma_coding_block_size_minus2** plus 2 may specify the minimum luma coding block size.

[0050] **sps_max_luma_transform_size_64_flag** equal to 1 may specify that the maximum transform size in luma samples may be equal to 64. sps_max_luma_transform_size_64_flag equal to 0 may specify that the maximum transform size in luma samples may be equal to 32. When not present, the value of sps_max_luma_transform_size_64_flag may be inferred to be equal to 0.

[0051] The variables CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, MinTbLog2SizeY, MaxTbLog2SizeY, MinTbSizeY, MaxTbSizeY, PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC may be derived as:

```
if ( use_smallest_ctu_size_flag )
        CtbLog2SizeY = 0
elseif ( use_largest_ctu_size_flag )
        CtbLog2SizeY = 2
else
        CtbLog2SizeY = 1
CtbSizeY = 1 << CtbLog2SizeY
```

[0052] Referring now to FIG. 4, an operational flowchart illustrating the steps of a method 400 for coding video data is depicted. In some implementations, one or more process blocks of FIG. 4 may be performed by the computer 102 (FIG. 1) and the server computer 114 (FIG. 1). In some implementations, one or more process blocks of FIG. 4 may be performed by another device or a group of devices separate from or including the computer 102 and the server computer 114.

[0053] At 402, the method 400 includes receiving video data having a coding tree unit size.

[0054] At 404, the method 400 includes signaling the coding tree unit size associated with the video data by setting two or more flags.

[0055] At 406, the method 400 includes coding the video data is coded based on the flags

CLAIMS

What is claimed is:

1. A method of video coding, executable by a processor, the method comprising:

receiving (402) video data having a coding tree unit size;

signaling (404) the coding tree unit size associated with the video data by setting two or more flags; and

coding (406) the video data based on the flags corresponding to the signaled coding tree unit size.

2. The method of claim 1, wherein the two or more flags comprises two or more from among a 32-pixel coding tree unit size flag, a 64-pixel coding tree unit size flag, and a 128-pixel coding tree unit size flag.

3. The method of claim 2, wherein the 32-pixel coding tree unit size flag is signaled first, and wherein the 64-pixel coding tree unit size flag is signaled based on the 32-pixel coding tree unit size flag being not equal to 1.

4. The method of claim 2, wherein the 64-pixel coding tree unit size flag is signaled first, and wherein the 32-pixel coding tree unit size flag is signaled based on the 64-pixel coding tree unit size flag being not equal to 1.

5. The method of claim 2, wherein the 32-pixel coding tree unit size flag is signaled first, and wherein the 128-pixel coding tree unit size flag is signaled based on the 32-pixel coding tree unit size flag being not equal to 1.

6. The method of claim 2, wherein the 128-pixel coding tree unit size flag is signaled first, and wherein the 32-pixel coding tree unit size flag is signaled based on the 128-pixel coding tree unit size flag being not equal to 1.

7. The method of claim 2, wherein the 64-pixel coding tree unit size flag is signaled first, and wherein the 128-pixel coding tree unit size flag is signaled based on the 64-pixel coding tree unit size flag being not equal to 1.

8. The method of claim 2, wherein the 128-pixel coding tree unit size flag is signaled first , and wherein the 64-pixel coding tree unit size flag is signaled based on the 128-pixel coding tree unit size flag being not equal to 1.

9. The method of claim 2, wherein a 64-pixel max luma transform size flag is signaled based on the coding tree unit size associated with the video data being greater than or equal to 64 pixels by 64 pixels.

10. The method of claim 9, wherein based on the 32-pixel coding tree unit size flag being signaled first and being equal to 1, the 64-pixel max luma transform size flag is not signaled.

11. The method of claim 9, wherein based on the 64-pixel coding tree unit size flag being signaled first and being equal to 0, the 32-pixel coding tree unit size flag is signaled and set equal to 1, and the 64-pixel max luma transform size flag is not signaled.

12. The method of claim 9, wherein based on the 128-pixel coding tree unit size flag being signaled first and being equal to 0, the 32-pixel coding tree unit size flag is signaled and set equal to 1, and the 64-pixel max luma transform size flag is not signaled.

13. The method of claim 1, wherein a first flag and a second flag from among the two or more flags correspond to a sequence parameter set associated with the video data, the first flag indicating whether a smallest coding tree unit size is applied and the second flag indicating whether a largest coding tree unit size is applied.

14. The method of claim 13, wherein the first flag is signaled first, and wherein the second flag is signaled based on the smallest coding tree unit size not being applied.

15. The method of claim 13, wherein second flag is signaled first, and wherein the first flag is signaled based on the largest coding tree unit size not being applied.

16. The method of claim 13, wherein a 64-pixel max luma transform size flag is signaled based on the smallest coding tree unit size not being applied.

17. A computer system for coding video data, the computer system comprising:

one or more computer-readable non-transitory storage media configured to store computer program code; and

one or more computer processors configured to access said computer program code and operate as instructed by said computer program code, said computer program code including performing the method according to any one of claims 1 to 16.

18. A non-transitory computer readable medium having stored thereon a computer program for coding video data, the computer program configured to cause one or more computer processors to perform the method according to any one of claims 1 to 16.