[0091] FIG. 9 depicts an example of a process 900 for encoding a partition for a video, according to some embodiments of the present disclosure. One or more computing devices (e.g., the computing device implementing the video encoder 100) implement operations depicted in FIG. 9 by executing suitable program code (e.g., the program code implementing the entropy coding module 116). For illustrative purposes, the process 900 is described with reference to some examples depicted in the figures. Other implementations, however, are possible.

[0092] At block 902, the process 900 involves accessing a partition of a video signal. The partition can be a video frame, a slice, or a tile or any type of partition processed by a video encoder as a unit when performing the encoding. The partition includes a set of CTUs arranged in CTU rows as shown in FIG. 8. Each CTU includes one or more CUs and each CTU includes multiple TUs for encoding as shown in the example of FIG. 6.

[0093] At block 904, which includes 906-914, the process 900 involves processing each CTU of the set of CTUs in the partition to encode the partition into bits. At block 906, the process 900 involves determining whether the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row. In some examples, the parallel coding may be indicated by a flag with a value 0 indicating parallel coding is disabled and value 1 indicating parallel coding is enabled. If it is determined that the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row, the process 900 involves setting, at block 908, the history counter StatCoeff to an initial value. As discussed above, the initial value of the history counter may be set according to Eqn. (5) or (6) if the history-based Rice parameter derivation is enabled; otherwise, the initial value of the history counter is set to zero.

[0094] If it is determined that the parallel coding mechanism is not enabled or that the current CTU is not the first CTU of the CTU row, or after setting the history counter at block 908, the process 900 involves, at block 910, calculating the Rice parameters for the TUs in the CTU based on the history counter. As described above in detail with respect to FIGS. 6-8, if the history counter is reset at block 908, the Rice parameters for the TUs in the CTU are calculated based on the reset history counter or subsequently updated history counter. If the history counter is not reset at block 908, the Rice parameters for the TUs in the CTU are calculated based on the history counter updated in the previous CTU or subsequently updated history counter in the current CTU.

[0095] At block 912, the process 900 involves encoding the TUs in the CTU into binary representation based on the calculated Rice parameters, such as through a combination of truncated Rice (TR) and limited k-th order EGK as specified in the VVC specification. At block 914, the process 900 involves encoding the binary representation of the CTU into the bits for inclusion in the bitstream of the video. The encoding can be performed, for example, using the context-adaptive binary arithmetic coding (CABAC) discussed above. At block 916, the process 900 involves outputting the encoded video bitstream.

[0096] FIG. 10 depicts an example of a process 1000 for decoding a partition for a video, according to some embodiments of the present disclosure. One or more computing devices implement operations depicted in FIG. 10 by executing suitable program code. For example, a computing device implementing the video decoder 200 may implement the operations depicted in FIG. 10 by executing the program code for the entropy decoding module 216, the inverse quantization module 218, and the inverse transform module 219. For illustrative purposes, the process 1000 is described with reference to some examples depicted in the figures. Other implementations, however, are possible.

[0097] At block 1002, the process 1000 involves accessing a binary string or a binary representation that represents a partition of a video signal. The partition can be a video frame, a slice, or a tile or any type of partition processed by a video encoder as a unit when performing the encoding. The partition includes a set of CTUs arranged in CTU rows as shown in FIG. 8. Each CTU includes one or more CUs and each CTU includes multiple TUs for encoding as shown in the example of FIG. 6.

[0098] At block 1004, which includes 1006-1014, the process 1000 involves processing the binary string for each CTU of the set of CTUs in the partition to generate decoded samples for the partition. At block 1006, the process 1000 involves determining whether the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row. The parallel coding may be indicated by a flag with a value 0 indicating parallel coding is disabled and value 1 indicating parallel coding is enabled. If it is determined that the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row, the process 1000 involves setting, at block 1008, the history counter StatCoeff to an initial value. As discussed above, the initial value of the history counter may be set according to Eqn. (5) or (6) if the history-based Rice parameter derivation is enabled; otherwise, the initial value of the history counter is set to zero.

video encoder as a unit when performing the encoding. The partition includes a set of CTUs arranged in CTU rows as shown in FIG. 8. Each CTU includes one or more CUs and each CTU includes multiple TUs for encoding as shown in the example of FIG. 6.

[0119] At block 1104, which includes 1106-1118, the process 1100 involves processing each CTU of the set of CTUs in the partition to encode the partition into bits. At block 1106, the process 1100 involves determining whether the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row. In some examples, the parallel coding may be indicated by a flag with a value 0 indicating parallel coding is disabled and value 1 indicating parallel coding is enabled. If it is determined that the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row, the process 1100 involves determining, at block 1107, whether the current CTU row is the first CTU row in the partition. If so, the process 1100 involves setting, at block 1108, the history counter StatCoeff to an initial value. As discussed above, the initial value of the history counter may be set according to Eqn. (5) or (6). If the current CTU row is not the first CTU row in the partition, the process 1100 involves setting, at block 1109, the history counter StatCoeff to a value stored in a history counter storage variable as shown in Eqn. (13) or (20). In some examples, such as when the alternative Rice parameter derivation is utilized, the value of the replacement variable HistValue can also be reset to a stored value as shown in Eqn. (21).

[0120] If it is determined that the parallel coding mechanism is not enabled or that the current CTU is not the first CTU of the CTU row, or after setting the value of the history counter in block 1108 or 1109, the process 1100 involves, at block 1110, calculating the Rice parameters for the TUs in the CTU based on the history counter (and the replacement variable HistValue if it is also reset). As described above (e.g., with respect to FIG. 8 or in the alternative Rice parameter derivation), if the value of the history counter is reset at block 1108 or 1109, the Rice parameters for the TUs in the CTU are calculated based on the reset history counter or subsequently updated history counter. If the history counter is not reset at block 1108 or 1109, the Rice parameters for the TUs in the CTU are calculated based on the history counter updated in the previous CTU or subsequently updated history counter in the current CTU.

[0121] At block 1112, the process 1100 involves encoding the TUs in the CTU into binary representation based on the calculated Rice parameters, such as through a

combination of truncated Rice (TR) and limited k-th order EGK as specified in the VVC specification. At block 1114, the process 1100 involves encoding the binary representation of the CTU into the bits for inclusion in the bitstream of the video. The encoding can be performed, for example, using the context-adaptive binary arithmetic coding (CABAC) discussed above.

[0122] At block 1116, the process 1100 involves determining whether the parallel coding is enabled and the CTU is the first CTU of the current CTU row. If so, the process 1100 involves, at block 1118, storing the value of the history counter in a history counter storage variable, as shown in Eqn. (12) or (18). In some examples, such as when the alternative Rice parameter derivation is utilized, the value of the replacement variable HistValue can also be stored in a storage variable as shown in Eqn. (19). At block 1120, the process 1100 involves outputting the encoded video bitstream.

[0123] In some scenarios, a CTU in a non-first CTU row may be at the boundary of the partition. For example, the first CTU in the second CTU row does not have a CTU in the partition that locates on the top of this CTU. In these scenarios, the history counter for this CTU can be set to an initial value instead of the stored value. In this case, a new block 1107' can be added to FIG. 11 between block 1107 and block 1109 to determine if the CTU is at the boundary of the partition (e.g., the CTU does not have a top neighboring CTU within the partition). If so, the process 1100 proceeds to block 1108 to set the history counter to an initial value; if not, the process 1100 proceeds to block 1109 to set the history counter to a stored value. The rest blocks of FIG. 11 can remain the same.

[0124] FIG. 12 depicts an example of a process 1200 for decoding a partition for a video, according to some embodiments of the present disclosure. One or more computing devices implement operations depicted in FIG. 12 by executing suitable program code. For example, a computing device implementing the video decoder 200 may implement the operations depicted in FIG. 12 by executing the program code for the entropy decoding module 216, the inverse quantization module 218, and the inverse transform module 219. For illustrative purposes, the process 1200 is described with reference to some examples depicted in the figures. Other implementations, however, are possible.

[0125] At block 1202, the process 1200 involves accessing a binary string or a binary representation that represents a partition of a video signal. The partition can be a video frame, a slice, or a tile or any type of partition processed by a video encoder as a unit when

performing the encoding. The partition includes a set of CTUs arranged in CTU rows as shown in FIG. 8. Each CTU includes one or more CUs and each CTU includes multiple TUs for encoding as shown in the example of FIG. 6.

[0126] At block 1204, which includes 1206-1218, the process 1200 involves processing the binary string for each CTU of the set of CTUs in the partition to generate decoded samples for the partition. At block 1206, the process 1200 involves determining whether the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row. The parallel coding may be indicated by a flag with a value 0 indicating parallel coding is disabled and value 1 indicating parallel coding is enabled. If it is determined that the parallel coding mechanism is enabled and the current CTU is the first CTU of the CTU row, the process 1200 involves determining, at block 1207, whether the current CTU row is the first CTU row in the partition. If so, the process 1200 involves setting, at block 1208, the history counter StatCoeff to an initial value. As discussed above, the initial value of the history counter may be set according to Eqn. (5) or (6). If the current CTU row is not the first CTU row in the partition, the process 1200 involves setting, at block 1209, the history counter StatCoeff to a value stored in a history counter storage variable as shown in Eqn. (13) or (20). In some examples, such as when the alternative Rice parameter derivation is utilized, the value of the replacement variable HistValue can also be reset to a stored value as shown in Eqn. (21).

[0127] If it is determined that the parallel coding mechanism is not enabled or that the current CTU is not the first CTU of the CTU row, or after setting the history counter at block 1208 or 1209, the process 1200 involves, at block 1210, calculating the Rice parameters for the TUs in the CTU based on the history counter (and the replacement variable HistValue if its value is also set). As described above (e.g., with respect to FIG. 8 or in the alternative Rice parameter derivation), if the value of the history counter is reset at block 1208 or 1209, the Rice parameters for the TUs in the CTU are calculated based on the reset history counter or subsequently updated history counter. If the history counter is not reset at block 1208 or 1209, the Rice parameters for the TUs in the CTU are calculated based on the history counter updated in the previous CTU or subsequently updated history counter in the current CTU.

[0128] At block 1212, the process 1200 involves decoding the binary strings or binary representations of TUs in the CTU into coefficient values based on the calculated Rice

Claims

1. A method for decoding a video from a video bitstream encoded using Versatile Video Coding (VVC), the method comprising:

determining a bit depth of samples of the video based on Sequence Parameter Set (SPS) syntax element sps_bitdepth_minus8, wherein a value of the SPS syntax element sps_bitdepth_minus8 is in a range of 0 to 8;

determining a size of a decoded picture buffer (DPB) based on a Video Parameter Set (VPS) syntax element vps_ols_dpb_bitdepth_minus8**,** wherein a value of the VPS syntax element vps_ols_dpb_bitdepth_minus8 is in a range of 0 to 8;

allocating a storage space with the determined size of the DPB;

obtaining and storing a decoded picture in the DPB by decoding the video bitstream based on the determined bit depth; and

outputting the decoded picture.

2. The method of claim 1, wherein obtaining and storing the decoded picture in the DPB by decoding the video bitstream comprises:

obtaining and storing a first set of pictures in the DPB by decoding a first portion of the video bitstream; and

decoding a second portion of the video bitstream into a second set of pictures by using the first set of pictures in the DPB as reference pictures.

3. The method of claim 1, further comprising:

reordering one or more decoded pictures in the DPB according to an output order; and

outputting the one or more decoded pictures according to the output order.

4. The method of claim 1, further comprising:

holding one or more decoded pictures in the DPB for a period of time; and

outputting the one or more decoded pictures after the period of time has passed.

5. The method of claim 1, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU, determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row among the one or more CTU rows in the partition;

in response to determining that the parallel coding is enabled and that the CTU is the first CTU of the current CTU row in the partition,

setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

outputting a decoded partition of the video comprising the decoded plurality of CTUs in the partition.

6. The method of claim 1, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU and in response to determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row, determining whether the current CTU row is the first CTU row in the partition;

in response to determining that the current CTU row is the first CTU row in the partition, setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

in response to determining that the current CTU row is not the first CTU row in the partition, setting the history counter for a color component to a value stored in a history counter storage variable;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the value of the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

after decoding the CTU, in response to determining that parallel coding is enabled and that the CTU is the first CTU of the current CTU row, storing a current value of the history counter in the history counter storage variable.

7. A non-transitory computer-readable medium having program code that is stored thereon, the program code executable by one or more processing devices for performing operations comprising:

determining a size of a decoded picture buffer (DPB) for a video based on a Video Parameter Set (VPS) syntax element vps_ols_dpb_bitdepth_minus8 included in a video bitstream of the video, wherein the video bitstream is encoded using Versatile Video Coding (VVC) and a value of the VPS syntax element vps_ols_dpb_bitdepth_minus8 is in a range of 0 to 8;

allocating a storage space with the determined size of the DPB;

obtaining and storing a decoded picture in the DPB by decoding the video bitstream; and

outputting the decoded picture.

8. The non-transitory computer-readable medium of claim 7, wherein obtaining and storing the decoded picture in the DPB by decoding the video bitstream comprises:

obtaining and storing a first set of pictures in the DPB by decoding a first portion of the video bitstream; and

decoding a second portion of the video bitstream into a second set of pictures by using the first set of pictures in the DPB as reference pictures.

9. The non-transitory computer-readable medium of claim 7, wherein the operations further comprise:

reordering one or more decoded pictures in the DPB according to an output order; and

outputting the one or more decoded pictures according to the output order.

10. The non-transitory computer-readable medium of claim 7, wherein the operations further comprise:

holding one or more decoded pictures in the DPB for a period of time; and

outputting the one or more decoded pictures after the period of time has passed.

11. The non-transitory computer-readable medium of claim 7, wherein the operations further comprise:

determining a bit depth of samples of the video based on a Sequence Parameter Set (SPS) syntax element sps_bitdepth_minus8, wherein a value of the SPS syntax element sps_bitdepth_minus8 is in a range of 0 to 8, wherein decoding the video bitstream is performed based on the determined bit depth.

12. The non-transitory computer-readable medium of claim 11, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU, determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row among the one or more CTU rows in the partition;

in response to determining that the parallel coding is enabled and that the CTU is the first CTU of the current CTU row in the partition,

setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

outputting a decoded partition of the video comprising the decoded plurality of CTUs in the partition.

13. The non-transitory computer-readable medium of claim 11, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU and in response to determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row, determining whether the current CTU row is the first CTU row in the partition;

in response to determining that the current CTU row is the first CTU row in the partition, setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

in response to determining that the current CTU row is not the first CTU row in the partition, setting the history counter for a color component to a value stored in a history counter storage variable;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the value of the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

after decoding the CTU, in response to determining that parallel coding is enabled and that the CTU is the first CTU of the current CTU row, storing a current value of the history counter in the history counter storage variable.

14. A system comprising:

a processing device; and

a non-transitory computer-readable medium communicatively coupled to the processing device, wherein the processing device is configured to execute program code stored in the non-transitory computer-readable medium and thereby perform operations comprising:

determining a size of a decoded picture buffer (DPB) for a video based on a Video Parameter Set (VPS) syntax element vps_ols_dpb_bitdepth_minus8 included in a video bitstream of the video**,** wherein the video bitstream is encoded using Versatile Video Coding (VVC) and a value of the VPS syntax element vps_ols_dpb_bitdepth_minus8 is in a range of 0 to 8;

allocating a storage space with the determined size of the DPB;

obtaining and storing a decoded picture in the DPB by decoding the video bitstream; and

outputting the decoded picture.

15. The system of claim 14, wherein obtaining and storing the decoded picture in the DPB by decoding the video bitstream comprises:

obtaining and storing a first set of pictures in the DPB by decoding a first portion of the video bitstream; and

decoding a second portion of the video bitstream into a second set of pictures by using the first set of pictures in the DPB as reference pictures.

16. The system of claim 14, wherein the operations further comprise:

reordering one or more decoded pictures in the DPB according to an output order; and

outputting the one or more decoded pictures according to the output order.

17. The system of claim 14, wherein the operations further comprise:

holding one or more decoded pictures in the DPB for a period of time; and

outputting the one or more decoded pictures after the period of time has passed.

18. The system of claim 14, wherein the operations further comprise:

determining a bit depth of samples of the video based on Sequence Parameter Set (SPS) syntax element sps_bitdepth_minus8, wherein a value of the syntax element sps_bitdepth_minus8 is in a range of 0 to 8, wherein decoding the video bitstream is performed based on the determined bit depth.

19. The system of claim 18, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU, determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row among the one or more CTU rows in the partition;

in response to determining that the parallel coding is enabled and that the CTU is the first CTU of the current CTU row in the partition,

setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

outputting a decoded partition of the video comprising the decoded plurality of CTUs in the partition.

20. The system of claim 18, wherein decoding the video bitstream based on the determined bit depth comprises:

accessing a binary string representing a partition of the video, the partition comprising a plurality of coding tree units (CTUs) forming one or more CTU rows;

for each CTU of the plurality of CTUs in the partition,

prior to decoding the CTU and in response to determining that parallel coding is enabled and that the CTU is the first CTU of a current CTU row, determining whether the current CTU row is the first CTU row in the partition;

in response to determining that the current CTU row is the first CTU row in the partition, setting a history counter for a color component for calculating Rice parameters to an initial value based on the determined bit depth;

in response to determining that the current CTU row is not the first CTU row in the partition, setting the history counter for a color component to a value stored in a history counter storage variable;

decoding the CTU, comprising:

calculating the Rice parameters for transform units (TUs) in the CTU based on the value of the history counter;

decoding the binary string corresponding to the TUs in the CTU into coefficient values of the TUs based on the calculated Rice parameters; and

determining pixel values for the TUs in the CTU from the coefficient values; and

after decoding the CTU, in response to determining that parallel coding is enabled and that the CTU is the first CTU of the current CTU row, storing a current value of the history counter in the history counter storage variable.