United States Patent Application Publication 20250260842
Kind Code A1
Publication Date August 14, 2025
Inventor(s) LAROCHE; Guillaume et al.

HIGH LEVEL SYNTAX FOR VIDEO CODING AND DECODING

Abstract

There is provided a method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices. The bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements. The method includes omitting parsing one or more syntax elements for a slice relating to use or availability of a decoding tool or parameter for that slice if one or more syntax elements are parsed that indicate that the picture contains only one slice; and decoding said bitstream using said syntax elements.

Inventors: LAROCHE; Guillaume (Saint Aubin d'Aubigné, FR), OUEDRAOGO; Naël (Val d'Anast, FR), ONNO; Patrice (RENNES, FR)

Applicant: CANON KABUSHIKI KAISHA (Tokyo, JP)

Family ID: 70546559

Appl. No.: 19/195469

Filed: April 30, 2025

Foreign Application Priority Data

GB	2004095.2	Mar. 20, 2020

Related U.S. Application Data

parent US continuation 17913114 20220920 parent-grant-document US 12323627 WO continuation PCT/EP2021/056870 20210317 child US 19195469

Publication Classification

Int. Cl.: **H04N19/70** (20140101); **H04N19/105** (20140101); **H04N19/156** (20140101); **H04N19/157** (20140101); **H04N19/172** (20140101); **H04N19/174** (20140101); **H04N19/186** (20140101); **H04N19/44** (20140101)

U.S. Cl.:

CPC **H04N19/70** (20141101); **H04N19/105** (20141101); **H04N19/156** (20141101); **H04N19/157** (20141101); **H04N19/172** (20141101); **H04N19/174** (20141101); **H04N19/186** (20141101); **H04N19/44** (20141101);

Background/Summary

CROSS REFERENCE TO RELATED APPLICATION [0001] This application is a Continuation Application of U.S. patent application Ser. No. 17/913,114, filed on Sep. 20, 2022, which is the National Phase application of PCT Application No. PCT/EP2021/056870, filed on Mar. 17, 2021. This application claims the benefit under 35 U.S.C. § 119 (a)-(d) of United Kingdom Patent Application No. 2004095.2, filed on Mar. 20, 2020. The above cited patent applications are incorporated herein by reference in their entirety.

FIELD OF INVENTION

[0002] The present invention relates to video coding and decoding, and in particular to the high level syntax used in the bitstream.

BACKGROUND

[0003] Recently, the Joint Video Experts Team (JVET), a collaborative team formed by MPEG and ITU-T Study Group 16's VCEG, commenced work on a new video coding standard referred to as Versatile Video Coding (VVC). The goal of VVC is to provide significant improvements in compression performance over the existing HEVC standard (i.e., typically twice as much as before) and to be completed in 2020. The main target applications and services include—but not limited to—360-degree and high-dynamic-range (HDR) videos. In total, JVET evaluated responses from 32 organizations using formal subjective tests conducted by independent test labs. Some proposals demonstrated compression efficiency gains of typically 40% or more when compared to using HEVC. Particular effectiveness was shown on ultra-high definition (UHD) video test material. Thus, we may expect compression efficiency gains well-beyond the targeted 50% for the final standard.

[0004] The JVET exploration model (JEM) uses all the HEVC tools and has introduced a number of new tools. These changes have necessitated a change to the structure of the bitstream, and in particular to the high-level syntax which can have a impact on the overall bitrate of the bitstream.

SUMMARY

[0005] The present invention relates to an improvement to the high level syntax structure, which leads to a reduction in complexity and/or signalling without any significant degradation in coding performance.

[0006] In a first aspect according to the present invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and omitting parsing one or more syntax elements for a slice relating to use or availability of a decoding tool or parameter for that slice if one or more syntax elements are parsed that indicate that the picture contains only one slice; and decoding said bitstream using said syntax elements. In another aspect according to the present invention, there is provided method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, said bitstream being constrained so that in a case where the bitstream includes a syntax element having a value indicating that the picture contains only one slice, the bitstream includes a syntax element indicating that the parsing of one or more syntax elements for a slice relating to use or availability of a decoding tool or parameter for that slice is to be omitted, the method comprising decoding said bitstream using said syntax elements. Thus, a coding efficiency improvement is achieved as certain syntax elements are not transmitted when not needed. In particular, when the current picture contains only one slice, there is no additional flexibility to signal certain parameters in the picture header and then slice header.

[0007] A syntax element relating to the use or availability of the decoding tool or parameter may be signalled in a picture parameter set (PPS) or a sequence parameter set (SPS).

[0008] The one or more syntax elements indicating that the current picture contains only one slice may comprise a picture header in slice header syntax element which indicates that a picture header is signalled in the slice header. The advantage is a coding efficiency improvement when the picture is in the slice header. Indeed, the signalling of picture header in the slice header is efficient for low delay and low bitrate applications. In that cases the overriding flag cost for a multitude of pictures is larger than the cost of setting set of the several reference picture lists in the SPS. Indeed, generally for these use cases, the amount of reference frames is limited to one or two reference frames per list.

[0009] The one or more syntax elements indicating that the current picture contains only one slice comprise a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, and wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0010] The syntax element relating to use or availability of a decoding tool or parameter may comprise a flag for indicating use of the decoding mode or parameter at the slice level, and further comprising predicting the overriding the decoding mode or parameter at the slice level from the value of a flag at the picture header level. For example, the decoding tool or parameter may relate to reference frame signalling and the syntax element relating to use or availability of the decoding tool or parameter is a flag for overriding of the use of the reference picture list. If the slice uses a reference picture list (or reference picture lists) transmitted in the SPS, there is an advantage to override one or more list to limit the number of reference pictures. Surprisingly, however, in terms of coding efficiency compromise for real applications, it is preferable to avoid this overriding to save the bits related to its signalling. Moreover, it simplifies the slice header parsing for some implementations.

[0011] The decoding tool or parameter may relate to LCMS and the syntax element comprise an activation flag for LCMS.

[0012] The decoding tool or parameter may relate to a scaling list and the syntax element comprise an activation flag for the scaling list.

[0013] In a second aspect according to the present invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and setting a value

of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if one or more syntax elements are parsed that indicate that the picture contains only one slice; and decoding said bitstream using said syntax elements. In terms of implementation of the decoding, this simplifies the slice header parsing.

[0014] The one or more syntax elements indicating that the current picture contains only one slice may comprise a picture header in slice header syntax element which indicates that a picture header is signalled in the slice header

[0015] Each slice may include one or more tiles and the one or more syntax elements indicating that the current picture contains only one slice may comprise a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0016] In a third aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and setting a value of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if a picture header in slice header syntax element indicates that a picture header is signalled in the slice header; and decoding said bitstream using said syntax elements.

[0017] In a fourth aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and setting a value of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice are parsed, and the number of tiles in the picture is greater than one and number of tiles in the slice is equal to the number of tiles in the picture; and decoding said bitstream using said syntax elements.

[0018] In at least the second to fourth aspects, the syntax element relating to the overriding of a reference picture list decoding tool may comprise a flag and setting the flag to be not enabled indicates that the reference picture list is not overridden.

[0019] In a fifth aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises one or more syntax elements, the decoding comprising: parsing one or more syntax elements indicating whether a reference picture list for a picture to be decoded refers to a sequence parameter set (SPS) reference picture list; omitting parsing in a slice header one or more syntax elements relating to the overriding of a reference picture list for a slice of the picture if one or more syntax elements indicate that the reference picture list for the picture to be decoded does not refer to a sequence parameter set (SPS) reference picture list; and decoding said bitstream using said syntax elements. Thus, parsing can be simplified and unnecessary syntax elements avoided.

[0020] The omitting of the parsing of the one or more syntax elements relating to the overriding of the reference picture list may further require that the picture has only one slice. When a picture contains one slice, an encoder typically should not override a reference picture list which is explicitly transmitted in slice header or in the picture header. But if it uses a reference picture lists transmitted in the SPS, there is an advantage to override the list to limit the number of reference pictures. But surprisingly in terms of coding efficiency compromise for real application, it is preferable to avoid this overriding to save the bits related to its signalling. Moreover, it simplifies the slice header parsing for some implementations.

[0021] The one or more syntax elements may comprise a picture header in slice header syntax element which indicates whether a picture header is signalled in the slice header and the omitting further requires that the picture header is signalled in the slice header. An advantage is a coding efficiency improvement when the picture is in the slice header. Indeed, the signalling of the picture header in the slice header is efficient for low delay and low bitrate applications, in that cases the overriding flag cost for a multitude of pictures is larger than the cost of setting the several reference picture lists in the SPS.

[0022] Each slice may include one or more tiles, and the one or more syntax elements a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, and the omitting requires that the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture. Thus, this gives an effective way of determining if a picture has just one slice.

[0023] The parsing of the one or more syntax elements relating to the overriding of the reference picture list is omitted if a reference picture list is signalled in the slice header. An advantage is a coding efficiency improvement because it is not needed to update the reference picture lists if they are explicitly transmitted in the slice header.

[0024] In a sixth aspect according to the invention there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises one or more syntax elements, the decoding comprising: parsing a high-level syntax element at a level in the bitstream higher than a slice, the high-level syntax element indicating whether it is permitted to indicate the use or availability of a decoding tool or parameter at a slice level; omitting parsing or inferring, for a slice header, one or more syntax elements indicating the use or availability of a decoding tool or parameter for the slice, if the high level syntax element indicates that it is not permitted to indicate the use or availability of the decoding tool or parameter at a slice level;

and decoding said bitstream using said syntax elements. In a further aspect according to the invention there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises one or more syntax elements, said bitstream being constrained so that in a case where the bitstream includes a high-level syntax element at a level in the bitstream higher than a slice, the high-level syntax element indicating whether it is permitted to indicate the use or availability of a decoding tool or parameter at a slice level, the bitstream also includes a syntax element indicating that parsing or inferring, for a slice header, one or more syntax elements indicating the use or availability of a decoding tool or parameter for the slice is to be omitted, if the high level syntax element indicates that it is not permitted to indicate the use or availability of the decoding tool or parameter at a slice level, the method comprising decoding said bitstream using said syntax elements. Thus, a more flexible implementation compared to the previous aspects can be provided but with a similar coding efficiency improvement.

[0025] The high-level syntax element may be signalled in one or more of a sequence parameter set (SPS), a picture parameter set (PPS), a video parameter set (VPS), and a picture header (PH).

[0026] The high-level syntax element may not be decoded if a number of reference pictures lists in a sequence parameter set (SPS) is zero.

[0027] The high-level syntax element may not be decoded if the picture has only one slice.

[0028] The decoding tool or parameter may relate to a reference picture list. The high-level syntax element may not be decoded if a reference picture list is signalled in the slice header.

[0029] The syntax element indicating the use or availability of a decoding tool or parameter may comprise an activation flag. The decoding tool or parameter may be a luma mapping with chroma scaling (LMCS) tool. The decoding tool or parameter may be a scaling list. The activation flag in the slice header may depend on the (corresponding) activation flag on picture header. For example, the value of the activation flag for a slice may be inferred from a value of an activation flag for the decoding tool or parameter signalled in a picture header if the high level syntax element indicates that it is permitted to indicate the use or availability of the decoding tool or parameter at a slice level.

[0030] In a seventh aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, and wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and if a picture contains only one slice, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for a slice in the picture to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in a picture header for the picture containing the slice; and decoding said bitstream using said syntax elements. According to a yet further aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, and [0031] wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, said bitstream being constrained so that [0032] in a case where the bitstream includes a syntax element having a value indicating that a picture contains only one slice, the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for a slice in the picture is restricted to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in a picture header for the picture containing the slice, the method comprising decoding said bitstream using said syntax elements. An advantage, is a coding efficiency improvement as the syntax elements are not transmitted when they are not needed. Indeed, when the current picture contains only one slice, there is no additional flexibility to signal it in the picture header and then slice header.

[0033] The method may further comprise parsing a picture header in slice header syntax element, wherein if the picture header in slice header syntax element indicates that a picture header is signalled in the slice header, the picture contains only one slice. The advantage is a coding efficiency improvement when the picture is in the slice header. Indeed, the picture header in the slice header is efficient for low delay and low bitrate applications, in that cases the signalling at slice level has a significant cost on the global bitrate.

[0034] Each slice may include one or more tiles, and the method may further comprise parsing a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0035] In an eighth aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and if a picture header is signalled in the slice header, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for a slice in the picture to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in the picture header for the picture containing the slice; and decoding said bitstream using said syntax elements In an additional aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, said bitstream being constrained so that in a case where the bitstream includes a syntax element having a value indicating that a picture header is signalled in the slice header, the value of one or more syntax elements

indicating the use or availability of one or more decoding tool or parameter for a slice in the picture is restricted to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in the picture header for the picture containing the slice, the method comprising decoding said bitstream using said syntax elements.

[0036] In a ninth aspect according to the invention, there is provided a method of decoding video data from a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein each slice may include one or more tiles, and wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: parsing the syntax elements, and if a number of tiles in a picture being decoded is greater than one and number of tiles in a slice is equal to the number of tiles in the picture, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for the slice to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in a picture header for the picture containing the slice; and decoding said bitstream using said syntax elements.

[0037] In at least the seventh to ninth aspects, the one or more decoding tool or parameter may comprise a luma mapping with chroma scaling (LMCS) tool.

[0038] In at least the seventh to ninth aspects, the one or more decoding tool or parameter may comprise a scaling list.

[0039] In a tenth aspect according to the present invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: determining the syntax elements; omitting encoding of one or more syntax elements for a slice relating to use or availability of a decoding tool or parameter for that slice if one or more syntax elements are determined that indicate that the picture contains only one slice; and encoding said video data using said syntax elements.

[0040] The syntax element relating to the use or availability of the decoding tool or parameter may be signalled in a picture parameter set (PPS).

[0041] The one or more syntax elements indicating that the current picture contains only one slice may comprise a picture header in slice header syntax element which indicates that a picture header is signalled in the slice header.

[0042] The one or more syntax elements indicating that the current picture contains only one slice may comprise a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, and wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0043] The syntax element relating to use or availability of a decoding tool or parameter may comprise a flag for indicating use of the decoding mode or parameter at the slice level, and further comprising predicting the overriding the decoding mode or parameter at the slice level from the value of a flag at the picture header level.

[0044] The decoding tool or parameter is reference frame signalling and the syntax element relating to use or availability of the decoding tool or parameter may comprise a flag for overriding of the use of the reference picture list.

[0045] The decoding tool or parameter may comprise luma mapping with chroma scaling (LMCS) and the syntax element is an activation flag for luma mapping with chroma scaling (LMCS).

[0046] The decoding tool or parameter may comprise a scaling list and the syntax element may comprise an activation flag for the scaling list.

[0047] In an eleventh aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: determining the syntax elements, and setting a value of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if one or more syntax elements are determined that indicate that the picture contains only one slice; and encoding said video data using said syntax elements.

[0048] The one or more syntax elements indicating that the current picture contains only one slice may comprise a picture header in slice header syntax element which indicates that a picture header are signalled in the slice header.

[0049] Each slice may include one or more tiles, and the one or more syntax elements indicating that the current picture contains only one slice may comprise a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0050] In a twelfth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: determining the syntax elements, and setting a value of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if a picture header in slice header syntax element indicates that a picture header is signalled in the slice header; and encoding said video data using said syntax elements.

[0051] In a thirteenth aspect according to the invention, there is provided a method of encoding video data from a bitstream, the bitstream comprising video data corresponding to one or more slices, wherein each slice may include one or more tiles,

and wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: determining the syntax elements, and setting a value of one or more syntax elements for a slice relating to the overriding of a reference picture list decoding tool or parameter to indicate for the slice that the reference picture list is not to be overridden if a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice are determined, and the number of tiles in the picture is greater than one and number of tiles in the slice is equal to the number of tiles in the picture; and encoding said video data using said syntax elements.

[0052] In at least the eleventh to thirteenth aspects, the syntax element relating to the overriding of a reference picture list decoding tool may comprise a flag and setting the flag to be not enabled indicates that the reference picture list is not overridden.

[0053] In a fourteenth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises one or more syntax elements, the encoding comprising: determining one or more syntax elements indicating whether a reference picture list for a picture to be decoded refers to a sequence parameter set (SPS) reference picture list; omitting encoding in a slice header one or more syntax elements relating to the overriding of a reference picture list for a slice of the picture if one or more syntax elements indicate that the reference picture list for the picture to be decoded does not refer to a sequence parameter set (SPS) reference picture list; and encoding said video data using said syntax elements.

[0054] The omitting of the parsing of the one or more syntax elements relating to the overriding of the reference picture list may further require that the picture has only one slice. The one or more syntax elements comprise a picture header in slice header syntax element which indicates whether a picture header is signalled in the slice header and the omitting requires that the picture header is signalled in the slice header each slice may include one or more tiles, and the one or more syntax elements indicating that the current picture has only one slice comprises a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture.

[0055] The encoding of the one or more syntax elements relating to the overriding of the reference picture list may be omitted if a reference picture list is signalled in the slice header.

[0056] In a fifteenth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, and the bitstream comprising one or more syntax elements, the encoding comprising: encoding a high-level syntax element at a level in the bitstream higher than a slice, the high-level syntax element indicating whether it is permitted to indicate the use or availability of a decoding tool or parameter at a slice level; omitting encoding, for a slice header, one or more syntax elements indicating the use or availability of a decoding tool or parameter for the slice, if the high level syntax element indicates that it is not permitted to indicate the use or availability of the decoding tool or parameter at a slice level; and encoding said video data using said syntax elements.

[0057] The high-level syntax element may be signalled in one or more of a sequence parameter set (SPS), a picture parameter set (PPS), a video parameter set (VPS), and a picture header (PH).

[0058] Optionally, the high-level syntax element is not encoded if a number of reference pictures lists in a sequence parameter set (SPS) is zero.

[0059] Optionally, the high-level syntax element is not encoded if the picture has only one slice.

[0060] The decoding tool or parameter may relate to a reference picture list. Optionally, the high-level syntax element is not encoded if a reference picture list is signalled in the slice header.

[0061] The syntax element indicating the use or availability of a decoding tool or parameter may comprise an activation flag. The decoding tool or parameter may comprise a luma mapping with chroma scaling (LMCS) tool. The decoding tool or parameter may comprise a scaling list.

[0062] The activation flag in the slice header may depends on a (corresponding) activation flag in the picture header. For example, the value of the activation flag for a slice may be inferred from a value of an activation flag for the decoding tool or parameter signalled in a picture header if the high level syntax element indicates that it is permitted to indicate the use or availability of the decoding tool or parameter at a slice level.

[0063] In a sixteenth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: parsing the syntax elements, and if a picture contains only one slice, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for a slice in the picture to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in a picture header for the picture containing the slice; and encoding said video data using said syntax elements.

[0064] Optionally, the method may comprise encoding a picture header in slice header syntax element, wherein if the picture header in slice header syntax element indicates that a picture header is signalled in the slice header, the picture contains only one slice.

[0065] Optionally, the method may comprise encoding a syntax element indicating a number of tiles in the current picture and a syntax element indicating a number of tiles in the slice, wherein the number of tiles in the picture being greater than one and number of tiles in the slice being equal to the number of tiles in the picture indicates that the current picture contains only one slice.

[0066] In a seventeenth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the encoding comprising: determining the syntax elements, and if a picture header is signalled in the slice header, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for a slice in the picture to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in the picture header for the picture containing the slice; and encoding said video data using said syntax elements.

[0067] In an eighteenth aspect according to the invention, there is provided a method of encoding video data into a bitstream, the bitstream comprising video data comprising a sequence of pictures having one or more slices, wherein each slice may include one or more tiles, and wherein the bitstream comprises a picture header comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the decoding comprising: determining the syntax elements, and if a number of tiles in a picture being encoded is greater than one and number of tiles in a slice is equal to the number of tiles in the picture, restricting the value of one or more syntax elements indicating the use or availability of one or more decoding tool or parameter for the slice to a same value as a corresponding syntax element indicating the use or availability of the decoding tool or parameter signalled in a picture header for the picture containing the slice; and encoding said video data using said syntax elements.

[0068] The one or more decoding tool or parameter may comprise a luma mapping with chroma scaling (LMCS) tool.

[0069] The one or more decoding tool or parameter may comprise a scaling list. In a nineteenth aspect according to the invention, there is provided a device for decoding video data from a bitstream, the device being configured to perform the method of any of claims first to ninth aspects.

[0070] In a twentieth aspect according to the invention, there is provided a device for encoding video data into a bitstream, the device being configured to perform the method of any of the tenth to eighteenth aspects.

[0071] In a twenty first aspect according to the invention, there is provided a computer program comprising executable instructions which upon execution cause the method of any of the above-mentioned aspects to be performed.

[0072] The program may be provided on its own or may be carried on, by or in a carrier medium. The carrier medium may be non-transitory, for example a storage medium, in particular a computer-readable storage medium. The carrier medium may also be transitory, for example a signal or other transmission medium. The signal may be transmitted via any suitable network, including the Internet. Further features of the invention are characterised by the independent and dependent claims

[0073] Any feature in one aspect of the invention may be applied to other aspects of the invention, in any appropriate combination. In particular, method aspects may be applied to apparatus aspects, and vice versa.

[0074] Furthermore, features implemented in hardware may be implemented in software, and vice versa. Any reference to software and hardware features herein should be construed accordingly

[0075] Any apparatus feature as described herein may also be provided as a method feature, and vice versa. As used herein, means plus function features may be expressed alternatively in terms of their corresponding structure, such as a suitably programmed processor and associated memory.

[0076] It should also be appreciated that particular combinations of the various features described and defined in any aspects of the invention can be implemented and/or supplied and/or used independently.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0077] Reference will now be made, by way of example, to the accompanying drawings, in which:

[0078] FIG. **1** is a diagram for use in explaining a coding structure used in HEVC and VVC;

[0079] FIG. **2** is a block diagram schematically illustrating a data communication system in which one or more embodiments of the invention may be implemented;

[0080] FIG. **3** is a block diagram illustrating components of a processing device in which one or more embodiments of the invention may be implemented;

[0081] FIG. **4** is a flow chart illustrating steps of an encoding method according to embodiments of the invention;

[0082] FIG. **5** is a flow chart illustrating steps of a decoding method according to embodiments of the invention;

[0083] FIG. **6** illustrates the structure of the bitstream in the exemplary coding system VVC

[0084] FIG. **7** illustrates another structure of the bitstream in the exemplary coding system VVC

[0085] FIG. **8** illustrates Luma Modelling Chroma Scaling (LMCS);

[0086] FIG. **9** shows a sub tool of LMCS;

[0087] FIG. **10** is the illustration of the raster-scan slice mode and the rectangular slice mode of the current VVC draft standard.

[0088] FIG. **11** is a diagram showing a system comprising an encoder or a decoder and a communication network according to embodiments of the present invention;

[0089] FIG. **12** is a schematic block diagram of a computing device for implementation of one or more embodiments of the invention;

[0090] FIG. **13** is a diagram illustrating a network camera system; and

[0091] FIG. **14** is a diagram illustrating a smart phone.

DETAILED DESCRIPTION

[0092] FIG. **1** relates to a coding structure used in the High Efficiency Video Coding (HEVC) video standard. A video sequence **1** is made up of a succession of digital images i. Each such digital image is represented by one or more matrices. The matrix coefficients represent pixels.

[0093] An image **2** of the sequence may be divided into slices **3**. A slice may in some instances constitute an entire image. These slices are divided into non-overlapping Coding Tree Units (CTUs). A Coding Tree Unit (CTU) is the basic processing unit of the High Efficiency Video Coding (HEVC) video standard and conceptually corresponds in structure to macroblock units that were used in several previous video standards. A CTU is also sometimes referred to as a Largest Coding Unit (LCU). A CTU has luma and chroma component parts, each of which component parts is called a Coding Tree Block (CTB). These different color components are not shown in FIG. **1**.

[0094] A CTU is generally of size 64 pixels×64 pixels. Each CTU may in turn be iteratively divided into smaller variable-size Coding Units (CUs) 5 using a quadtree decomposition.

[0095] Coding units are the elementary coding elements and are constituted by two kinds of sub-unit called a Prediction Unit (PU) and a Transform Unit (TU). The maximum size of a PU or TU is equal to the CU size. A Prediction Unit corresponds to the partition of the CU for prediction of pixels values. Various different partitions of a CU into PUs are possible as shown by **606** including a partition into 4 square PUs and two different partitions into 2 rectangular PUs. A Transform Unit is an elementary unit that is subjected to spatial transformation using DCT. A CU can be partitioned into TUs based on a quadtree representation **607**.

[0096] Each slice is embedded in one Network Abstraction Layer (NAL) unit. In addition, the coding parameters of the video sequence are stored in dedicated NAL units called parameter sets. In HEVC and H.264/AVC two kinds of parameter sets NAL units are employed: first, a Sequence Parameter Set (SPS) NAL unit that gathers all parameters that are unchanged during the whole video sequence. Typically, it handles the coding profile, the size of the video frames and other parameters. Secondly, a Picture Parameter Set (PPS) NAL unit includes parameters that may change from one image (or frame) to another of a sequence. HEVC also includes a Video Parameter Set (VPS) NAL unit which contains parameters describing the overall structure of the bitstream. The VPS is a new type of parameter set defined in HEVC, and applies to all of the layers of a bitstream. A layer may contain multiple temporal sub-layers, and all version 1 bitstreams are restricted to a single layer. HEVC has certain layered extensions for scalability and multiview and these will enable multiple layers, with a backwards compatible version 1 base layer.

[0097] In the current definition of the Versatile Video Coding (VVC), there is three high level possibilities for the partitioning of a picture: subpictures, slices and tiles. Each having their own characteristics and usefulness. The partitioning into subpictures is for the spatial extraction and/or merging of regions of a video. The partitioning into slices is based on a similar concept as the previous standards and corresponds to packetization for video transmission even if it can be used for other applications. The partitioning into Tiles is conceptually an encoder parallelisation tool as it splits the picture into independent coding regions of the same size (almost) of the picture. But this tool can be used also for other applications.

[0098] As these three high level available possible ways of partitioning of a picture can be used together, there is several modes for their usage. As defined in the current draft specifications of VVC, two modes of slices are defined. For the raster-scan slice mode, a slice contains a sequence of complete tiles in a tile raster scan of the picture. This mode in the current VVC specification is illustrated in FIG. **10**(*a*). As shown in this figure, the picture contains 18 by 12 luma CTUs is shown that is partitioned into 12 tiles and 3 raster-scan slices.

[0099] For the second one, the rectangular slice mode, a slice contains either a number of complete tiles that collectively from a rectangular region of the picture. This mode in the current VVC specification is illustrated in FIG. **10**(*b*). In this example, a picture with 18 by 12 luma CTUs is shown that is partitioned into 24 tiles and 9 rectangular slices.

[0100] FIG. **2** illustrates a data communication system in which one or more embodiments of the invention may be implemented. The data communication system comprises a transmission device, in this case a server **201**, which is operable to transmit data packets of a data stream to a receiving device, in this case a client terminal **202**, via a data communication network **200**. The data communication network **200** may be a Wide Area Network (WAN) or a Local Area Network (LAN). Such a network may be for example a wireless network (Wifi/802.11a or b or g), an Ethernet network, an Internet network or a mixed network composed of several different networks. In a particular embodiment of the invention the data communication system may be a digital television broadcast system in which the server **201** sends the same data content to multiple clients.

[0101] The data stream **204** provided by the server **201** may be composed of multimedia data representing video and audio data. Audio and video data streams may, in some embodiments of the invention, be captured by the server **201** using a microphone and a camera respectively.

[0102] In some embodiments data streams may be stored on the server **201** or received by the server **201** from another data provider, or generated at the server **201**. The server **201** is provided with an encoder for encoding video and audio streams in particular to provide a compressed bitstream for transmission that is a more compact representation of the data presented as input to the encoder.

[0103] In order to obtain a better ratio of the quality of transmitted data to quantity of transmitted data, the compression of the video data may be for example in accordance with the HEVC format or H.264/AVC format.

[0104] The client **202** receives the transmitted bitstream and decodes the reconstructed bitstream to reproduce video images on a display device and the audio data by a loud speaker.

[0105] Although a streaming scenario is considered in the example of FIG. **2**, it will be appreciated that in some embodiments of the invention the data communication between an encoder and a decoder may be performed using for

example a media storage device such as an optical disc.

[0106] In one or more embodiments of the invention a video image is transmitted with data representative of compensation offsets for application to reconstructed pixels of the image to provide filtered pixels in a final image.

[0107] FIG. **3** schematically illustrates a processing device **300** configured to implement at least one embodiment of the present invention. The processing device **300** may be a device such as a micro-computer, a workstation or a light portable device. The device **300** comprises a communication bus **313** connected to: [0108] a central processing unit **311**, such as a microprocessor, denoted CPU; [0109] a read only memory **306**, denoted ROM, for storing computer programs for implementing the invention; [0110] a random access memory **312**, denoted RAM, for storing the executable code of the method of embodiments of the invention as well as the registers adapted to record variables and parameters necessary for implementing the method of encoding a sequence of digital images and/or the method of decoding a bitstream according to embodiments of the invention; and [0111] a communication interface **302** connected to a communication network **303** over which digital data to be processed are transmitted or received

[0112] Optionally, the apparatus **300** may also include the following components: [0113] a data storage means **304** such as a hard disk, for storing computer programs for implementing methods of one or more embodiments of the invention and data used or produced during the implementation of one or more embodiments of the invention; [0114] a disk drive **305** for a disk **306**, the disk drive being adapted to read data from the disk **306** or to write data onto said disk; [0115] a screen **309** for displaying data and/or serving as a graphical interface with the user, by means of a keyboard **310** or any other pointing means.

[0116] The apparatus **300** can be connected to various peripherals, such as for example a digital camera **320** or a microphone **308**, each being connected to an input/output card (not shown) so as to supply multimedia data to the apparatus **300**.

[0117] The communication bus provides communication and interoperability between the various elements included in the apparatus **300** or connected to it. The representation of the bus is not limiting and in particular the central processing unit is operable to communicate instructions to any element of the apparatus **300** directly or by means of another element of the apparatus **300**.

[0118] The disk **306** can be replaced by any information medium such as for example a compact disk (CD-ROM), rewritable or not, a ZIP disk or a memory card and, in general terms, by an information storage means that can be read by a microcomputer or by a microprocessor, integrated or not into the apparatus, possibly removable and adapted to store one or more programs whose execution enables the method of encoding a sequence of digital images and/or the method of decoding a bitstream according to the invention to be implemented.

[0119] The executable code may be stored either in read only memory **306**, on the hard disk **304** or on a removable digital medium such as for example a disk **306** as described previously. According to a variant, the executable code of the programs can be received by means of the communication network **303**, via the interface **302**, in order to be stored in one of the storage means of the apparatus **300** before being executed, such as the hard disk **304**.

[0120] The central processing unit **311** is adapted to control and direct the execution of the instructions or portions of software code of the program or programs according to the invention, instructions that are stored in one of the aforementioned storage means. On powering up, the program or programs that are stored in a non-volatile memory, for example on the hard disk **304** or in the read only memory **306**, are transferred into the random access memory **312**, which then contains the executable code of the program or programs, as well as registers for storing the variables and parameters necessary for implementing the invention.

[0121] In this embodiment, the apparatus is a programmable apparatus which uses software to implement the invention. However, alternatively, the present invention may be implemented in hardware (for example, in the form of an Application Specific Integrated Circuit or ASIC).

[0122] FIG. **4** illustrates a block diagram of an encoder according to at least one embodiment of the invention. The encoder is represented by connected modules, each module being adapted to implement, for example in the form of programming instructions to be executed by the CPU **311** of device **300**, at least one corresponding step of a method implementing at least one embodiment of encoding an image of a sequence of images according to one or more embodiments of the invention.

[0123] An original sequence of digital images i0 to in **401** is received as an input by the encoder **400**. Each digital image is represented by a set of samples, known as pixels.

[0124] A bitstream **410** is output by the encoder **400** after implementation of the encoding process. The bitstream **410** comprises a plurality of encoding units or slices, each slice comprising a slice header for transmitting encoding values of encoding parameters used to encode the slice and a slice body, comprising encoded video data.

[0125] The input digital images i0 to in **401** are divided into blocks of pixels by module **402**. The blocks correspond to image portions and may be of variable sizes (e.g. 4×4, 8×8, 16×16, 32×32, 64×64, 128×128 pixels and several rectangular block sizes can be also considered). A coding mode is selected for each input block. Two families of coding modes are provided: coding modes based on spatial prediction coding (Intra prediction), and coding modes based on temporal prediction (Inter coding, Merge, SKIP). The possible coding modes are tested.

[0126] Module **403** implements an Intra prediction process, in which the given block to be encoded is predicted by a predictor computed from pixels of the neighbourhood of said block to be encoded. An indication of the selected Intra predictor and the difference between the given block and its predictor is encoded to provide a residual if the Intra coding is selected.

[0127] Temporal prediction is implemented by motion estimation module **404** and motion compensation module **405**. Firstly, a reference image from among a set of reference images **416** is selected, and a portion of the reference image, also called reference area or image portion, which is the closest area to the given block to be encoded, is selected by the motion

estimation module **404**. Motion compensation module **405** then predicts the block to be encoded using the selected area. The difference between the selected reference area and the given block, also called a residual block, is computed by the motion compensation module **405**. The selected reference area is indicated by a motion vector.

[0128] Thus, in both cases (spatial and temporal prediction), a residual is computed by subtracting the prediction from the original block.

[0129] In the INTRA prediction implemented by module **403**, a prediction direction is encoded. In the temporal prediction, at least one motion vector is encoded. In the Inter prediction implemented by modules **404**, **405**, **416**, **418**, **417**, at least one motion vector or data for identifying such motion vector is encoded for the temporal prediction.

[0130] Information relative to the motion vector and the residual block is encoded if the Inter prediction is selected. To further reduce the bitrate, assuming that motion is homogeneous, the motion vector is encoded by difference with respect to a motion vector predictor. Motion vector predictors of a set of motion information predictors is obtained from the motion vectors field **418** by a motion vector prediction and coding module **417**.

[0131] The encoder **400** further comprises a selection module **406** for selection of the coding mode by applying an encoding cost criterion, such as a rate-distortion criterion. In order to further reduce redundancies a transform (such as DCT) is applied by transform module **407** to the residual block, the transformed data obtained is then quantized by quantization module **408** and entropy encoded by entropy encoding module **409**. Finally, the encoded residual block of the current block being encoded is inserted into the bitstream **410**.

[0132] The encoder **400** also performs decoding of the encoded image in order to produce a reference image for the motion estimation of the subsequent images. This enables the encoder and the decoder receiving the bitstream to have the same reference frames. The inverse quantization module **411** performs inverse quantization of the quantized data, followed by an inverse transform by reverse transform module **412**. The reverse intra prediction module **413** uses the prediction information to determine which predictor to use for a given block and the reverse motion compensation module **414** actually adds the residual obtained by module **412** to the reference area obtained from the set of reference images **416**.

[0133] Post filtering is then applied by module **415** to filter the reconstructed frame of pixels. In the embodiments of the invention an SAO loop filter is used in which compensation offsets are added to the pixel values of the reconstructed pixels of the reconstructed image

[0134] FIG. **5** illustrates a block diagram of a decoder **60** which may be used to receive data from an encoder according an embodiment of the invention. The decoder is represented by connected modules, each module being adapted to implement, for example in the form of programming instructions to be executed by the CPU **311** of device **300**, a corresponding step of a method implemented by the decoder **60**.

[0135] The decoder **60** receives a bitstream **61** comprising encoding units, each one being composed of a header containing information on encoding parameters and a body containing the encoded video data. The structure of the bitstream in VVC is described in more detail below with reference to FIG. **6**. As explained with respect to FIG. **4**, the encoded video data is entropy encoded, and the motion vector predictors' indexes are encoded, for a given block, on a predetermined number of bits. The received encoded video data is entropy decoded by module **62**. The residual data are then dequantized by module **63** and then a reverse transform is applied by module **64** to obtain pixel values.

[0136] The mode data indicating the coding mode are also entropy decoded and based on the mode, an INTRA type decoding or an INTER type decoding is performed on the encoded blocks of image data.

[0137] In the case of INTRA mode, an INTRA predictor is determined by intra reverse prediction module **65** based on the intra prediction mode specified in the bitstream.

[0138] If the mode is INTER, the motion prediction information is extracted from the bitstream so as to find the reference area used by the encoder. The motion prediction information is composed of the reference frame index and the motion vector residual. The motion vector predictor is added to the motion vector residual in order to obtain the motion vector by motion vector decoding module **70**.

[0139] Motion vector decoding module **70** applies motion vector decoding for each current block encoded by motion prediction. Once an index of the motion vector predictor, for the current block has been obtained the actual value of the motion vector associated with the current block can be decoded and used to apply reverse motion compensation by module **66**. The reference image portion indicated by the decoded motion vector is extracted from a reference image **68** to apply the reverse motion compensation **66**. The motion vector field data **71** is updated with the decoded motion vector in order to be used for the inverse prediction of subsequent decoded motion vectors.

[0140] Finally, a decoded block is obtained. Post filtering is applied by post filtering module **67**. A decoded video signal **69** is finally provided by the decoder **60**.

[0141] FIG. **6** illustrates the organisation of the bitstream in the exemplary coding system VVC as describe in JVET-Q2001-vD.

[0142] A bitstream **600** according to the VVC coding system is composed of an ordered sequence of syntax elements and coded data. The syntax elements and coded data are placed into Network Abstraction Layer (NAL) units **601-608**. There are different NAL unit types. The network abstraction layer provides the ability to encapsulate the bitstream into different protocols, like RTP/IP, standing for Real Time Protocol/Internet Protocol, ISO Base Media File Format, etc. The network abstraction layer also provides a framework for packet loss resilience.

[0143] NAL units are divided into Video Coding Layer (VCL) NAL units and non-VCL NAL units. The VCL NAL units contain the actual encoded video data. The non-VCL NAL units contain additional information. This additional information may be parameters needed for the decoding of the encoded video data or supplemental data that may enhance usability of the decoded video data. NAL units **606** correspond to slices and constitute the VCL NAL units of the bitstream.

[0144] Different NAL units **601-605** correspond to different parameter sets, these NAL units are non-VCL NAL units. The Decoder Parameter Set (DPS) NAL unit **301** contains parameters that are constant for a given decoding process. The Video Parameter Set (VPS) NAL unit **602** contains parameters defined for the whole video, and thus the whole bitstream. The DPS NAL unit may define parameters more static than the parameters in the VPS. In other words, the parameters of DPS change less frequently than the parameter of the VPS.

[0145] The Sequence Parameter Set (SPS) NAL unit **603** contains parameters defined for a video sequence. In particular, the SPS NAL unit may define the sub pictures layout and associated parameters of the video sequences. The parameters associated to each subpicture specifies the coding constraints applied to the subpicture. In particular, it comprises a flag indicating that the temporal prediction between subpictures is restricted to the data coming from the same subpicture. Another flag may enable or disable the loop filters across the subpicture boundaries.

[0146] The Picture Parameter Set (PPS) NAL unit **604**, PPS contains parameters defined for a picture or a group of pictures. The Adaptation Parameter Set (APS) NAL unit **605**, contains parameters for loop filters typically the Adaptive Loop Filter (ALF) or the reshaper model (or luma mapping with chroma scaling (LMCS) model) or the scaling matrices that are used at the slice level.

[0147] The syntax of the PPS as proposed in the current version of VVC comprises syntax elements that specifies the size of the picture in luma samples and also the partitioning of each picture in tiles and slices.

[0148] The PPS contains syntax elements that make it possible to determine the slices location in a frame. Since a subpicture forms a rectangular region in the frame, it is possible to determine the set of slices, the parts of tiles or the tiles that belong to a subpicture from the Parameter Sets NAL units. The PPS as with the APS have an ID mechanism to limit the amount of same PPS's transmitted.

[0149] The main difference between the PPS and Picture Header is it transmission, the PPS is generally transmitted for a group of pictures compared to the PH which is systematically transmitted for each Picture. Accordingly, the PPS compared to the PH contains parameters which can be constant for several picture.

[0150] The bitstream may also contain Supplemental Enhancement Information (SEI) NAL units (not represented in FIG. **6**). The periodicity of occurrence of these parameter sets in the bitstream is variable. A VPS that is defined for the whole bitstream may occur only once in the bitstream. To the contrary, an APS that is defined for a slice may occur once for each slice in each picture. Actually, different slices may rely on the same APS and thus there are generally fewer APS than slices in each picture. In particular, the APS is defined in the picture header. Yet, the ALF APS can be refined in the slice header.

[0151] The Access Unit Delimiter (AUD) NAL unit **607** separates two access units. An access unit is a set of NAL units which may comprise one or more coded pictures with the same decoding timestamp. This optional NAL unit contains only one syntax element in current VVC specification: pic_type, this syntax element. indicates that the slice_type values for all slices of the coded pictures in the AU. If pic_type is set equal to 0, the AU contain only Intra slice. If equal to 1, it contains P and I slices. If equal to 2 it contains B, P or Intra slice This NAL unit contains only one syntax element the pic-type.

TABLE-US-00001 TABLE 1 Syntax AUD Descriptor access_unit_delimiter_rbsp( ) {   pic_type u(3) rbsp_trailing_bits( ) }

In JVET-Q2001-vD the pic_type is defined as follow: [0152] "pic_type indicates that the slice_type values for all slices of the coded pictures in the AU containing the AU delimiter NAL unit are members of the set listed in Table 2 for the given value of pic_type. The value of pic_type shall be equal to 0, 1 or 2 in bitstreams conforming to this version of this Specification. Other values of pic_type are reserved for future use by ITU-T|ISO/IEC. Decoders conforming to this version of this Specification shall ignore reserved values of pic_type."

The rbsp_trailing_bits ( ) is a function which adds bits in order to be aligned to the end of a byte. So after, this function, the amount of bitstream parsed is an integer number of bytes.

TABLE-US-00002 TABLE 2 Interpretation of pic_type slice_type values that pic_type may be present in the AU 0 I 1 P, I 2 B, P, I

[0153] The PH NAL unit **608** is the Picture Header NAL unit which groups parameters common to a set of slices of one coded picture. The picture may refer to one or more APS to indicate the AFL parameters, reshaper model and the scaling matrices used by the slices of the Picture.

[0154] Each of the VCL NAL units **606** contains a slice. A slice may correspond to the whole picture or sub picture, a single tile or a plurality of tiles or a fraction of a tile. For example the slice of the FIG. **3** contains several tiles **620**. A slice is composed of a slice header **610** and a raw byte sequence payload, RBSP **611** that contains the coded pixels data encoded as coded blocks **640**.

[0155] The syntax of the PPS as proposed in the current version of VVC comprises syntax elements that specifies the size of the picture in luma samples and also the partitioning of each picture in tiles and slices.

[0156] The PPS contains syntax elements that make it possible to determine the slices location in a frame. Since a subpicture forms a rectangular region in the frame, it is possible to determine the set of slices, the parts of tiles or the tiles that belong to a subpicture from the Parameter Sets NAL units.

NAL Unit Slice

[0157] The NAL unit slice layer contains the slice header and the slice data as illustrated in Table 3.

TABLE-US-00003 TABLE 3 Slice layer syntax Descriptor slice_layer_rbsp( ) {   slice_header( )   slice_data( )   rbsp_slice_trailing_bits( ) }

APS

[0158] The Adaptation Parameter Set (APS) NAL unit **605**, is defined in Table 4 showing the syntax elements.

[0159] As depicted in table Table 4, there are 3 possible types of APS given by the aps_params_type syntax element: [0160]

ALF_AP: for the ALF parameters [0161] LMCS_APS for the LMCS parameters [0162] SCALING_APS for Scaling list relative parameters

TABLE-US-00004 TABLE 4 Adaptation parameter set syntax Descriptor adaptation_parameter_set_rbsp( ) { adaptation_parameter_setid u(5) aps_params_type u(3) if(aps_params_type = = ALF_APS) alf_data( ) else if(aps_params_type = = LMCS_APS) lmcs_data( ) else if(aps_params-type = = SCALING_APS) scaling_list_data( ) aps_extension_flag u(1) if(aps_extension_flag) while(more_rbsp_data( )) aps_extension_data_flag u(1) rbsp_trailing_bits( ) } [0163] These three types of APS parameters are discussed in turn below

ALF APS

[0164] The ALF parameters are described in Adaptive loop filter data syntax elements (Table 5). First, four flags are dedicated to specify whether or not the ALF filters are transmitted for Luma and/or for Chroma and if the CC-ALF (Cross Component Adaptive Loop Filtering) is enabled for Cb component and Cr component. If the Luma filter flag is enabled, another flag is decoded to know if the clip values are signalled (alf_luma_clip_flag). Then the number of filters signalled is decoded using the alf_luma_num_filters_signalled_minus1 syntax element. If needed, the element representing the ALF coefficients delta "alf_luma_coeff_delta_idx" is decoded for each enabled filter. Then absolute value and the sign for each coefficient of each filter are decoded.

[0165] If the alf_luma_clip_flag is enabled, the clip index for each coefficient of each enabled filter is decoded.

[0166] In the same way, the ALF chroma coefficients are decoded if needed.

[0167] If CC-ALF is enabled for Cr or Cb the number of filter are decoded (alf_cc_cb_filters_signalled_minuslor alf_cc_cr_filters_signalled_minus1) and the related coefficients are decoded (alf_cc_cb_mapped_coeff_abs and alf_cc_cb_coeff_sign or respectively alf_cc_cr_mapped_coeff_abs and alf_cc_cr_coeff_sign)

TABLE-US-00005 TABLE 5 Adaptive loop filter data syntax Descriptor alf_data( ) { alf_luma_filter_signal_flag u(1) alf_chroma_filter_signal_flag u(1) alf_cc_cb_filter_signal_flag u(1) alf_cc_cr_filtcr_signal_flag u(1) if(alf_luma_filter_signal_flag) { alf_luma_clip_flag u(1) alf_luma_num_filters_signalled_minus1 ue(v) if(alf_luma_num_filters_signalled_minus1 > 0) for(filtIdx = 0; filtIdx < NumAlfFilters; filtIdx++) alf_luma_coeff_delta_idx[filtIdx] u(v) for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) for(j = 0; j < 12; j++){ alf_luma_coeff_abs[sfIdx][j] ue(v) if(alf_luma_coeff_abs[sfIdx][j]) alf_luma_coeff_sign[sfIdx][j] u(1) } if(alf_luma_clip_flag) for(sfIdx = 0; sfIdx <= alf_luma_num_filters_signalled_minus1; sfIdx++) for(j = 0; j < 12; j++) alf_luma_clip_idx[sfIdx][j] u(2) } if(alf_chroma_filter_signal_flag) { alf_chroma_clip_flag u(1) alf_chroma_num_alt_filters_minus1 ue(v) for(altIdx = 0; altIdx <= alf_chroma_num_alt_filters_minus1; altIdx++) { for(j = 0; j < 6; j++) { alf_chroma_coeff_abs[altIdx][j] ue(v) if(alf_chroma_coeff_abs[altIdx][j] > 0) alf_chroma_coeff_sign[altIdx][j] u(1) } if(alf_chroma_clip_flag) for(j = 0; j < 6; j++) alf_chroma_clip_idx[altIdx][j] u(2) } } if(alf_cc_cb_filter_signal_flag) { alf_cc_cb_filters_signalled_minusl ue(v) for(k = 0; k < alf_cc_cb_filters_signalled_minus1 + 1; k++) { for(j = 0; j < 7; j++) { alf_cc_cb_mapped_coeff_abs[k][j] u(3) if(alf_cc_cb_mapped_coeff_abs[k][j]) alf_cc_cb_coeff_sign[k] [j] u(1) } } } if(alf_cc_cr_filter_signal_flag) { alf_cc_cr_filters_signalled_minus1 ue(v) for(k = 0; k < alf_cc_cr_filters_signalled_minus1 + 1; k++) { for(j = 0; j < 7; j++) { alf_cc_cr_mapped_coeff_abs[k][j] u(3) if(alf_cc_cr_mapped_coeff_abs[k][j]) alf_cc_cr_coeff_sign[k][j] u(1) } } } }

LMCS Syntax Elements for Both Luma Mapping and Chroma Scaling

[0168] The Table 6 below gives all the LMCS syntax elements which are coded in the adaptation parameter set (APS) syntax structure when the aps_params_type parameter is set to 1 (LMCS_APS). Up to four LMCS APS's can be used in a coded video sequence, however, only a single LMCS APS can be used for a given picture.

[0169] These parameters are used to build the forward and inverse mapping functions for Luma and the scaling function for Chroma.

TABLE-US-00006 TABLE 6 Luma mapping with chroma scaling data syntax Descriptor lmcs_data ( ) { lmcs_min_bin_idx ue(v) lmcs_delta_max_bin_idx ue(v) lmcs_delta_cw_prec_minus1 ue(v) for(i = lmcs_min_bin_idx; i <= LmcsMaxBinIdx; i++) { lmcs_delta_abs_cw[i] u(v) if(lmcs_delta_abs_cw[i]) > 0) lmcs_delta_sign_cw_flag[i] u(1) } lmcs_delta_abs_crs u(3) if(lmcs_delta_abs_crs) > 0) lmcs_delta_sign_crs_flag u(1) }

Scaling List APS

[0170] The scaling list offers the possibility to update the quantization matrix used for quantification. In VVC this scaling matrix is signalled in the APS as described in Scaling list data syntax elements (Table 7 Scaling list data syntax). The first syntax element specifies if the scaling matrix is used for the LFNST (Low Frequency Non-Separable Transform) tool based on the flag scaling_matrix_for_Ifnst_disabled_flag. The second one is specified if the scaling list are used for Chroma components (scaling_list_chroma_present_flag). Then the syntax elements needed to build the scaling matrix are decoded (scaling_list_copy_mode_flag, scaling_list_pred_mode_flag, scaling_list_pred_id_delta, scaling_list_dc_coef, scaling_list_delta_coef).

TABLE-US-00007 TABLE 7 Scaling list data syntax Descriptor scaling_list_data( ) { scaling_matrix_for_lfnst_disabled_flag u(1) scaling_list_chroma_present_flag u(1) for(id = 0; id < 28; id ++) matrixSize = (id< 2) ? 2 : ((id < 8) ? 4 : 8) if(scaling_list_chroma_present_flag ‖ (id % 3 = = 2) ‖ (id = = 27)) {

```
            scaling_list_copy_mode_flag[ id ] u(1)          if( !scaling_list_copy_mode_flag[ id ] )
scaling_list_pred_mode_flag[ id ] u(1)          if( ( scaling_list_copy_mode_flag[ id ] scaling_list_pred_mode_flag[ id ] )
&& id != 0 && id != 2 && id != 8 )          scaling_list_pred_id_delta[ id ] ue(v)          if(
!scaling_list_copy_mode_flag[ id ] ) {          nextCoef = 0          if( id > 13 ) {          scaling_list_dc_coef|[ id
− 14 ] se(v)          nextCoef += scaling_list_dc_coef[ id − 14 ]          }          for( i = 0; i < matrixSize *
matrixSize; i++) {          x = DiagScanOrder[ 3 ] [ 3 ] [ i ] [ 0 ]          y = DiagScanOrder[ 3 ] [ 3 ] [ i ] [ 1 ]
          if( !(id > 25 && x >= 4 && y >= 4 ) ) {          scaling_list_delta_coef[ id ] [ i ] se(v)
nextCoef += scaling_list_delta_coef[ id ] [ i ]          }          ScalingList[ id ] [ i ] = nextCoef          }
          }          }          } }
```

Picture Header

[0171] The picture header is transmitted at the beginning of each picture before the other Slice Data. This is very large compared to the previous headers in the previous drafts of the standard. A complete description of all these parameters can be found in JVET-Q2001-vD. Table 10 shows these parameters in the current picture header decoding syntax.

[0172] The related syntax elements which can be decoded are related to: [0173] the usage of this picture, reference frame or not [0174] The type of picture [0175] output frame. [0176] The number of the Picture [0177] subpicture usage if needed [0178] reference picture lists if needed [0179] colour plane if needed [0180] partitioning update if overriding flag is enabled [0181] delta QP parameters if needed [0182] Motion information parameters if needed [0183] ALF parameters if needed [0184] SAO parameters if needed [0185] quantification parameters if needed [0186] LMCS parameters if needed [0187] Scaling list parameters if needed [0188] picture header extension if needed [0189] Etc. . . .

Picture "Type"

[0190] The first flag is the gdr_or_irap_pic_flag which indicates if the current picture is a resynchronisation picture (IRAP or GDR). If this flag is true, the gdr_pic_flag is decoded to know if the current picture is an IRAP or a GDR picture.

[0191] Then the ph_inter_slice_allowed_flag is decoded to identify that the Inter slice is allowed.

[0192] When they are allowed, the flag ph_intra_slice_allowed_flag is decoded to know if the Intra slice are allowed for the current picture.

[0193] Then the non_reference_picture flag, the ph_pic_parameter_set_id indicating the PPS ID and the picture order count ph_pic_order_cnt_lsb are decoded. The picture order count gives the number of the current picture.

[0194] If the picture is a GDR or an IRAP picture, the flag no_output_of_prior_pics_flag is decoded.

[0195] And if the picture is a GDR the recovery_poc_cnt is decoded. Then ph_poc_msb_present_flag and poc_msb_val are decoded if needed.

ALF

[0196] After these parameters describing important information on the current picture, the set of ALF APS id syntax elements are decoded if ALF is enabled at SPS level and if ALF is enabled at picture header level. ALF is enabled at SPS level thanks to the sps_alf_enabled_flag flag. And ALF signalling is enabled at picture header level thanks to the alf_info_in_ph_flag equal to 1 otherwise (alf_info_in_ph_flag equal to 0) ALF is signalled at slice level. The alf_info_in_ph_flag is defined as the following: [0197] "alf_info_in_ph_flag equal to 1 specifies that ALF information is present in the PH syntax structure and not present in slice headers referring to the PPS that do not contain a PH syntax structure. alf_info_in_ph_flag equal to 0 specifies that ALF information is not present in the PH syntax structure and may be present in slice headers referring to the PPS that do not contain a PH syntax structure."

[0198] First the ph_alf_enabled_present_flag is decoded to determine whether or not if the ph_alf_enabled_flag should be decoded. If the ph_alf_enabled_flag is enabled, ALF is enabled for all slices of the current picture.

[0199] If ALF is enabled, the amount of ALF APS id for luma is decoded using the pic_num_alf_aps_ids_luma syntax element. For each APS id, the APS id value for luma is decoded "ph_alf_aps_id_luma".

[0200] For chroma the syntax element, ph_alf_chroma_idc is decoded to determine whether or not ALF is enabled for Chroma, for Cr only, or for Cb only. If it is enabled, the value of the APS ID for Chroma is decoded using the ph_alf_aps_id_chroma syntax element. In the way the APS ID for CC-ALF method are decoded if needed for Cb and/or CR components

LMCS

[0201] The set of LMCS APS ID syntax elements is then decoded if LMCS was enabled at SPS level. First the ph_lmcs_enabled_flag is decoded to determine whether or not LMCS is enabled for the current picture. If LMCS is enabled, the ID value is decoded ph_lmcs_aps_id. For Chorma only the ph_chroma_residual_scale_flag is decoded to enable or disable the method for Chroma.

Scaling List

[0202] The set of scaling list APS ID is then decoded if the scaling list is enabled at SPS level. The ph_scaling_list_present_flag is decoded to determine whether or not the scaling matrix is enabled for the current picture. And the value of the APS ID, ph_scaling_list_aps_id, is then decoded.

Subpicture

[0203] The Subpicture parameters are enabled when they are enabled at SPS and if the subpicture id signalling is disabled. It also contains some information on virtual boundaries. For the sub picture parameters eight syntax elements are defined: [0204] ph_virtual_boundaries_present_flag [0205] ph_num_ver_virtual_boundaries [0206] ph_virtual_boundaries_pos_x[i] [0207] ph_num_hor_virtual_boundaries [0208] ph_virtual_boundaries_pos_y[i]

Output Flag

[0209] These subpicture parameters are followed by the pic_output_flag if present.

Reference Picture Lists

[0210] If the reference picture lists are signalled in the picture header (thanks to rpl_info_in_ph_flag equal to 1), then the parameters for the reference picture lists are decoded ref_pic_lists ( ) it contains the following syntax elements: [0211] rpl_sps_flag[ ] [0212] rpl_idx[ ] [0213] poc_lsb_lt[ ][ ] [0214] delta_poc_msb_present_flag[ ][ ] [0215] delta_poc_msb_cycle_lt[ ][ ]

[0216] And it is defined in following syntax table:

TABLE-US-00008 TABLE 8 Reference picture lists syntax Descriptor ref_pic_lists( ) {   for(i = 0; i < 2; i++) {   if(num_ref_pic_lists_in_sps[i] > 0 &&   (i = = 0 ‖ (i = = 1 && rpl1_idx_present_flag)))   rpl_sps_flag[i] u(1)   if(rpl_sps_flag[i]) {   if(num_ref_pic_lists_in_sps[i] > 1 &&   (i = = 0 ‖ (i = = 1 && rpl1_idx_present_flag)))   rpl_idx[i] u(v)   } else   ref_pic_list_struct(i, num_ref_pic_lists_in_sps[i])   for(j = 0; j < NumLtrpEntries[i][RplsIdx[i]]; j++) {   if(ltrp in header flag[i] [Rplsldx[i]])   poc_lsb_lt[i][j] u(v)   delta_poc_msb_present_flag[i][j] u(1)   if(delta_poc_msb_present_flag[i][j])   delta_poc_msb_cycle_lt[i][j] ue(v)   }   } }

Partitioning

[0217] The set of partitioning parameters is decoded if needed and contains the following syntax elements: [0218] partition_constraints_override_flag [0219] ph_log 2_diff_min_qt_min_cb_intra_slice_luma [0220] ph_max_mtt_hierarchy_depth_intra_slice_luma [0221] ph_log 2_diff_max_bt_min_qt_intra_slice_luma [0222] ph_log 2_diff_max_tt_min_qt_intra_slice_luma [0223] ph_log 2_diff_min_qt_min_cb_intra_slice_chroma [0224] ph_max_mtt_hierarchy_depth_intra_slice_chroma [0225] ph_log 2_diff_max_bt_min_qt_intra_slice_chroma [0226] ph_log 2_diff_max_tt_min_qt_intra_slice_chroma [0227] ph_log 2_diff_min_qt_min_cb_inter_slice [0228] ph_max_mtt_hierarchy_depth_inter_slice [0229] ph_log 2_diff_max_bt_min_qt_inter_slice [0230] ph_log 2_diff_max_tt_min_qt_inter_slice

Weighted Prediction

[0231] The weighted prediction parameters pred_weight_table ( ) are decoded if the weighted prediction method is enabled at PPS level and if the weighted prediction parameters are signalled in the picture header (wp_info_in_ph_flag equal to 1).

[0232] The pred_weight_table ( ) contains the weighted prediction parameters for List L0 and for list L1 when bi-prediction weighted prediction is enabled. When the weighted prediction parameters are transmitted in the picture header the number of weights for each list are explicitly transmitted as depicted in the pred_weight_table ( ) syntax table Table 9.

TABLE-US-00009 TABLE 9 Weighted prediction parameters syntax Descriptor pred_weight_table( ) {   luma_log2_weight_denom ue(v)   if(ChromaArrayType != 0)   delta_chroma_log2_weight_denom se(v)   if(wp_info_in_ph_flag)   num_10_weights ue(v)   for(i = 0; i < NumWeightsL0; i++)   luma_weight_10_flag[i] u(1)   if(ChromaArrayType != 0)   for(i = 0; i < NumWeightsL0; i++)   chroma_weight_10_flag[i] u(1)   for(i = 0; i < NumWeightsL0; i++) {   if(luma_weight_10_flag[i]) {   delta_luma_weight_10[i] se(v)   luma_offset_10[i] se(v)   }   if(chroma_weight_10_flag[i])   for(j = 0; j < 2; j++) {   delta_chroma_weight_10[i] [j] se(v)   delta_chroma_offset_10[i] [j] se(v)   }   }   if(pps_weighted_bipred_flag && wp_info_in_ph_flag)   num_l1_weights ue(v)   for(i = 0; i < NumWeightsL1; i++)   luma_weight_l1_flag[i] u(1)   if(ChromaArrayType != 0)   for(i = 0; i < NumWeightsL1; i++)   chroma_weight_l1_flag[i] u(1)   for(i = 0; i < NumWeightsL1; i++) {   if(luma_weight_l1_flag[i]) {   delta_luma_weight_l1[i] se(v)   luma_offset_l1[i] se(v)   }   if(chroma_weight_l1_flag[i])   for(j = 0; j < 2; j++) {   delta_chroma_weight_l1[i][j] se(v)   delta_chroma_offset_l1[i][j] se(v)   }   } }

Delta QP

[0233] When the picture is Intra the ph_cu_qp_delta_subdiv_intra_slice and the ph_cu_chroma_qp_offset_subdiv_intra_slice are decoded if needed. And if Inter slice is allowed the ph_cu_qp_delta_subdiv_inter_slice and the ph_cu_chroma_qp_offset_subdiv_inter_slice are decoded if needed. Finally, the picture header extension syntax elements are decoded if needed.

[0234] All parameters alf_info_in_ph_flag, rpl_info_in_ph_flag, qp_delta_info_in_ph_flag sao_info_in_ph_flag, dbf_info_in_ph_flag, wp_info_in_ph_flag are signalled in the PPS.

TABLE-US-00010 TABLE 10 Picture header structure Descriptor picture_header_structure( ) {   gdr_or_irap_pic_flag u(1)   if(gdr_or_irap_pic_flag)   gdr_pic_flag u(1)   ph_inter_slice_allowed_flag u(1)   if(ph_inter_slice_allowed_flag)   ph_intra_slice_allowed_flag u(1)   non_reference_picture_flag u(1)   ph_pic_parameter_set_id ue(v)   ph_pic_order_cnt_lsb u(v)   if(gdr_or_irap_pic_flag)   no_output_of_prior_pics_flag u(1)   if(gdr_pic_flag)   recovery_poc_cnt ue(v)   for(i = 0; i < NumExtraPhBits; i++)   ph_extra_bit[i] u(1)   if(sps_poc_msb_flag) {   ph_poc_msb_present_flag u(1)   if(ph_poc_msb_present_flag)   poc_msb_val u(v)   }   if(sps_alf_enabled_flag && alf_info_in_ph_flag) {   ph_alf_enabled_flag u(1)   if(ph_alf_enabled_flag) {   ph_num_alf_aps_ids_luma u(3)   for(i = 0; i < ph_num_alf_aps_ids_luma; i++)   ph_alf_aps_id_luma[i] u(3)   if(ChromaArrayType != 0)   ph_alf_chroma_idc u(2)   if(ph_alf_chroma_idc > 0)   ph_alf_aps_id_chroma u(3)   if(sps_ccalf_enabled_flag) {   ph_cc_alf_cb_enabled_flag u(1)   if(ph_cc_alf_cb_enabled_flag)   ph_cc_alf_cb_aps_id u(3)   ph_cc_alf_cr_enabled_flag u(1)   if(ph_cc_alf_cr_enabled_flag)   ph_cc_alf_cr_aps_id u(3)   }   }   }   if(sps_lmcs_enabled_flag) {   ph_lmcs_enabled_flag u(1)   if(ph_lmcs_enabled_flag) {   ph_lmcs_aps_id u(2)   if(ChromaArrayType != 0)   ph_chroma_residual_scale_flag u(1)   }   }   if(sps_scaling_list_enabled_flag) {

ph_scaling_list_present_flag u(1)   if(ph_scaling_list_present_flag)   ph_scaling_list_aps_id u(3)   }   if(sps_virtual_boundaries_enabled_flag && !sps_virtual_boundaries_present_flag) {   ph_virtual_boundaries_present_flag u(1)   if(ph_virtual_boundaries_present_flag) {   ph_num_ver_virtual_boundaries u(2)   for(i = 0; i < ph_num_ver_virtual_boundaries; i++)   ph_virtual_boundaries_pos_x[i] u(13)   ph_num_hor_virtual_boundaries u(2)   for(i = 0; i < ph_num_hor_virtual_boundaries; i++)   ph_virtual_boundaries_pos_y[i] u(13)   }   }   if(output_flag_present_flag)   pic_output_flag u(1)   if(rpl_info_in_ph_flag)   ref_pic_lists( )   if(partition_constraints_override_enabled_flag)   partition_constraints_override_flag u(1)   if(ph_intra_slice_allowed_flag) {   if(partition_constraints_override_flag) {   ph_log2_diff_min_qt_min_cb_intra_slice_luma ue(v)   ph_max_mtt_hierarchy_depth_intra_slice_luma ue(v)   if(ph_max_mtt_hierarchy_depth_intra_slice_luma != 0) {   ph_log2_diff_max_bt_min_qt_intra_slice_luma ue(v)   ph_log2_diff_max_tt_min_qt_intra_slice_luma ue(v)   }   if(qtbtt_dual_tree_intra_flag) {   ph_log2_diff_min_qt_min_cb_intra_slice_chroma ue(v)   ph_max_mtt_hierarchy_depth_intra_slice_chroma ue(v)   if(ph_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {   ph_log2_diff_max_bt_min_qt_intra_slice_chroma ue(v)   ph_log2_diff_max_tt_min_qt_intra_slice_chroma ue(v)   }   }   }   if(cu_qp_delta_enabled_flag)   ph_cu_qp_delta_subdiv_intra_slice ue(v)   if(pps_cu_chroma_qp_offset_list_enabled_flag)   ph_cu_chroma_qp_offset_subdiv_intra_slice ue(v)   }   if(ph_inter_slice_allowed_flag) {   if(partition_constraints_override_flag) {   ph_log2_diff_min_qt_min_cb_inter_slice ue(v)   ph_max_mtt_hierarchy_depth_inter_slice ue(v)   if(ph_max_mtt_hierarchy_depth_inter_slice != 0) {   ph_log2_diff_max_bt_min_qt_inter_slice ue(v)   ph_log2_diff_max_tt_min_qt_inter_slice ue(v)   }   }   if(cu_qp_delta_enabled_flag)   ph_cu_qp_delta_subdiv_inter_slice ue(v)   if(pps_cu_chroma_qp_offset_list_enabled_flag)   ph_cu_chroma_qp_offset_subdiv_inter_slice ue(v)   if(sps_temporal_mvp_enabled_flag) {   ph_temporal_mvp_enabled_flag u(1)   if(ph_temporal_mvp_enabled_flag && rpl_info_in_ph_flag) {   ph_collocated_from_l0_flag u(1)   if((ph_collocated_from_l0_flag &&   num_ref_entries[0][RplsIdx[0]] > 1) | | (!ph_collocated_from_l0_flag &&   num_ref_entries[1][RplsIdx[1]] > 1))   ph_collocated_ref_idx ue(v)   }   }   mvd_l1_zero_flag u(1)   if(sps_fpel_mmvd_enabled_flag)   ph_fpel_mmvd_enabled_flag u(1)   if(sps_bdof_pic_present_flag)   ph_disable_bdof_flag u(1)   if(sps_dmvr_pic_present_flag)   ph_disable_dmvr_flag u(1)   if(sps_prof_pic_present_flag)   ph_disable_prof_flag u(1)   if((pps_weighted_pred_flag | | pps_weighted_bipred_flag) && wp_info_in_ph_flag)   pred_weight_table( )   }   if(qp_delta_info_in_ph_flag)   ph_qp_delta se(v)   if(sps_joint_cbcr_enabled_flag)   ph_joint_cbcr_sign_flag u(1)   if(sps_sao_enabled_flag && sao_info_in_ph_flag) {   ph_sao_luma_enabled_flag u(1)   if(ChromaArrayType != 0)   ph_sao_chroma_enabled_flag u(1)   }   if(sps_dep_quant_enabled_flag)   ph_dep_quant_enabled_flag u(1)   if(sps_sign_data_hiding_enabled_flag && !ph_dep_quant_enabled_flag)   pic_sign_data_hiding_enabled_flag u(1)   if(deblocking_filter_override_enabled_flag && dbf_info_in_ph_flag) {   ph_deblocking_filter_override_flag u(1)   if(ph_deblocking_filter_override_flag) {   ph_deblocking_filter_disabled_flag u(1)   if(!ph_deblocking_filter_disabled_flag) {   ph_beta_offset_div2 se(v)   ph_tc_offset_div2 se(v)   ph_cb_beta_offset_div2 se(v)   ph_cb_tc_offset_div2 se(v)   ph_cr_beta_offset_div2 se(v)   ph_cr_tc_offset_div2 se(v)   }   }   }   if(picture_header_extension_present_flag) {   ph_extension_length ue(v)   for(i = 0; i < ph_extension_length; i++)   ph_extension_data_byte[i] u(8)   } }

Slice Header

[0235] The Slice header is transmitted at the beginning of each slice. The slice header contains about 65 syntax elements. This is very large compared to the previous slice header in earlier video coding standards. A complete description of all the slice header parameters can be found in JVET-Q2001-vD. Table 11 shows these parameters in a current slice header decoding syntax.

TABLE-US-00011 TABLE 11 Partial Slice header Descriptor slice_header( ) {   picture_header_in_slice_header_flag u(1)   if(picture_header_in_slice_header_flag)   picture_header_structure( )   if(subpic_info_present_flag)   slice_subpic_id u(v)   if((rect_slice_flag && NumSlicesInSubpic[CurrSubpicIdx] > 1) | | (!rect_slice_flag && NumTilesInPic > 1))   slice_address u(v)   for(i = 0; i < NumExtraShBits; i++)   sh_extra_bit[i] u(1)   if(!rect_slice_flag && NumTilesInPic > 1)   num_tiles_in_slice_minus1 ue(v)   if(ph_inter_slice_allowed_flag)   slice_type ue(v)   if(sps_alf_enabled_flag && !alf_info_in_ph_flag) {   slice_alf_enabled_flag u(1)   if(slice_alf_enabled_flag) {   slice_num_alf_aps_ids_luma u(3)   for(i = 0; i < slice_num_alf_aps_ids_luma; i++)   slice_alf_aps_id_luma[i] u(3)   if(ChromaArrayType != 0)   slice_alf_chroma_idc u(2)   if(slice_alf_chroma_idc)   slice_alf_aps_id_chroma u(3)   if(sps_ccalf_enabled_flag) {   slice_cc_alf_cb_enabled_flag u(1)   if(slice_cc_alf_cb_enabled_flag)   slice_cc_alf_cb_aps_id u(3)   slice_cc_alf_cr_enabled_flag u(1)   if(slice_cc_alf_cr_enabled_flag)   slice_cc_alf_cr_aps_id u(3)   }   }   }   if(separate_colour_plane_flag = = 1)   colour_plane_id u(2)   if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) | | sps_idr_rpl_present_flag))   ref_pic_lists( )   if((rpl_info_in_ph_flag | | ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) | | sps_idr_rpl_present_flag)) &&   ((slice_type != I && num_ref_entries[0][RplsIdx[0]] > 1) | | (slice_type = = B && num_ref_entries[1][RplsIdx[1]] > 1))) {   num_ref_idx_active_override_flag u(1)   if(num_ref_idx_active_override_flag)   for(i = 0; i < (slice_type = = B ? 2: 1); i++)   if(num_ref_entries[i][

RplsIdx[i]] > 1)   num_ref_idx_active_minus1[i] ue(v)   }   if(slice_type != I) {   if(
cabac_init_present_flag)   cabac_init_flag u(1)   if(ph_temporal_mvp_enabled_flag && !rpl_info_in_ph_flag) {
        if(slice_type = = B)   slice_collocated_from_l0_flag u(1)   if((slice_collocated_from_l0_flag &&
NumRefIdxActive[0] > 1) || (! slice_collocated_from_l0_flag && NumRefIdxActive[1] > 1))
slice_collocated_ref_idx ue(v)   }   if(!wp_info_in_ph_flag && ((pps_weighted_pred_flag && slice_type = = P) ||
        (pps_weighted_bipred_flag && slice_type = = B)))   pred_weight_table( )   }   if(
!qp_delta_info_in_ph_flag)   slice_qp_delta se(v)   if(pps_slice_chroma_qp_offsets_present_flag) {
slice_cb_qp_offset se(v)   slice_cr_qp_offset se(v)   if(sps_joint_cbcr_enabled_flag)
slice_joint_cbcr_qp_offset se(v)   }   if(pps_cu_chroma_qp_offset_list_enabled_flag)
cu_chroma_qp_offset_enabled_flag u(1)   if(sps_sao_enabled_flag && !sao_info_in_ph_flag) {   slice_sao_luma_flag
u(1)   if(ChromaArrayType != 0)   slice_sao_chroma_flag u(1)   }   if(
deblocking_filter_override_enabled_flag && !dbf_info_in_ph_flag)   slice_deblocking_filter_override_flag u(1)   if(
slice_deblocking_filter_override_flag) {   slice_deblocking_filter_disabled_flag u(1)   if(
!slice_deblocking_filter_disabled_flag) {   slice_beta_offset_div2 se(v)   slice_tc_offset_div2 se(v)
slice_cb_beta_offset_div2 se(v)   slice_cb_tc_offset_div2 se(v)   slice_cr_beta_offset_div2 se(v)
slice_cr_tc_offset_div2 se(v)   }   }   slice_ts_residual_coding_disabled_flag u(1)   if(ph_lmcs_enabled_flag)
slice_lmcs_enabled_flag u(1)   if(ph_scaling_list_present_flag)   slice_scaling_list_present_flag u(1)   if(
NumEntryPoints > 0) {   offset_len_minus1 ue(v)   for(i = 0; i < NumEntryPoints; i++)
entry_point_offset_minus1[i] u(v)   }   if(slice_header_extension_present_flag) {   slice_header_extension_length
ue(v)   for(i = 0; i < slice_header_extension_length; i++)   slice_header_extension_data_byte[i] u(8)   }
byte_alignment( ) }

[0236] First the picture_header_in_slice_header_flag is decoded to know if the picture_header_structure ( ) is present in the slice header.

[0237] The slice_subpic_id if needed, is then decoded to determine the sub picture id of the current slice. Then the slice_address is decoded to determine the address of the current slice. The slice address is decoded if the current slice mode is the rectangular slice mode (rect_slice_flag equal to 1) and if the number of slices in the current subpicture is superior to 1. The slice address can be also decoded if the current slice mode is the raster scan mode (rect_slice_flag equal to 0) and if the number of tiles in the current picture is superior to 1 computed based on variables defined in the PPS.

[0238] The num_tiles_in_slice_minus is then decoded if the number of tiles in the current picture is greater than one and if the current slice mode is not the rectangular slice mode. In the current VVC draft specification, num_tiles_in_slice_minus/is defined as follows: [0239] "num_tiles_in_slice_minus1 plus 1, when present, specifies the number of tiles in the slice. The value of num_tiles_in_slice_minus1 shall be in the range of 0 to NumTilesInPic−1, inclusive."

[0240] Then the slice_type is decoded.

[0241] If ALF is enabled at SPS level (sps_alf_enabled_flag) and if ALF is signalled in the slice header (alf_info_in_ph_flag equal to 0), then ALF information is decoded. This includes a flag indicating that ALF is enabled for the current slice (slice_alf_enabled_flag). If it is enabled, the number of APS ALF ID for luma (slice_num_alf_aps_ids_luma) is decoded, then the APS ID are decoded (slice_alf_aps_id_luma[i]). Then the slice_alf_chroma_idc is decoded to know if ALF is enabled for the Chroma components and which chroma component it is enabled. Then the APS ID for Chroma is decoded slice_alf_aps_id_chroma if needed. In the same way, the slice_cc_alf_cb_enabled_flag is decoded, if needed, to know if the CC ALF method is enabled. IF CC ALF is enabled, the related APS ID for CR and/or CB are decoded if CC ALF is enabled for CR and/or CB.

[0242] If the colour planes are transmitted independently (separate_colour_plane_flag equals to 1) the colour_plane_id is decoded.

[0243] When the reference picture lists ref_pic_lists ( ) are not transmitted in the picture header (rpl_info_in_ph_flag equal to 0) and when the Nal unit is not an IDR or if the reference pictures lists are transmitted for IDR pictures (sps_idr_rpl_present_flag equals to 1) then the Reference picture lists parameters are decoded; these are similar to those in the picture header.

[0244] If the reference picture lists are transmitted in the picture header (rpl_info_in_ph_flag equal to 1) or the Nal unit is not an IDR or if the reference picture lists are transmitted for IDR pictures (sps_idr_rpl_present_flag equals to 1) and if the number of reference for at least one list is superior to 1, the override flag num_ref_idx_active_override_flag is decoded. This flag is defined as the following in the VVC draft specification: [0245] "mim ref_idx_active_override flag equal to 1 specifies that the syntax element num_ref_idx_active_minus1[0] is present for P and B slices and the syntax element num_ref_idx_active_minus1[1] is present for B slices. num_ref_idx_active_override flag equal to 0 specifies that the syntax elements num_ref_idx_active_minus1[0] and num_ref_idx_active_minus1[1] are not present. When not present, the value of mim ref_idx_active_override flag is inferred to be equal to 1."

[0246] If num_ref_idx_active_override_flag is enabled, the number of reference indexes num_ref_idx_active_minus1[i] for each list "i" are decoded if needed. The number of reference indexes overrides for the current list should be inferior or equal to the number of reference frames indexes signalled in the ref_pic_lists ( ). So the overriding reduces or not the maximum number of reference frames for each list.

[0247] When the slice type is not intra and if needed the cabac_init_flag is decoded. If the reference picture lists are transmitted in the slice header and come other conditions, the slice_collocated_from_10_flag and the slice_collocated_ref_idx are decoded. These data are related to the CABAC coding and the motion vector collocated.

[0248] In the same way, when the slice type is not Intra, the parameters of the weighted prediction pred_weight_table ( ) are

decoded.

[0249] The slice_qp_delta is decoded bif the delta QP information is transmitted in the slice header (qp_delta_info_in_ph_flag equal to 0). If needed the syntax elements, slice_cb_qp_offset, slice_cr_qp_offset, slice_joint_cbcr_qp_offset, cu_chroma_qp_offset_enabled_flag are decoded.

[0250] If the SAO information are transmitted in the slice header (sao_info_in_ph_flag equal to 0) and if it is enabled at SPS level (sps_sao_enabled_flag), the enabled flags for SAO are decoded for both luma and chroma: slice_sao_luma_flag, slice_sao_chroma_flag.

[0251] Then the deblocking filter parameters are decoded if they are signalled in the slice header (dbf_info_in_ph_flag equal to 0).

[0252] The flag slice_ts_residual_coding_disabled_flag is systematically decoded to know if the Transform Skip residual coding method is enabled for the current slice.

[0253] If LMCS was enabled in the picture header (ph_lmcs_enabled_flag equal 1), the flag slice_lmcs_enabled_flag is decoded. In the current VVC specification the slice_lmcs_enabled_flag is defined as the following: [0254] "slice_lmcs_enabled_flag equal to 1 specifies that luma mapping with chroma scaling is enabled for the current slice. slice_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not enabled for the current slice. When slice_lmcs_enabled_flag is not present, it is inferred to be equal to 0."

[0255] In the same way, if the scaling list was enabled in the picture header (phpic_scaling_list_presentenabled_flag equal 1), the flag slice_scaling_list_present_flag is decoded. In the current VVC specification, the slice_scaling_list_present_flag is defined as the following: [0256] "slice_scaling_list_present_flag equal to 1 specifies that the scaling list data used for the current slice is derived based on the scaling list data contained in the referenced scaling list APS with aps_params_type equal to SCALING_APS and adaptation_parameter_set_id equal to ph_scaling_list_aps_id. slice_scaling_list_present_flag equal to 0 specifies that the scaling list data used for the current picture is the default scaling list data derived specified in clause 7.4.3.21. When not present, the value of slice_scaling_list_present_flag is inferred to be equal to 0."

[0257] Then other parameters are decoded if needed.

Picture Header in the Slice Header

[0258] In a particular signalling way, the picture header **708** can be signalled inside the slice header **710** as depicted in the FIG. **7**. In that case there is no Nal unit containing only the picture header **608**. The units **701**, **702**, **703**, **704**, **705**, **706**, **707**, **720**, and **740** correspond to **601**, **602**, **603**, **604**, **605**, **606**, **606**, **620**, and **640** of FIG. **6** and, thus can be understood from the preceding description. This can be enabled in the slice header thanks to the flag picture_header_in_slice_header_flag. Moreover, when the picture header is signalled inside the slice header, the picture shall contain only one slice. So, there is always only one picture header per picture. Moreover, the flag picture_header_in_slice_header_flag shall have the same value for all pictures of a CLVS (Coded Layer Video Sequence). It means that all pictures between two IRAP including the first IRAP has only one slice per picture.

[0259] The flag picture_header_in_slice_header_flag is defined as the following: "picture_header_in_slice_header_flag equal to 1 specifies that the PH syntax structure is present in the slice header. picture_header_in_slice_header_flag equal to 0 specifies that the PH syntax structure is not present in the slice header.

[0260] It is a requirement of bitstream conformance that the value of picture_header_in_slice_header_flag shall be the same in all coded slices in a CLVS. When picture_header_in_slice_header_flag is equal to 1 for a coded slice, it is a requirement of bitstream conformance that no VCL NAL unit with nal_unit_type equal to PH_NUT shall be present in the CLVS.

[0261] When picture_header_in_slice_header_flag is equal to 0, all coded slices in the current picture shall have picture_header_in_slice_header_flag is equal to 0, and the current PU shall have a PH NAL unit.

[0262] The picture_header_structure ( ) contains syntax elements of the picture_rbsp ( ) except the stuffing bits rbsp_trailing_bits ( )"

Streaming Applications

[0263] Some streaming applications only extract certain parts of the bitstream. These extractions can be spatial (as the sub-picture) or temporal (a subpart of the video sequence). Then these extracted parts can be merged with other bitstreams. Some other reduce the frame rate by extracting only some frames. Generally, the main aim of these streaming applications is to use the maximum of the allowed bandwidth to produce the maximum quality to the end user.

[0264] In VVC, the APS ID numbering has been limited for frame rate reduction, in order that a new APS id number for a frame can't be used for a frame at an upper level in the temporal hierarchy. However, for streaming applications which extract parts of the bitstream the APS ID needs to be tracked to determine which APS should be keep for a sub part of the bitstream as the frame (as IRAP) don't reset the numbering of the APS ID.

LMCS (Luma Mapping with Chroma Scaling)

[0265] The Luma Mapping with Chroma scaling (LMCS) technique is a sample value conversion method applied on a block before applying the loop filters in a video decoder like VVC.

[0266] The LMCS can be divided into two sub-tools. The first one is applied on Luma block while the second sub-tool is applied on Chroma blocks as described below: [0267] 1) The first sub-tool is an in-loop mapping of the Luma component based on adaptive piecewise linear models. The in-loop mapping of the Luma component adjusts the dynamic range of the input signal by redistributing the codewords across the dynamic range to improve compression efficiency. Luma mapping makes use of a forward mapping function into the "mapped domain" and a corresponding inverse mapping function to come back in the "input domain". [0268] 2) The second sub-tool is related to the chroma components where a luma-dependent chroma residual scaling is applied. Chroma residual scaling is designed to compensate for the interaction between the luma signal and its corresponding chroma signals. Chroma residual scaling depends on the average value of top and/or left

reconstructed neighbouring luma samples of the current block.

[0269] Like most other tools in video coder like VVC, LMCS can be enabled/disabled at the sequence level using an SPS flag. Whether chroma residual scaling is enabled or not is also signalled at the slice level. If luma mapping is enabled, an additional flag is signalled to indicate if luma-dependent chroma residual scaling is enabled or not. When luma mapping is not used, luma-dependent chroma residual scaling is fully disabled. In addition, luma-dependent chroma residual scaling is always disabled for the chroma blocks whose size is less than or equal to 4.

[0270] FIG. **8** shows the principle of the LMCS as explained above for the Luma mapping sub-tool. The hatched blocks in FIG. **8** are the new LMCS functional blocks, including forward and inverse mapping of the luma signal. It is important to note that, when using LMCS, some decoding operations are applied in the "mapped domain". These operations are represented by blocks in dashed lines in this FIG. **8**. They typically correspond to the inverse quantization, the inverse transform, the luma intra prediction and the reconstruction step which consists in adding the luma prediction with the luma residual. Conversely, the solid line blocks in FIG. **8** indicate where the decoding process is applied in the original (i.e., non-mapped) domain and this includes the loop filtering such as deblocking, ALF, and SAO, the motion compensated prediction, and the storage of decoded pictures as reference pictures (DPB).

[0271] FIG. **9** shows a similar diagram as FIG. **8** but this time this is for the Chroma scaling sub-tool of the LMCS tool. The hatched block in FIG. **9** is the new LMCS functional block which includes the luma-dependent chroma scaling process. However, in Chroma, there are some important differences compared to the Luma case. Here only the inverse quantization and the inverse transform represented by block in dash lines are performed in the "mapped domain" for the Chroma samples. All the other steps of Intra Chroma prediction, motion compensation, loop filtering are performed in the original domain. As depicted in FIG. **9**, there is only a scaling process and there is no forward and inverse processing as for the Luma mapping.

Luma Mapping by Using Piece Wise Linear Model.

[0272] The luma mapping sub-tool is using a piecewise linear model. It means that the piecewise linear model separates the input signal dynamic range into 16 equal sub-ranges, and for each sub-range, its linear mapping parameters are expressed using the number of codewords assigned to that range.

Semantics for Luma Mapping

[0273] The syntax element lmcs_min_bin_idx specifies the minimum bin index used in the luma mapping with chroma scaling (LMCS) construction process. The value of lmcs_min_bin_idx shall be in the range of 0 to 15, inclusive.

[0274] The syntax element lmcs_delta_max_bin_idx specifies the delta value between 15 and the maximum bin index LmcsMaxBinIdx used in the luma mapping with chroma scaling construction process. The value of lmcs_delta_max_bin_idx shall be in the range of 0 to 15, inclusive. The value of LmcsMaxBinIdx is set equal to 15-lmcs_delta_max_bin_idx. The value of LmcsMaxBinIdx shall be greater than or equal to lmcs_min_bin_idx.

[0275] The syntax element lmcs_delta_cw_prec_minus/plus 1 specifies the number of bits used for the representation of the syntax lmcs_delta_abs_cw[i].

[0276] The syntax element lmcs_delta_abs_cw[i] specifies the absolute delta codeword value for the i.sub.th bin.

[0277] The syntax element lmcs_delta_sign_cw_flag[i] specifies the sign of the variable lmcsDeltaCW[i]. When lmcs_delta_sign_cw_flag[i] is not present, it is inferred to be equal to 0.

LMCS Intermediate Variables Computation for Luma Mapping

[0278] In order to apply the forward and inverse Luma mapping processes, some intermediate variables and data arrays are needed.

[0279] First of all, the variable OrgCW is derived as follows:

[00001] $OrgCW = (1 \ll BitDepth) / 16$

Then, the variable lmcsDeltaCW[i], with i=lmcs_min_bin_idx . . . LmcsMaxBinIdx, is computed as follows:

[00002] $lmcsDeltaCW[i] = (1 - 2 * lmcs\_delta\_sign\_cw\_flag[i]) * lmcs\_delta\_abs\_cw[i]$

The new variable lmcsCW[i] is derived as follows: [0280] For i=0 . . . lmcs_min_bin_idx-1, lmcsCW[i] is set equal 0.

[0281] For i=lmcs_min_bin_idx . . . LmcsMaxBinIdx, the following applies:

[00003] $lmcsCW[i] = OrgCW + lmcsDeltaCW[i]$

The value of lmcsCW[i] shall be in the range of (OrgCW>>3) to (OrgCW<<3-1), inclusive. [0282] For i=LmcsMaxBinIdx+1 . . . 15, lmcsCW[i] is set equal 0.

[0283] The variable InputPivot[i], with i=0 . . . 16, is derived as follows:

[00004] $InputPivot[i] = i * OrgCW$

[0284] The variable LmcsPivot[i] with i=0 . . . 16, the variables ScaleCoeff[i] and InvScaleCoeff[i] with i=0 . . . 15, are computed as follows:

TABLE-US-00012 LmcsPivot[0] = 0; for(i = 0; i <= 15; i++) {   LmcsPivot[i + 1] = LmcsPivot[i] + lmcsCW[i]   ScaleCoeff[i] = (lmcsCW[i] * (1 << 11) + (1 << (Log2(OrgCW) − 1)))) >> (Log2(OrgCW))   if(lmcsCW[i] = = 0)   InvScaleCoeff[i] = 0   else   InvScaleCoeff[i] = OrgCW * (1 << 11) / lmcsCW[i]

Forward Luma Mapping

[0285] As illustrated by FIG. **8** when the LMCS is applied for Luma, the Luma remapped sample called predMapSamples[i][j] is obtained from the prediction sample predSamples[i][j].

[0286] The predMapSamples[i][j] is computed as follows:

[0287] First of all, an index idx Y is computed from the prediction sample predSamples[i][j], at location (i,j) [0288] idx Y=predSamples[i][j]>>Log2 (OrgCW)

[0289] Then predMapSamples[i][j] is derived as follows by using the intermediate variables [0290] idx Y, LmcsPivot[idx Y] and InputPivot[idx Y] of section 0:

[00005]

predMapSamples[i][j] = LmcsPivot[idxY] + (ScaleCoeff[idxY] * (predSamples[i][j] - InputPivot[idxY]) + (1 $\ll$ 10)) $\gg$ 11

Luma Reconstruction Samples

[0291] The reconstruction process is obtained from the predicted luma sample predMapSample[i][j] and the residual luma samples resiSamples[i][j].

[0292] The reconstructed luma picture sample recSamples[i][j] is simply obtained by adding predMapSample[i][j] to resiSamples[i][j] as follows: [0293] recSamples[i][j]=Clip1 (predMapSamples[i][j]+resiSamples[i][j]])

[0294] In this above relation, the Clip 1 function is a clipping function to make sure that the reconstructed sample is between 0 and 1<<BitDepth-1.

Inverse Luma Mapping

[0295] When applying the inverse luma mapping according to FIG. **8**, the following operations are applied on each sample recSample[i][j] of the current block being processed:

[0296] First, an index idx Y is computed from the reconstruction sample [0297] recSamples[i][j], at location (i,j) [0298] idx Y=recSamples[i][j]>>Log2 (OrgCW) [0299] The inverse mapped luma sample invLumaSample[i][j] is derived as follows based on the:

[00006]

invLumaSample[i][j] = InputPivot[idxYInv] + (InvScaleCoeff[idxYInv] * (recSample[i][j] - LmcsPivot[idxYInv]) + (1 $\ll$ 10)) $\gg$ 1

[0300] A clipping operation is then done to get the final sample: [0301] finalSample[i][j]=Clip1 (invLumaSample[i][j])

Chroma Scaling

LMCS Semantics for Chroma Scaling

[0302] The syntax element lmcs_delta_abs_crs in Table 6 specifies the absolute codeword value of the variable lmcsDeltaCrs. The value of lmcs_delta_abs_crs shall be in the range of 0 and 7, inclusive. When not present, lmcs_delta_abs_crs is inferred to be equal to 0.

[0303] The syntax element lmcs_delta_sign_crs_flag specifies the sign of the variable lmcsDeltaCrs. When not present, lmcs_delta_sign_crs_flag is inferred to be equal to 0.

LMCS Intermediate Variable Computation for Chroma Scaling

[0304] To apply the Chroma scaling process, some intermediate variables are needed.

[0305] The variable lmcsDeltaCrs is derived as follows:

[00007]lmcsDeltaCrs = (1 - 2 * lmcs_delta_sign_crs_flag) * lmcs_delta_abs_crs

[0306] The variable ChromaScaleCoeff[i], with i=0 . . . 15, is derived as follows:

TABLE-US-00013 if(lmcsCW[i] == 0)   ChromaScaleCoeff[i] = (1 << 11) else   ChromaScaleCoeff[i] = OrgCW * (1 << 11) / (lmcsCW[i] +   lmcsDeltaCrs)

Chroma Scaling Process

[0307] In a first step, the variable invAvgLuma is derived in order to compute the average luma value of reconstructed Luma samples around the current corresponding Chroma block. The average Luma is computed from left and top luma block surrounding the corresponding Chroma block

[0308] If no sample is available the variable invAvgLuma is set as follows:

[00008]invAvgLuma = 1 $\ll$ (BitDepth - 1)

[0309] Based on the intermediate arrays LmcsPivot[ ] of section 0, the variable idx YInv is then derived as follows:

TABLE-US-00014 For (idxYInv = lmcs_min_bin_idx; idxYInv <= LmcsMaxBinIdx; idxYInv++) {   if(invAvgLuma < LmcsPivot [idxYInv + 1])   break } IdxYInv = Min(idxYInv, 15)

[0310] The variable varScale is derived as follows: [0311] varScale=ChromaScaleCoeff[idx YInv]

[0312] When a transform is applied on the current Chroma block, the reconstructed Chroma picture sample array recSamples is derived as follows

[00009]

recSamples[i][j] = Clip1(predSamples[i][j] + Sign(resiSamples[i][j]) * ((Abs(resiSamples[i][j]) * varScale + (1 $\ll$ 10)) $\gg$ 11))

[0313] If no transform has been applied for the current block, the following applies: [0314] recSamples[i][j]=Clip 1 (predSamples[i][j])

Encoder Consideration

[0315] The basic principle of an LMCS encoder is to first assign more codewords to ranges where those dynamic range segments have lower codewords than the average variance. In an alternative formulation of this, the main target of LMCS is to assign fewer codewords to those dynamic range segments that have higher codewords than the average variance. In this way, smooth areas of the picture will be coded with more codewords than average, and vice versa.

[0316] All the parameters (see Table 6) of the LMCS tools which are stored in the APS are determined at the encoder side. The LMCS encoder algorithm is based on the evaluation of local luma variance and is optimizing the determination of the LMCS parameters according to the basic principle described above. The optimization is then conducted to get the best PSNR metrics for the final reconstructed samples of a given block.

EMBODIMENTS

Reference Frame Signalling

Avoid Additional Reference Frame Signalling when Only One Slice

[0317] In an embodiment, the overriding of reference picture lists is not signalled in the slice header when at least one syntax element indicates that the current picture contains only one slice. Indeed, when a picture contains one slice, an

encoder should not override the reference picture lists as it should write it only once. Similarly, a decoder should not seek to parse syntax elements for overriding reference picture lists where the current picture contains only one slice. If the slice uses a reference picture list (or reference picture lists) transmitted in the SPS, there is an advantage to override one or more list to limit the number of reference pictures. Surprisingly, however, in terms of coding efficiency compromise for real applications, it is preferable to avoid this overriding to save the bits related to its signalling. Moreover, it simplifies the slice header parsing for some implementations.

Avoid Additional Reference Frame Signalling when PH is in the SH

[0318] In an embodiment, the syntax element(s) relating to overriding of reference frames is not signalled in the slice header when the picture header is in the slice header. More precisely, the syntax elements "num_ref_idx_active_override_flag" and "num_ref_idx_active_minus1[i]" are not transmitted when the flag picture_header_in_slice_header_flag is set equal to 1 as depicted in Table 12.

[0319] Moreover, the definition of "num_ref_idx_active_override_flag" should be modified as the following: [0320] "num_ref_idx_active_override_flag equal to 1 specifies that the syntax element num_ref_idx_active_minus1[0] is present for P and B slices and the syntax element num_ref_idx_active_minus1[1] is present for B slices.

num_ref_idx_active_override_flag equal to 0 specifies that the syntax elements num_ref_idx_active_minus1[0] and num_ref_idx_active_minus1[1] are not present. [0321] When not present, and when the slice headers referring to the PPS does not contain the PH syntax structure, the value of num_ref_idx_active_override flag is inferred to be equal to 1. Otherwise when not present, and when the slice headers referring to the PPS contain the PH syntax structure, the value of num_ref_idx_active_override flag is inferred to be equal to 0."

[0322] The advantage is a coding efficiency improvement when the picture is in the slice header. Indeed, the signalling of picture header in the slice header is efficient for low delay and low bitrate applications. In that cases the overriding flag cost for a multitude of pictures is larger than the cost of setting set of the several reference picture lists in the SPS. Indeed, generally for these use cases, the amount of reference frames is limited to one or two reference frames per list.

TABLE-US-00015 TABLE 12 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ...   if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag))   ref_pic_lists( )   if(!picture_header_in_slice_header_flag){   if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag)) &&   ((slice_type != I && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type = = B && num_ref_entries[1][RplsIdx[1]] > 1))) { num_ref_idx_active_override_flag u(1)   if(num_ref_idx_active_override_flag)   for(i = 0; i < (slice_type = = B ? 2: 1); i++)   if(num_ref_entries[i][RplsIdx[i]] > 1)   num_ref_idx_active_minus1[i] ue(v)   } } ...

Avoid Additional Reference Frame Signalling when Tiles in Slice is Equal to Tiles in Picture and the Number of Tiles in Picture is Greater than 1

[0323] In an embodiment, the overriding of reference frames is not signalled in the slice header the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture. In that case it is sure that the current picture contains only one slice. Table 13 illustrates this embodiment. Further, in an embodiment the requirement for the syntax for the overriding of reference frames to not be signalled (i.e. encoded or decoded) that the raster-scan slice mode is enabled is not required. That is to say, the syntax for the overriding of the reference frames may not be signalled even without the raster-scan slice mode being enabled.

[0324] Moreover, the definition of "num_ref_idx_active_override_flag" should be modified as the following: [0325] "num_ref_idx_active_override flag equal to 1 specifies that the syntax element num_ref_idx_active_minus1[0] is present for P and B slices and the syntax element num_ref_idx_active_minus1[1] is present for B slices.

num_ref_idx_active_override_flag equal to 0 specifies that the syntax elements num_ref_idx_active_minus1[0] and num_ref_idx_active_minus1[1] are not present. When not present, and when the raster-scan slice mode is not enabled or the number of tiles in the current picture is not superior to 1 or when the number of tiles in the slice is not equal to the number of tiles in the current picture, the value of num_ref_idx_active_override flag is inferred to be equal to 1. Otherwise, when not present, and when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture, the value of num_ref_idx_active_override flag is inferred to be equal to 0."

TABLE-US-00016 TABLE 13 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ...   if(!rect_slice_flag && NumTilesInPic > 1)   num_tiles_in_slice_minus1 ue(v) ... if(! (!rect_slice_flag && NumTilesInPic > 1 && num_tiles_in_slice_minus1 == NumTilesInPic-1))){   if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag)) &&   ((slice_type != I && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type = = B && num_ref_entries[1][RplsIdx[1]] > 1))) {   num_ref_idx_active_override_flag u(1)   if(num_ref_idx_active_override_flag)   for(i = 0; i < (slice type = = B ? 2: 1); i++)   if(num_ref_entries[i][RplsIdx[i]] > 1)   num_ref_idx_active_minus1[i] ue(v)   } } ...

[0326] In a further embodiment, the overriding of reference frames is not signalled in the slice header when the picture header is in the slice header or when the raster-scan slice mode is enabled, the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture as depicted in Table 14.

[0327] Moreover, the definition of "num_ref_idx_active_override_flag" should be modified as the following: [0328]

"num_ref_idx_active_override_flag equal to 1 specifies that the syntax element num_ref_idx_active_minus1[0] is present for P and B slices and the syntax element num_ref_idx_active_minus1[1] is present for B slices. num_ref_idx_active_override flag equal to 0 specifies that the syntax elements num_ref_idx_active_minus1[0] and num_ref_idx_active_minus1[1] are not present. When not present, and when the raster-scan slice mode not enabled or the number of tiles in the current picture is not superior to 1 or when the number of tiles in the slice is not equal to the number of tiles in the current picture or when the slice headers referring to the PPS does not contain the PH syntax structure, the value of num_ref_idx_active_override_flag is inferred to be equal to 1. Otherwise when not present, and when the raster-scan slice mode is enabled, the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture or when the slice headers referring to the PPS contain the PH syntax structure, the value of num_ref_idx_active_override flag is inferred to be equal to 0."

TABLE-US-00017 TABLE 14 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ... if((!picture_header_in_slice_header_flag) && (! (!rect_slice_flag && NumTilesInPic > 1 && num_tiles_in_slice_minus1 == NumTilesInPic-1)))){   if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag)) &&   ((slice_type != I && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type = = B && num_ref_entries[1][RplsIdx[1]] > 1))) {   num_ref_idx_active_override_flag u(1)   if(num_ref_idx_active_override_flag)   for(i = 0; i < (slice_type = = B ? 2: 1); i++)   if(num_ref_entries[i][RplsIdx[i]] > 1)   num_ref_idx_active_minus1[i] ue(v)   } } ...

Force Num_Ref_Idx_Active_Override_Flag Equal to 0 when Only One Slice

[0329] In embodiment, it is a bitstream requirement that the flag num_ref_idx_active_override_flag is set equal to 0 when the picture contains only one slice. Indeed, when a picture contains one slice, an encoder should not override the reference picture lists as it should write it only once. In terms of implementation, this simplifies the slice header parsing.

Force num_ref_idx_active_override flag equal to 0 when PH in SH

[0330] In one embodiment, it is a bitstream requirement that the flag num_ref_idx_active_override_flag is set equal to 0 when the picture header is in the slice header. More precisely, the picture header is in the slice header when the flag picture_header_in_slice_header_flag is set equal to 1.

Force Num_Ref_Idx_Active_Override Flag Equal to 0 when Tiles in Slice is Equal to Tiles in Pic and the Number of Tiles in Picture is Greater than 1

[0331] In one embodiment, it is a bitstream requirement that the flag num_ref_idx_active_override_flag is set equal to 0 when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture.

[0332] In a further embodiment, it is a bitstream requirement that the flag num_ref_idx_active_override_flag is set equal to 0 when the picture header is in the slice header or when the raster-scan slice mode is enabled, the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture. More precisely, when the flag picture_header_in_slice_header_flag is set equal to 1.

Avoid Additional the Reference Picture List Signalling when Current Picture List Refers to a SPS Reference Picture Lists

[0333] In an embodiment, the additional reference picture lists signalling is allowed only when the reference picture list for each list refers to a reference picture list signalled in the SPS. In the current VVC specifications, a reference picture list signalled in the SPS can be identified thanks the variable rpl_sps_flag[0] for list L0 and rpl_sps_flag[1] for list L1. Table 15 illustrates this embodiment, where the syntax element num_ref_idx_active_override_flag is extracted from the bitstream if the slice type is P or B and the number of reference frame for L0 is superior to 1 and if the reference picture list for L0 in signalled in the SPS or if the slice type is B and the number of reference frame for L1 is superior to 1 and if the reference picture list for L1 in signalled in the SPS. In the same way, if this flag is true, the number of reference active minus 1 is decoded for L0 if the slice type is P or B and the number of reference frame for L0 is superior to 1 and if the reference picture list for L0 is signalled in the SPS or if the slice type is B and the number of reference frame for L1 is superior to 1 and if the reference picture list for L1 is signalled in the SPS.

TABLE-US-00018 TABLE 15 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ...   if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag))   ref_pic_lists( )   if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag)) &&   ((slice_type != I && rpl_sps_flag[0] && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type = = B && rpl_sps_flag[1] && num_ref_entries[1][RplsIdx[1]] > 1))) {   num_ref_idx_active_override_flag u(1)   if(num_ref_idx_active_override_flag)   for(i = 0; i < (slice_type = = B ? 2: 1); i++)   if((num_ref_entries[i][RplsIdx[i]] > 1) && rpl_sps_flag[i])   num_ref_idx_active_minus1[i] ue(v)   } ...

[0334] In an embodiment, the reference picture list signalled in the SPS can be identified thanks the variable num_ref_pic_lists_in_sps[0] for list L0 and num_ref_pic_lists_in_sps[1] for list L1. This variable give the number of list signalled in the SPS. when it is equal to 0 it means that there is no reference picture list in the SPS. So this variable doesn't give the information for the current reference picture list signalling but for all reference picture lists using the same SPS. Table 16 illustrates this embodiment.

TABLE-US-00019 TABLE 16 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ...   if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag))   ref_pic_lists( )   if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type !=   IDR_N_LP) || sps_idr_rpl_present_flag)) &&   ((

slice_type != I && num_ref_pic_lists_in_sps [0] && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type == B && num_ref_pic_lists_in_sps [1] && num_ref_entries[1][RplsIdx[1]] > 1))) { num_ref_idx_active_override_flag u(1) if(num_ref_idx_active_override_flag) for(i = 0; i < (slice_type == B ? 2: 1); i++) if((num_ref_entries[i][RplsIdx[i]] > 1) && num_ref_pic_lists_in_sps [i]) num_ref_idx_active_minus1[i] ue(v) } ...

[0335] In the same way the num_ref_pic_lists_in_sps[1] can be replaced by the variable rpl1_idx_present_flag.

[0336] In an embodiment, the reference picture list signalled in the SPS can be identified thanks to the comparison of RplsIdx[i] to num_ref_pic_lists_in_sps[i]. For a list i when the reference picture list index RplsIdx[i] is equal to the number of reference picture lists transmitted in the SPS (num_ref_pic_lists_in_sps[i]), it is sure that the reference picture list has been transmitted in the current picture or slice and that it does not refer to reference picture list transmitted in the SPS. This embodiment gives a better precision than the previous embodiment.

Further Condition of Only One Slice

[0337] In a further embodiment, the additional reference picture lists signalling is allowed only when the reference picture list for each list refers to a reference picture list signalled in the SPS and when at least one syntax element indicates that current picture can contain more than one slice. Indeed, when a picture contains one slice, an encoder should not override a reference picture list which is explicitly transmitted in slice header or in the picture header. But if it uses a reference picture lists transmitted in the SPS, there is an advantage to override the list to limit the number of reference pictures. But in term of coding efficiency compromise for real application, it is preferable to avoid this overriding to save the bits related to its signalling. Moreover, it simplifies the slice header parsing for some implementations.

Further Condition of PH is in the SH

[0338] In another embodiment, the additional reference picture lists signalling is allowed only when the reference picture list for each list refers to a reference picture list signalled in the SPS and when the picture header is not in the slice header. More precisely, the syntax element "num_ref_idx_active_override_flag" and "num_ref_idx_active_minus1[i]" are not transmitted when flag picture_header_in_slice_header_flag is set equal to 1 as depicted in Table 17.

TABLE-US-00020 TABLE 17 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ... if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) || sps_idr_rpl_present_flag)) ref_pic_lists( ) if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) || sps_idr_rpl_present_flag)) && ((slice_type != I && rpl_sps_flag[0] && !picture_header_in_slice_header_flag && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type == B && rpl_sps_flag[1] && !picture_header_in_slice_header_flag && num_ref_entries[1][RplsIdx[1]] > 1)))) { num_ref_idx_active_override_flag u(1) if(num_ref_idx_active_override_flag) for(i = 0; i < (slice_type == B ? 2: 1); i++) if((num_ref_entries[i][RplsIdx[i]] > 1) && rpl_sps_flag[i] && !picture_header_in_slice_header_flag) num_ref_idx_active_minus1[i] ue(v) } ...

[0339] The advantage is a coding efficiency improvement when the picture is in the slice header. Indeed, the signalling of the picture header in the slice header is efficient for low delay and low bitrate applications, in that cases the overriding flag cost for a multitude of pictures is larger than the cost of setting the several reference picture lists in the SPS.

Further Condition of Tiles in Slice is Equal to Tiles in Pic and the Number of Tiles in Picture is Greater than 1

[0340] In one additional embodiment, the additional reference frame signalling is allowed only when the reference frame lists for each list refers to a reference picture list signalled in the SPS and when the raster-scan slice mode is disabled or the number of tiles in the current picture is equal 1 or the number of tiles in the slice is not equal to the number of tiles in the current picture. In that case, it is certain that the current picture contains only one slice. This can be implemented by changing !picture_header_in_slice_header_flag by (! (!rect_slice_flag && NumTilesInPic>1 & & num_tiles_in_slice_minus1==NumTilesInPic-1) in Table 17.

Signalling Conditional on Reference Picture List (RPL) Transmitted in Slice Header

[0341] In an embodiment, the additional reference picture lists signalling is allowed only when the reference picture list for each list refers to a reference picture list signalled in the SPS and when the reference picture lists are not transmitted in the slice header. The advantage is a coding efficiency improvement because it is not needed to update the reference picture lists if they are explicitly transmitted in the slice header.

Avoid Additional Reference Picture List Signalling Using a Syntax Element Transmitted in a Higher Level

[0342] In one embodiment, the additional reference picture lists signalling is allowed only when a high-level flag indicates that the reference picture lists can be override in the slice header.

[0343] Table 18 illustrates a possible implementation of this embodiment, where the flag high_level_slice_rpl_override_enabled_flag, when it equal to 1, specifies that the num_ref_idx_active_override_flag can be decoded if needed to override the current reference picture list. Otherwise, num_ref_idx_active_override_flag is not decoded.

TABLE-US-00021 TABLE 18 Partial Slice header showing modifications Descriptor slice_header( ) { ... if(!rpl_info_in_ph_flag && ((nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) || sps_idr_rpl_present_flag)) ref_pic_lists( ) if(high_level_slice_rpl_override_enabled_flag) { if((rpl_info_in_ph_flag || ((nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) || sps_idr_rpl_present_flag)) && ((slice_type != I && num_ref_entries[0][RplsIdx[0]] > 1) || (slice_type == B && num_ref_entries[1][RplsIdx[1]] > 1))) { num_ref_idx_active_override_flag u(1) if(num_ref_idx_active_override_flag) for(i = 0; i < (slice_type == B ? 2: 1); i++) if(num_ref_entries[i][RplsIdx[i]] > 1) num_ref_idx_active_minus1[i] ue(v) } } ...

[0344] The semantics of this flag should be defined as the following: [0345] "high level slice rpl override enabled_flag equal to 1 specifies that reference picture list syntax elements can be override in slice headers. high level_slice_rpl override enabled_flag equal to 0 specifies that reference picture list syntax elements can't be override in slice headers. When it is not present it is inferred to be equal to 0."

[0346] The advantage of this embodiment is a more flexibility compared to the previous embodiments with a similar coding efficiency improvement.

Signalling in SPS

[0347] In an embodiment, the high_level_slice_rpl_override_enabled_flag is transmitted in the SPS. In this embodiment, the name of this flag is sps_slice_rpl_override_enabled_flag.

Decoding Conditional on Num_Ref_Pic_Lists in Sps[i]

[0348] In an embodiment, the decoding of sps_slice_rpl_override_enabled_flag depends on the number of reference picture lists for each list. When they are both equal to 0, the sps_slice_rpl_override_enabled_flag is not decoded. Indeed, when there is no reference picture lists in the SPS the reference picture lists are transmitted for each picture or each slice. So it is not needed to override this information.

Flag not Decoded or Inferred when Only One Slice

[0349] In an embodiment, the sps_slice_rpl_override_enabled_flag is not decoded and/or inferred to be equal to 1 when there are more than one slice in pictures referring to the current SPS. The usage of only one slice can depends on syntax element signalling that the picture header is transmitted in the slice header or when the number of tiles in picture is the same as in the slice.

Flag not Decoded or Inferred when Rpl in SH

[0350] In one additional embodiment, the sps_slice_rpl_override_enabled_flag is not decoded when the reference picture lists are transmitted in the slice header. In that case, the flag rpl_info_in_ph_flag is set equal to 0. Indeed, if the reference picture lists are transmitted in the slice header (rpl_info_in_ph_flag equal to 0) and there is no reference picture list transmitted in the SPS, it is sure that the reference picture lists are transmitted for each slice. Accordingly, it is not needed to override this information. Table 19 illustrates this embodiment.

TABLE-US-00022 TABLE 19 Partial SPS showing modifications Descriptor seq_parameter_set_rbsp( ) { ... long_term_ref_pics_flag u(1)   inter_layer_ref_pics_present_flag u(1)   sps_idr_rpl_present_flag u(1) rpl1_same_as_rpl0_flag u(1)   for(i = 0; i < rpl1_same_as_rpl0_flag ? 1 : 2; i++) {   num_ref_pic_lists_in_sps[i] ue(v)   for(j = 0; j < num_ref_pic_lists_in_sps[i]; j++)   ref_pic_list_struct(i, j)   }   if(num_ref_pic_lists_in_sps[i] > 0 || num_ref_pic_lists_in_sps[i] > 0)   sps_slice_rpl_override_enabled_flag u(1) ...

Signalling in PPS

[0351] In an embodiment, the high_level_slice_rpl_override_enabled_flag is transmitted in the PPS. In this embodiment, the name of this flag is pps_slice_rpl_override_enabled_flag.

Decoding Conditional on Num_Ref_Pic_Lists in Sps[i]

[0352] In an embodiment, the decoding of pps_slice_rpl_override_enabled_flag depends on the number of reference picture lists for each list. When they are both equal to 0, the pps_slice_rpl_override_enabled_flag is not decoded. Indeed, when there is no reference picture lists in the SPS the reference picture lists are transmitted for each picture or each slice. So it is not needed to override this information.

Flag not Decoded or Inferred when Only One Slice

[0353] In one additional embodiment, the pps_slice_rpl_override_enabled_flag is not decoded and/or inferred to be equal to 1 when there are more than one slice in pictures referring to the current PPS. The usage of only one slice can depends on syntax element signalling that the picture header is transmitted in the slice header or when the number of tiles in picture is the same as in the slice.

Flag not Decoded or Inferred when Reference Picture List (Rpl) in Slice Header (SH)

[0354] In one additional embodiment, the pps_slice_rpl_override_enabled_flag is not decoded when the reference picture lists are transmitted in the slice header. In that case, the flag rpl_info_in_ph_flag is set equal to 0. Indeed, if the reference picture lists are transmitted in the slice header (rpl_info_in_ph_flag equal to 0) and they is no reference picture list transmitted in the SPS, it is sure that the reference picture lists are transmitted for each slice. So it is not needed to override this information. Table 20 illustrates this embodiment.

TABLE-US-00023 TABLE 20 Partial PPS showing modifications Descriptor pic_parameter_set_rbsp( ) { ... rpl_info_in_ph_flag u(1)   if(rpl_info_in_ph_flag && (num_ref_pic_lists_in_sps[i] > 0 || num_ref_pic_lists_in_sps[i] > 0))   pps_slice_rpl_override_enabled_flag u(1) ...

Signalling in Video Parameter Set (VPS)

[0355] In one additional embodiment, the high_level_slice_rpl_override_enabled_flag is transmitted in the VPS. In this embodiment, the name of this flag is vps_slice_rpl_override_enabled_flag.

Signalling in Picture Header (PH)

[0356] In one additional embodiment, the high_level_slice_rpl_override_enabled_flag is transmitted in the picture header. In this embodiment, the name of this flag ph_slice_rpl_override_enabled_flag.

Decoding Conditional on Num_Ref_Pic_Lists in Sps[i]

[0357] In one additional embodiment, the decoding of ph_slice_rpl_override_enabled_flag depends on the number of reference picture lists for each list. When these numbers are both equal to 0, the ph_slice_rpl_override_enabled_flag is not decoded. Indeed, when there is no reference picture lists in the SPS the reference picture lists are transmitted for each picture or each slice. So it is not needed to override this information.

Flag not Decoded or Inferred when Only One Slice

[0358] In one additional embodiment, the ph_slice_rpl_override_enabled_flag is not decoded and/or inferred to be equal to 1 when there are more than one slice in picture referring to the current picture header. The usage of only one slice can depends on syntax element signalling that the picture header is transmitted in the slice header or when the number of tiles in picture is the same as in the slice.

Flag not Decoded or Inferred when Rpl in Slice Header (SH)

[0359] In one additional embodiment, the ph_slice_rpl_override_enabled_flag is not decoded when the reference picture lists are transmitted in the slice header. In that case, the flag rpl_info_in_ph_flag is set equal to 0. Indeed, if the reference picture lists are transmitted in the slice header (rpl_info_in_ph_flag equal to 0) and they are not reference picture list transmitted in the picture header it is sure that the reference picture lists are transmitted for each slice so it is not needed to override this information. Table 21 illustrates an implementation of this embodiment.

TABLE-US-00024 TABLE 21 Partial Picture Header showing modifications Descriptor picture_header_structure( ) { ... if(rpl_info_in_ph_flag)   ref_pic_lists( )   else   if(num_ref_pic_lists_in_sps[i] > 0 || num_ref_pic_lists_in_sps[i] > 0)   ph_slice_rpl_override_enabled_flag u(1) ...

Embodiments Related to LMCS and Scaling Lists

Avoid Signalling of XXX Activation Flag when Only One Slice

[0360] In one embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool (or parameter) XXX and which depends on at least one variable which enables or specifies the presence of this tool (or parameter) XXX in the picture header, is not transmitted in the slice header when the current picture contains only one slice.

[0361] The advantage of this embodiment, is a coding efficiency improvement as the syntax elements are not transmitted when it is not needed. Indeed, when the current picture contains only one slice, there is no additional flexibility to signal it in the picture header and then slice header.

Avoid Signalling of XXX Activation Flag when PH in SH

[0362] In an embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool (or parameter) and which depends on at least one variable which enables or specifies the presence of this tool in the picture header, is not transmitted in the slice header when the picture header is in the slice header.

[0363] The advantage of this additional embodiment, is a coding efficiency improvement when the picture is in the slice header. Indeed, the picture header in the slice header is efficient for low delay and low bitrate applications, in that cases the signalling at slice level has a significant cost on the global bitrate.

[0364] Table 22 shows an implementation of this embodiment.

Avoid Signalling of XXX Flag when Tiles in Slice is Equal to Tiles in Pic and the Number of Tiles in Picture is Greater than 1

[0365] In an embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool and which depends on at least one variable which enables or specifies the presence of this tool in the picture header, is not transmitted in the slice when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture.

[0366] This embodiment can be implemented by changing the 2 conditions "&&! picture_header_in_slice_header_flag" by "&& (! (!rect_slice_flag && NumTilesInPic>1 && num_tiles_in_slice_minus1==NumTilesInPic-1)))" in Table 22.

Slice XXX Flag Predicted by Value pH XXX Flag

[0367] In an embodiment, when the syntax element transmitted in the slice header which enables or specifies the presence of a tool (or parameter) and which depends on at least one variable which enables or specifies the presence of this tool (or parameter) in the picture header, is not transmitted thanks to the conditions defined above, its value is predicted by the value of the variable transmitted or obtained in the picture header.

XXX is LMCS

[0368] In an embodiment, when the syntax element slice_lmcs_enabled_flag which enables LMCS at slice level is not transmitted thanks to the conditions defined above.

[0369] Table 22 illustrates this embodiment when the condition is the picture header is in the slice header.

[0370] Moreover, the variable slice_lmcs_enabled_flag is predicted by the value of ph_lmcs_enabled_flag when the conditions defined above are respected. For example, for the condition the picture header is in the slice header, the slice_lmcs_enabled_flag is defined as follows: [0371] "slice_lmcs_enabled_flag equal to 1 specifies that luma mapping with chroma scaling is enabled for the current slice. slice_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not enabled for the current slice. When slice_lmcs_enabled_flag is not present, and when slice headers referring to the PPS don't contain the PH syntax structure it is inferred to be equal to 0. When slice headers referring to the PPS contain the PH syntax structure slice_lmcs_enabled_flag is inferred to be equal to ph_lmcs_enabled_flag."

TABLE-US-00025 TABLE 22 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ...   if(ph_lmcs_enabled_flag && ! picture_header_in_slice_header_flag) slice_lmcs_enabled_flag u(1)   if(ph_scaling_list_present_flag && ! picture_header_in_slice_header_flag) slice_scaling_list_present_flag u(1) ... }

XXX is Scaling List

[0372] In one embodiment, when the syntax element slice_scaling_list_present_flag which specifies that the scaling list are present for the current slice is not transmitted thanks to the conditions defined above.

[0373] Table 23 illustrates an implementation of this embodiment when the condition is the picture header is in the slice header.

[0374] Moreover, the variable slice_scaling_list_present_flag is predicted by the value of ph_scaling_list_present_flag when the conditions defined above are respected. For example, for the condition the "picture header is in the slice header", the slice_lmcs_enabled_flag is defined as follows: [0375] "slice_scaling_list_present_flag equal to 1 specifies that the scaling list data used for the current slice is derived based on the scaling list data contained in the referenced scaling list APS with aps_params_type equal to SCALING_APS and adaptation_parameter_set_id equal to ph_scaling_list_aps_id. slice_scaling_list_present_flag equal to 0 specifies that the scaling list data used for the current picture is the default scaling list data derived specified in clause 7.4.3.21. When not present, and when slice headers referring to the PPS don't contain the PH syntax structure, the value of slice_scaling_list_present_flag is inferred to be equal to 0. When slice headers referring to the PPS contain the PH syntax structure slice_lmcs_enabled_flag is inferred to be equal to ph scaling list_present_flag."

[0376] In an embodiment the proposed restriction on variable XXX is applied for both slice_lmcs_enabled_flag and slice_scaling_list_present_flag

[0377] In an embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool and which depends on at least one variable which enables or specifies the presence of this tool in the picture header, is not transmitted in the slice header when the picture header is in the slice header or when the raster-scan slice mode is disabled or the number of tiles in the current picture is equal 1 or the number of tiles in the slice is not equal to the number of tiles in the current picture. Table 23 illustrates an implementation of this embodiment.

[0378] In a further embodiment, slice_lmcs_enabled_flag and slice_scaling_list_present_flag are transmitted in the slice header which respectively enable LMCS and specify the presence of scaling list and which depend respectively on the ph_lmcs_enabled_flag and on the ph_scaling_list_present_flag in the picture header, are not transmitted in the slice header when the picture header is in the slice header or when the raster-scan slice mode is disabled or the number of tiles in the current picture is equal 1 or the number of tiles in the slice is not equal to the number of tiles in the current picture.

TABLE-US-00026 TABLE 23 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) .. if((rect_slice_flag && NumSlicesInSubpic[CurrSubpicIdx] > 1) || (!rect_slice_flag && NumTilesInPic > 1)) slice_address u(v) ... if(!rect_slice_flag && NumTilesInPic > 1) num_tiles_in_slice_minus1 ue(v) ... if(ph_lmcs_enabled_flag && ! picture_header_in_slice_header_flag &&(! (!rect_slice_flag && NumTilesInPic > 1 && num_tiles_in_slice_minus1 == NumTilesInPic-1)) slice_lmcs_enabled_flag u(1) if(ph_scaling_list_present_flag && ! picture_header_in_slice_header_flag &&(! (!rect_slice_flag && NumTilesInPic > 1 && num_tiles_in_slice_minus1 == NumTilesInPic- 1)) slice_scaling_list_present_flag u(1) ... }

Bitstream Constraint of Slice XXX Flag Equal to pH_XXX_Flag when Only One Slice

[0379] In an embodiment, a syntax element is transmitted in the slice header which enables or specifies the presence of a tool or parameter XXX and which depends on at least one variable which enables or specifies the presence of this tool in the picture header when the current picture contains only one slice. A bitstream conformance requirement may be in place to make the syntax element in the slice header have the same value as the syntax element in the picture header which enables or specifies the presence of the same tool or parameter when there is only one slice.

Bitstream Constraint of Slice XXX Flag Equal to pH_XXX_Flag when PH in SH

[0380] In one embodiment, a syntax element is transmitted in the slice header which enables or specifies the presence of a tool or parameter XXX and which depends on at least one variable which enables or specifies the presence of this tool in the picture header when the picture header is in the slice header. That is to say a bitstream conformance requirement may be in place to make the syntax element in the slice header have the same value as the syntax element in the picture header which enables or specifies the presence of the same tool or parameter when the picture header is in the slice header.

Bitstream Constraint of Slice XXX Flag Equal to pH_XXX_Flag when Tiles in Slice is Equal to Tiles in Pic and the Number of Tiles in Picture is Greater than 1

[0381] In one embodiment, a syntax element is transmitted in the slice header which enables or specifies the presence of a tool or parameter XXX and which depends on at least one variable which enables or specifies the presence of this tool in the picture header when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture. A bitstream conformance requirement may be in place to make the syntax element in the slice header have the same value as the syntax element in the picture header when the number of tiles in slice is equal to tiles in pic and the number of tiles in picture is greater than 1.

XXX is LMCS

[0382] In one embodiment, syntax element slice_lmcs_enabled_flag which enables LMCS at slice level is systematically equal to ph_lmcs_enabled_flag when one condition defined above is true.

[0383] For example the slice_lmcs_enabled_flag is defined as follows when the condition is "picture header in the slice header" [0384] "slice_lmcs_enabled_flag equal to 1 specifies that luma mapping with chroma scaling is enabled for the current slice. slice_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not enabled for the current slice. When slice_lmcs_enabled_flag is not present, it is inferred to be equal to 0. When slice headers referring to the PPS contain the PH syntax structure it is a requirement of bitstream conformance that slice_lmcs_enabled_flag shall be equal to ph_lmcs_enabled_flag."

XXX is Scaling List

[0385] In one embodiment, syntax element slice_scaling_list_present_flag which specifies that the scaling list are present for the current slice is systematically equal to ph_scaling_list_present_flag when one condition defined above is true.

[0386] For example the slice_scaling_list_present_flag is defined as follow when the condition is "picture header in the slice header": [0387] "slice_scaling_list_present_flag equal to 1 specifies that the scaling list data used for the current slice is

derived based on the scaling list data contained in the referenced scaling list APS with aps_params_type equal to SCALING_APS and adaptation_parameter_set_id equal to ph_scaling_list_aps_id. slice_scaling_list_present_flag equal to 0 specifies that the scaling list data used for the current picture is the default scaling list data derived specified in clause 7.4.3.21. When not present, the value of slice_scaling_list_present_flag is inferred to be equal to 0. When slice headers referring to the PPS contain the PH syntax structure slice_scaling_list_present_flag is inferred to be equal to ph_scaling_list_present_flag." When slice headers referring to the PPS contain the PH syntax structure it is a requirement of bitstream conformance that slice_lmcs_enabled_flag shall be equal to ph_lmcs_enabled_flag."

[0388] In an embodiment, the restriction is applied to LMCS and Scaling list

[0389] In an embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool (or parameter) and which depends on at least one variable which enables or specifies the presence of this tool (or parameter) in the picture header when the picture header is in the slice header or when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture.

[0390] In an embodiment, slice_lmcs_enabled_flag and slice_scaling_list_present_flag which respectively enable LMCS and specify the presence of scaling list and which respectively depend on ph_lmcs_enabled_flag and ph_scaling_list_present_flag which respectively enable and specify the presence of LMCS and Scaling list in the picture header when the picture header is in the slice header or when the raster-scan slice mode is enabled, and the number of tiles in the current picture is superior to 1 and when the number of tiles in the slice is equal to the number of tiles in the current picture.

Avoid Signalling of XXX Activation Flag when Thanks to a Flag Transmitted in an Higher Level

[0391] In an embodiment, a syntax element transmitted in the slice header which enables or specifies the presence of a tool or parameter XXX is transmitted only when a high-level flag indicates the presence of this syntax element.

Signalling in SPS, PPS, VPS, PH

[0392] In an embodiment, this high-level flag is transmitted in the SPS or PPS or VPS or picture header. It is optimal to transmit it at the highest possible level.

Slice XXX Flag Predicted by Value pH XXX Flag

[0393] In an embodiment, when the syntax element transmitted in the slice header which enables or specifies the presence of a tool and which depends to at least one variable which enables or specifies the presence of this tool in the picture header, is transmitted only when a high level flag indicates the presence of this syntax element and its value is predicted by the value of the variable transmitted or obtained in the picture header.

XXX is LMCS

[0394] In an embodiment, when the syntax element slice_lmcs_enabled_flag which enables LMCS at slice level is transmitted only when a high-level flag indicates the presence of the slice_lmcs_enabled_flag.

[0395] Table 24 illustrates this embodiment for a high-level flag transmitted in the SPS sps_override_slice_lmcs_enabled_flag.

TABLE-US-00027 TABLE 24 Partial Slice header showing modifications Descriptor slice_header( ) { picture_header_in_slice_header_flag u(1) ... if(ph_lmcs_enabled_flag && sps_override_slice_lmcs_enabled_flag) slice_lmcs_enabled_flag u(1) if(ph_scaling_list_present_flag && sps_override_slice_scaling_list_present_flag) slice_scaling_list_present_flag u(1) ... }

XXX is Scaling List

[0396] In an embodiment, when the syntax element slice_scaling_list_present_flag which specifies that the scaling list are present for the current slice when a high level flag indicates the presence of slice_scaling_list_present_flag.

[0397] Table 24 illustrates this embodiment for a high level flag transmitted in the SPS sps_override_slice_scaling_list_present_flag.

[0398] In an embodiment the proposed restriction on variable XXX is applied for both slice_lmcs_enabled_flag and slice_scaling_list_present_flag

Implementations

[0399] FIG. **11** shows a system **191 195** comprising at least one of an encoder **150** or a decoder **100** and a communication network **199** according to embodiments of the present invention. According to an embodiment, the system **195** is for processing and providing a content (for example, a video and audio content for displaying/outputting or streaming video/audio content) to a user, who has access to the decoder **100**, for example through a user interface of a user terminal comprising the decoder **100** or a user terminal that is communicable with the decoder **100**. Such a user terminal may be a computer, a mobile phone, a tablet or any other type of a device capable of providing/displaying the (provided/streamed) content to the user. The system **195** obtains/receives a bitstream **101** (in the form of a continuous stream or a signal—e.g. while earlier video/audio are being displayed/output) via the communication network **199**. According to an embodiment, the system **191** is for processing a content and storing the processed content, for example a a video and audio content processed for displaying/outputting/streaming at a later time. The system **191** obtains/receives a content comprising an original sequence of images **151**, which is received and processed (including filtering with a deblocking filter according to the present invention) by the encoder **150**, and the encoder **150** generates a bitstream **101** that is to be communicated to the decoder **100** via a communication network **191**. The bitstream **101** is then communicated to the decoder **100** in a number of ways, for example it may be generated in advance by the encoder **150** and stored as data in a storage apparatus in the communication network **199** (e.g. on a server or a cloud storage) until a user requests the content (i.e. the bitstream data) from the storage apparatus, at which point the data is communicated/streamed to the decoder **100** from the storage apparatus.

The system **191** may also comprise a content providing apparatus for providing/streaming, to the user (e.g. by communicating data for a user interface to be displayed on a user terminal), content information for the content stored in the storage apparatus (e.g. the title of the content and other meta/storage location data for identifying, selecting and requesting the content), and for receiving and processing a user request for a content so that the requested content can be delivered/streamed from the storage apparatus to the user terminal. Alternatively, the encoder **150** generates the bitstream **101** and communicates/streams it directly to the decoder **100** as and when the user requests the content. The decoder **100** then receives the bitstream **101** (or a signal) and performs filtering with a deblocking filter according to the invention to obtain/generate a video signal **109** and/or audio signal, which is then used by a user terminal to provide the requested content to the user.

[0400] Any step of the method/process according to the invention or functions described herein may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the steps/functions may be stored on or transmitted over, as one or more instructions or code or program, or a computer-readable medium, and executed by one or more hardware-based processing unit such as a programmable computing machine, which may be a PC ("Personal Computer"), a DSP ("Digital Signal Processor"), a circuit, a circuitry, a processor and a memory, a general purpose microprocessor or a central processing unit, a microcontroller, an ASIC ("Application-Specific Integrated Circuit"), a 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 describe herein.

[0401] Embodiments of the present invention can also be realized by wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of JCs (e.g. a chip set). Various components, modules, or units are described herein to illustrate functional aspects of devices/apparatuses configured to perform those embodiments, but do not necessarily require realization by different hardware units. Rather, various modules/units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors in conjunction with suitable software/firmware.

[0402] Embodiments of the present invention can be realized by a computer of a system or apparatus that reads out and executes computer executable instructions (e.g., one or more programs) recorded on a storage medium to perform the modules/units/functions of one or more of the above-described embodiments and/or that includes one or more processing unit or circuits for performing the functions of one or more of the above-described embodiments, and by a method performed by the computer of the system or apparatus by, for example, reading out and executing the computer executable instructions from the storage medium to perform the functions of one or more of the above-described embodiments and/or controlling the one or more processing unit or circuits to perform the functions of one or more of the above-described embodiments. The computer may include a network of separate computers or separate processing units to read out and execute the computer executable instructions. The computer executable instructions may be provided to the computer, for example, from a computer-readable medium such as a communication medium via a network or a tangible storage medium. The communication medium may be a signal/bitstream/carrier wave. The tangible storage medium is a "non-transitory computer-readable storage medium" which may include, for example, one or more of a hard disk, a random-access memory (RAM), a read only memory (ROM), a storage of distributed computing systems, an optical disk (such as a compact disc (CD), digital versatile disc (DVD), or Blu-ray Disc (BD)™), a flash memory device, a memory card, and the like. At least some of the steps/functions may also be implemented in hardware by a machine or a dedicated component, such as an FPGA ("Field-Programmable Gate Array") or an ASIC ("Application-Specific Integrated Circuit").

[0403] FIG. **12** is a schematic block diagram of a computing device **2000** for implementation of one or more embodiments of the invention. The computing device **2000** may be a device such as a micro-computer, a workstation or a light portable device. The computing device **2000** comprises a communication bus connected to:—a central processing unit (CPU) **2001**, such as a microprocessor;—a random access memory (RAM) **2002** for storing the executable code of the method of embodiments of the invention as well as the registers adapted to record variables and parameters necessary for implementing the method for encoding or decoding at least part of an image according to embodiments of the invention, the memory capacity thereof can be expanded by an optional RAM connected to an expansion port for example;—a read only memory (ROM) **2003** for storing computer programs for implementing embodiments of the invention;—a network interface (NET) **2004** is typically connected to a communication network over which digital data to be processed are transmitted or received. The network interface (NET) **2004** can be a single network interface, or composed of a set of different network interfaces (for instance wired and wireless interfaces, or different kinds of wired or wireless interfaces). Data packets are written to the network interface for transmission or are read from the network interface for reception under the control of the software application running in the CPU **2001**;—a user interface (UI) **2005** may be used for receiving inputs from a user or to display information to a user;—a hard disk (HD) **2006** may be provided as a mass storage device;—an Input/Output module (IO) **2007** may be used for receiving/sending data from/to external devices such as a video source or display. The executable code may be stored either in the ROM **2003**, on the HD **2006** or on a removable digital medium such as, for example a disk. According to a variant, the executable code of the programs can be received by means of a communication network, via the NET **2004**, in order to be stored in one of the storage means of the communication device **2000**, such as the HD **2006**, before being executed. The CPU **2001** is adapted to control and direct the execution of the instructions or portions of software code of the program or programs according to embodiments of the invention, which instructions are stored in one of the aforementioned storage means. After powering on, the CPU **2001** is capable of executing instructions from main RAM memory **2002** relating to a software application after those instructions have been loaded from the program ROM **2003** or the HD **2006**, for example. Such a software application, when executed by the CPU **2001**, causes the steps of the method

according to the invention to be performed.

[0404] It is also understood that according to another embodiment of the present invention, a decoder according to an aforementioned embodiment is provided in a user terminal such as a computer, a mobile phone (a cellular phone), a table or any other type of a device (e.g. a display apparatus) capable of providing/displaying a content to a user. According to yet another embodiment, an encoder according to an aforementioned embodiment is provided in an image capturing apparatus which also comprises a camera, a video camera or a network camera (e.g. a closed-circuit television or video surveillance camera) which captures and provides the content for the encoder to encode. Two such examples are provided below with reference to FIGS. **13** and **14**.

Network Camera

[0405] FIG. **13** is a diagram illustrating a network camera system **2100** including a network camera **2102** and a client apparatus **2104**.

[0406] The network camera **2102** includes an imaging unit **2106**, an encoding unit **2108**, a communication unit **2110**, and a control unit **2112**.

[0407] The network camera **2102** and the client apparatus **2104** are mutually connected to be able to communicate with each other via the network **200**.

[0408] The imaging unit **2106** includes a lens and an image sensor (e.g., a charge coupled device (CCD) or a complementary metal oxide semiconductor (CMOS)), and captures an image of an object and generates image data based on the image. This image can be a still image or a video image.

[0409] The encoding unit **2108** encodes the image data by using said encoding methods described above

[0410] The communication unit **2110** of the network camera **2102** transmits the encoded image data encoded by the encoding unit **2108** to the client apparatus **2104**.

[0411] Further, the communication unit **2110** receives commands from client apparatus **2104**. The commands include commands to set parameters for the encoding of the encoding unit **2108**.

[0412] The control unit **2112** controls other units in the network camera **2102** in accordance with the commands received by the communication unit **2110**.

[0413] The client apparatus **2104** includes a communication unit **2114**, a decoding unit **2116**, and a control unit **2118**.

[0414] The communication unit **2114** of the client apparatus **2104** transmits the commands to the network camera **2102**.

[0415] Further, the communication unit **2114** of the client apparatus **2104** receives the encoded image data from the network camera **2102**.

[0416] The decoding unit **2116** decodes the encoded image data by using said decoding methods described above.

[0417] The control unit **2118** of the client apparatus **2104** controls other units in the client apparatus **2104** in accordance with the user operation or commands received by the communication unit **2114**.

[0418] The control unit **2118** of the client apparatus **2104** controls a display apparatus **2120** so as to display an image decoded by the decoding unit **2116**.

[0419] The control unit **2118** of the client apparatus **2104** also controls a display apparatus **2120** so as to display GUI (Graphical User Interface) to designate values of the parameters for the network camera **2102** includes the parameters for the encoding of the encoding unit **2108**.

[0420] The control unit **2118** of the client apparatus **2104** also controls other units in the client apparatus **2104** in accordance with user operation input to the GUI displayed by the display apparatus **2120**.

[0421] The control unit **2119** of the client apparatus **2104** controls the communication unit **2114** of the client apparatus **2104** so as to transmit the commands to the network camera **2102** which designate values of the parameters for the network camera **2102**, in accordance with the user operation input to the GUI displayed by the display apparatus **2120**.

Smart Phone

[0422] FIG. **14** is a diagram illustrating a smart phone **2200**.

[0423] The smart phone **2200** includes a communication unit **2202**, a decoding unit **2204**, a control unit **2206**, display unit **2208**, an image recording device **2210** and sensors **2212**. [0424] the communication unit **2202** receives the encoded image data via network **200**.

[0425] The decoding unit **2204** decodes the encoded image data received by the communication unit **2202**.

[0426] The decoding unit **2204** decodes the encoded image data by using said decoding methods described above.

[0427] The control unit **2206** controls other units in the smart phone **2200** in accordance with a user operation or commands received by the communication unit **2202**.

[0428] For example, the control unit **2206** controls a display unit **2208** so as to display an image decoded by the decoding unit **2204**.

[0429] While the present invention has been described with reference to embodiments, it is to be understood that the invention is not limited to the disclosed embodiments. It will be appreciated by those skilled in the art that various changes and modification might be made without departing from the scope of the invention, as defined in the appended claims. All of the features disclosed in this specification (including any accompanying claims, abstract and drawings), and/or all of the steps of any method or process so disclosed, may be combined in any combination, except combinations where at least some of such features and/or steps are mutually exclusive. Each feature disclosed in this specification (including any accompanying claims, abstract and drawings) may be replaced by alternative features serving the same, equivalent or similar purpose, unless expressly stated otherwise. Thus, unless expressly stated otherwise, each feature disclosed is one example only of a generic series of equivalent or similar features.

[0430] It is also understood that any result of comparison, determination, assessment, selection, execution, performing, or

consideration described above, for example a selection made during an encoding or filtering process, may be indicated in or determinable/inferable from data in a bitstream, for example a flag or data indicative of the result, so that the indicated or determined/inferred result can be used in the processing instead of actually performing the comparison, determination, assessment, selection, execution, performing, or consideration, for example during a decoding process.

[0431] In the claims, the word "comprising" does not exclude other elements or steps, and the indefinite article "a" or "an" does not exclude a plurality. The mere fact that different features are recited in mutually different dependent claims does not indicate that a combination of these features cannot be advantageously used.

[0432] Reference numerals appearing in the claims are by way of illustration only and shall have no limiting effect on the scope of the claims.

Claims

1. A method of decoding video data from a bitstream, wherein the bitstream comprises a picture header syntax structure comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the method comprising: parsing syntax elements, and decoding said video data from said bitstream using said parsed syntax elements, wherein a first flag relating to availability of Luma Mapping with Chroma Scaling (LMCS) is parsed from the slice header depending on a value of a second flag, the second flag being in the slice header and indicating whether the picture header syntax structure is present in the slice header, wherein parsing of the first flag from the slice header is omitted and a value of first flag is inferred to be equal to a value of a third flag, related to LMCS, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header, wherein a fourth flag relating to use of a scaling list for a slice is parsed from the slice header depending on a value of the second flag, and wherein parsing of the fourth flag from the slice header is omitted and a value of the fourth flag is inferred to be equal to a value of a fifth flag, related to a scaling list, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header.

2. The method according to claim 1, wherein the bitstream further comprises an adaptation parameter set (APS) having an aps_params_type syntax element equal to SCALING_APS, and wherein if a value of the fourth flag for the slice is 1, scaling list data contained in the adaptation parameter set (APS) is capable of being used for the slice.

3. A device for decoding video data from a bitstream, the device being configured to perform a method of decoding video data from a bitstream, wherein the bitstream comprises a picture header syntax structure comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the method comprising: parsing syntax elements, and decoding said video data from said bitstream using said parsed syntax elements, wherein a first flag relating to availability of Luma Mapping with Chroma Scaling (LMCS) is parsed from the slice header depending on a value of a second flag, the second flag being in the slice header and indicating whether the picture header syntax structure is present in the slice header, wherein parsing of the first flag from the slice header is omitted and a value of the first flag is inferred to be equal to a value of a third flag, related to LMCS, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header, wherein a fourth flag relating to use of a scaling list for a slice is parsed from the slice header depending on a value of the second flag, and wherein parsing of the fourth flag from the slice header is omitted and a value of the fourth flag is inferred to be equal to a value of a fifth flag, related to a scaling list, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header.

4. The device according to claim 3, wherein the bitstream further comprises an adaptation parameter set (APS) having an aps_params_type syntax element equal to SCALING_APS, and wherein if a value of the fourth flag for the slice is 1, scaling list data contained in the adaptation parameter set (APS) is capable of being used for the slice.

5. A method of encoding video data into a bitstream, wherein the bitstream comprises a picture header syntax structure comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the method comprising: encoding syntax elements, and encoding said video data into said bitstream, wherein a first flag relating to availability of Luma Mapping with Chroma Scaling (LMCS) is encoded into the slice header depending on a value of a second flag, the second flag being in the slice header and indicating whether the picture header syntax structure is present in the slice header, wherein encoding of the first flag into the slice header is omitted and a value of the first flag is to be inferred to be equal to a value of a third flag, related to LMCS, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header, wherein a fourth flag relating to use of a scaling list for a slice is encoded into the slice header depending on a value of the second flag, and wherein encoding of the fourth flag into the slice header is omitted and a value of the fourth flag is to be inferred to be equal to a value of a fifth flag, related to a scaling list, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header.

6. The method according to claim 5, wherein the bitstream further comprises an adaptation parameter set (APS) having an aps_params_type syntax element equal to SCALING_APS, and wherein if a value of the fourth flag for the slice is 1, scaling list data contained in the adaptation parameter set (APS) is capable of being used for the slice.

7. A device for encoding video data into a bitstream, the device being configured to perform a method of encoding video data into a bitstream, wherein the bitstream comprises a picture header syntax structure comprising syntax elements to be used when decoding one or more slices, and a slice header comprising syntax elements to be used when decoding a slice, the method comprising: encoding syntax elements, and encoding said video data into said bitstream, wherein a first flag relating

to availability of Luma Mapping with Chroma Scaling (LMCS) is encoded into the slice header depending on a value of a second flag, the second flag being in the slice header and indicating whether the picture header syntax structure is present in the slice header, wherein encoding of the first flag into the slice header is omitted and a value of the first flag is to be inferred to be equal to a value of a third flag, related to LMCS, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header, wherein a fourth flag relating to use of a scaling list for a slice is encoded into the slice header depending on a value of the second flag, and wherein encoding of the fourth flag into the slice header is omitted and a value of the fourth flag is to be inferred to be equal to a value of a fifth flag, related to a scaling list, in the picture header syntax structure if the second flag indicates that the picture header syntax structure is present in the slice header.

8. The device according to claim 7, wherein the bitstream further comprises an adaptation parameter set (APS) having an aps_params_type syntax element equal to SCALING_APS, and wherein if a value of the fourth flag for the slice is 1, scaling list data contained in the adaptation parameter set (APS) is capable of being used for the slice.