



- (51) **International Patent Classification:**
H04N 19/46 (2014.01)
- (21) **International Application Number:**
PCT/US2021/012035
- (22) **International Filing Date:**
04 January 2021 (04.01.2021)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
62/957,123 04 January 2020 (04.01.2020) US
- (71) **Applicant:** **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).
- (72) **Inventors:** **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).
- (74) **Agent:** **VINAY SATHE**; PERKINS COIE LLP, P.O. Box 1247, Seattle, Washington 98111-1247 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) **Title:** RESTRICTIONS ON INTER PREDICTION FOR SUBPICTURE

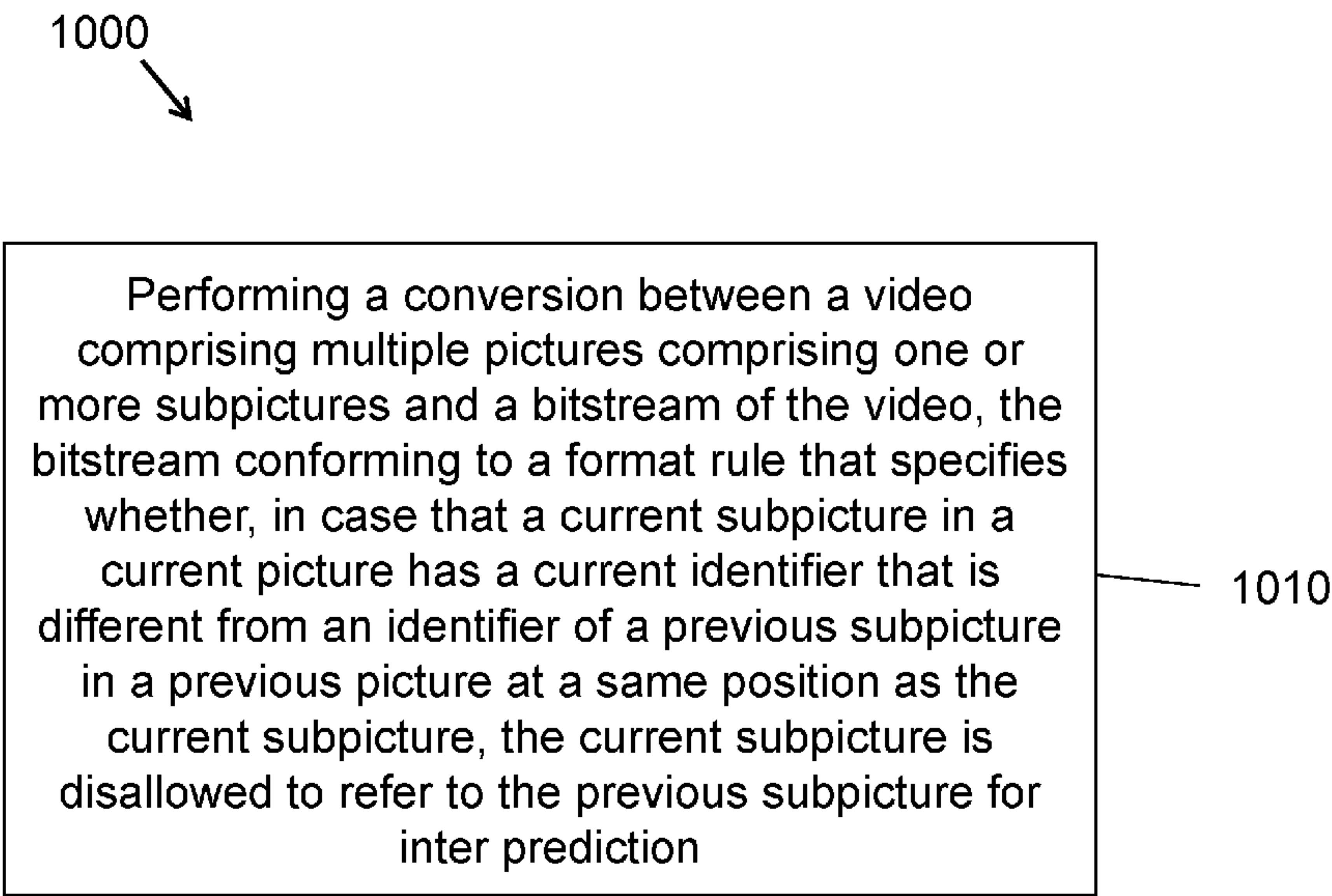


FIG. 10

(57) **Abstract:** Methods, devices and systems for video coding and encoding, which include restrictions on inter prediction for subpictures, are described. One example method of video processing includes performing a conversion between a video comprising multiple pictures comprising one or more subpictures and a bitstream of the video, wherein the bitstream conforms to a format rule that specifies whether, in case that a current subpicture in a current picture has a current identifier that is different from an identifier of a previous subpicture in a previous picture at a same position as the current subpicture, the current subpicture is disallowed to refer to the previous subpicture for inter prediction.

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

RESTRICTIONS ON INTER PREDICTION FOR SUBPICTURE

CROSS-REFERENCE TO RELATED APPLICATION

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of U.S. Provisional Patent Application No. US 62/957,123 filed on January 4, 2020. For all purposes under the law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This patent document relates to image and video coding and decoding.

BACKGROUND

[003] Digital video accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[004] The present document discloses techniques that can be used by video encoders and decoders for video encoding or decoding, and includes restrictions on inter prediction for subpictures.

[005] In one example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising multiple pictures comprising one or more subpictures and a bitstream of the video, wherein the bitstream conforms to a format rule that specifies whether, in case that a current subpicture in a current picture has a current identifier that is different from an identifier of a previous subpicture in a previous picture at a same position as the current subpicture, the current subpicture is disallowed to refer to the previous subpicture for inter prediction.

[006] In another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising a video region and a bitstream of the video comprising multiple coding layers, wherein the bitstream conforms to a format rule, and wherein

the format rule specifies whether an inter-layer prediction (ILP) between video regions in different coding layers of the multiple coding layers is allowed based on a condition.

[007] In yet another example aspect, a video processing method is disclosed. The method includes performing a conversion between a video comprising a current video region and a bitstream of the video comprising multiple coding layers, wherein the bitstream conforms to a format rule, and wherein the format rule specifies the bitstream includes an indication of whether an inter-layer prediction (ILP) between the current video region and a video region in a reference layer is allowed.

[008] In yet another example aspect, a video encoder apparatus is disclosed. The video encoder comprises a processor configured to implement above-described methods.

[009] In yet another example aspect, a video decoder apparatus is disclosed. The video decoder comprises a processor configured to implement above-described methods.

[0010] In yet another example aspect, a computer readable medium having code stored thereon is disclosed. The code embodies one of the methods described herein in the form of processor-executable code.

[0011] These, and other, features are described throughout the present document.

BRIEF DESCRIPTION OF DRAWINGS

[0012] FIG. 1 shows an example of partitioning a picture with luma coding tree units (CTUs).

[0013] FIG. 2 shows another example of partitioning a picture with luma CTUs.

[0014] FIG. 3 shows an example partitioning of a picture.

[0015] FIG. 4 shows another example partitioning of a picture.

[0016] FIG. 5 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[0017] FIG. 6 is a block diagram of an example hardware platform used for video processing.

[0018] FIG. 7 is a block diagram that illustrates a video coding system in accordance with some embodiments of the present disclosure.

[0019] FIG. 8 is a block diagram that illustrates an encoder in accordance with some embodiments of the present disclosure.

[0020] FIG. 9 is a block diagram that illustrates a decoder in accordance with some embodiments of the present disclosure.

[0021] FIGS. 10-12 show flowcharts for example methods of video processing.

DETAILED DESCRIPTION

[0022] Section headings are used in the present document for ease of understanding and do not limit the applicability of techniques and embodiments disclosed in each section only to that section. Furthermore, H.266 terminology is used in some description only for ease of understanding and not for limiting scope of the disclosed techniques. As such, the techniques described herein are applicable to other video codec protocols and designs also.

1. Initial discussion

[0023] This document is related to video coding technologies. Specifically, it is about signalling of subpictures, tiles, and slices. The ideas may be applied individually or in various combination, to any video coding standard or non-standard video codec that supports multi-layer video coding, e.g., the being-developed Versatile Video Coding (VVC).

2. Abbreviations

APS	Adaptation Parameter Set
AU	Access Unit
AUD	Access Unit Delimiter
AVC	Advanced Video Coding
CLVS	Coded Layer Video Sequence
CPB	Coded Picture Buffer
CRA	Clean Random Access
CTU	Coding Tree Unit
CVS	Coded Video Sequence
DPB	Decoded Picture Buffer
DPS	Decoding Parameter Set
EOB	End Of Bitstream
EOS	End Of Sequence
GDR	Gradual Decoding Refresh
HEVC	High Efficiency Video Coding
HRD	Hypothetical Reference Decoder
IDR	Instantaneous Decoding Refresh
JEM	Joint Exploration Model

MCTS	Motion-Constrained Tile Sets
NAL	Network Abstraction Layer
OLS	Output Layer Set
PH	Picture Header
PPS	Picture Parameter Set
PTL	Profile, Tier and Level
PU	Picture Unit
RBSP	Raw Byte Sequence Payload
SEI	Supplemental Enhancement Information
SPS	Sequence Parameter Set
SVC	Scalable Video Coding
VCL	Video Coding Layer
VPS	Video Parameter Set
VTM	VVC Test Model
VUI	Video Usability Information
VVC	Versatile Video Coding

3. Video coding introduction

[0024] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, the Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). The JVET meeting is concurrently held once every quarter, and the new coding standard is targeting at 50% bitrate reduction as compared to HEVC. The new video coding standard was officially named as Versatile Video Coding (VVC) in the April 2018 JVET meeting, and the first version of VVC test model (VTM) was released at that time. As there are continuous effort contributing

to VVC standardization, new coding techniques are being adopted to the VVC standard in every JVET meeting. The VVC working draft and test model VTM are then updated after every meeting. The VVC project is now aiming for technical completion (FDIS) at the July 2020 meeting.

3.1. Picture partitioning schemes in HEVC

[0025] HEVC includes four different picture partitioning schemes, namely regular slices, dependent slices, tiles, and Wavefront Parallel Processing (WPP), which may be applied for Maximum Transfer Unit (MTU) size matching, parallel processing, and reduced end-to-end delay.

[0026] Regular slices are similar as in H.264/AVC. Each regular slice is encapsulated in its own NAL unit, and in-picture prediction (intra sample prediction, motion information prediction, coding mode prediction) and entropy coding dependency across slice boundaries are disabled. Thus a regular slice can be reconstructed independently from other regular slices within the same picture (though there may still have interdependencies due to loop filtering operations).

[0027] The regular slice is the only tool that can be used for parallelization that is also available, in virtually identical form, in H.264/AVC. Regular slices based parallelization does not require much inter-processor or inter-core communication (except for inter-processor or inter-core data sharing for motion compensation when decoding a predictively coded picture, which is typically much heavier than inter-processor or inter-core data sharing due to in-picture prediction).

However, for the same reason, the use of regular slices can incur substantial coding overhead due to the bit cost of the slice header and due to the lack of prediction across the slice boundaries.

Further, regular slices (in contrast to the other tools mentioned below) also serve as the key mechanism for bitstream partitioning to match MTU size requirements, due to the in-picture independence of regular slices and that each regular slice is encapsulated in its own NAL unit. In many cases, the goal of parallelization and the goal of MTU size matching place contradicting demands to the slice layout in a picture. The realization of this situation led to the development of the parallelization tools mentioned below.

[0028] Dependent slices have short slice headers and allow partitioning of the bitstream at treeblock boundaries without breaking any in-picture prediction. Basically, dependent slices provide fragmentation of regular slices into multiple NAL units, to provide reduced end-to-end

delay by allowing a part of a regular slice to be sent out before the encoding of the entire regular slice is finished.

[0029] In WPP, the picture is partitioned into single rows of coding tree blocks (CTBs). Entropy decoding and prediction are allowed to use data from CTBs in other partitions. Parallel processing is possible through parallel decoding of CTB rows, where the start of the decoding of a CTB row is delayed by two CTBs, so to ensure that data related to a CTB above and to the right of the subject CTB is available before the subject CTB is being decoded. Using this staggered start (which appears like a wavefront when represented graphically), parallelization is possible with up to as many processors/cores as the picture contains CTB rows. Because in-picture prediction between neighboring treeblock rows within a picture is permitted, the required inter-processor/inter-core communication to enable in-picture prediction can be substantial. The WPP partitioning does not result in the production of additional NAL units compared to when it is not applied, thus WPP is not a tool for MTU size matching. However, if MTU size matching is required, regular slices can be used with WPP, with certain coding overhead.

[0030] Tiles define horizontal and vertical boundaries that partition a picture into tile columns and rows. Tile column runs from the top of a picture to the bottom of the picture. Likewise, tile row runs from the left of the picture to the right of the picture. The number of tiles in a picture can be derived simply as number of tile columns multiply by number of tile rows.

[0031] The scan order of CTBs is changed to be local within a tile (in the order of a CTB raster scan of a tile), before decoding the top-left CTB of the next tile in the order of tile raster scan of a picture. Similar to regular slices, tiles break in-picture prediction dependencies as well as entropy decoding dependencies. However, they do not need to be included into individual NAL units (same as WPP in this regard); hence tiles cannot be used for MTU size matching. Each tile can be processed by one processor/core, and the inter-processor/inter-core communication required for in-picture prediction between processing units decoding neighboring tiles is limited to conveying the shared slice header in cases a slice is spanning more than one tile, and loop filtering related sharing of reconstructed samples and metadata. When more than one tile or WPP segment is included in a slice, the entry point byte offset for each tile or WPP segment other than the first one in the slice is signaled in the slice header.

[0032] For simplicity, restrictions on the application of the four different picture partitioning schemes have been specified in HEVC. A given coded video sequence cannot include both tiles and wavefronts for most of the profiles specified in HEVC. For each slice and tile, either or both of the following conditions must be fulfilled: 1) all coded treeblocks in a slice belong to the same tile; 2) all coded treeblocks in a tile belong to the same slice. Finally, a wavefront segment contains exactly one CTB row, and when WPP is in use, if a slice starts within a CTB row, it must end in the same CTB row.

[0033] A recent amendment to HEVC is specified in the JCT-VC output document JCTVC-AC1005, J. Boyce, A. Ramasubramonian, R. Skupin, G. J. Sullivan, A. Tourapis, Y.-K. Wang (editors), "HEVC Additional Supplemental Enhancement Information (Draft 4)," Oct. 24, 2017, publicly available herein: http://phenix.int-evry.fr/jct/doc_end_user/documents/29_Macau/wg11/JCTVC-AC1005-v2.zip. With this amendment included, HEVC specifies three MCTS-related SEI messages, namely temporal MCTSs SEI message, MCTSs extraction information set SEI message, and MCTSs extraction information nesting SEI message.

[0034] The temporal MCTSs SEI message indicates existence of MCTSs in the bitstream and signals the MCTSs. For each MCTS, motion vectors are restricted to point to full-sample locations inside the MCTS and to fractional-sample locations that require only full-sample locations inside the MCTS for interpolation, and the usage of motion vector candidates for temporal motion vector prediction derived from blocks outside the MCTS is disallowed. This way, each MCTS may be independently decoded without the existence of tiles not included in the MCTS.

[0035] The MCTSs extraction information sets SEI message provides supplemental information that can be used in the MCTS sub-bitstream extraction (specified as part of the semantics of the SEI message) to generate a conforming bitstream for an MCTS set. The information consists of a number of extraction information sets, each defining a number of MCTS sets and containing RBSP bytes of the replacement VPSs, SPSs, and PPSs to be used during the MCTS sub-bitstream extraction process. When extracting a sub-bitstream according to the MCTS sub-bitstream extraction process, parameter sets (VPSs, SPSs, and PPSs) need to be rewritten or replaced, slice headers need to be slightly updated because one or all of the slice address related

syntax elements (including `first_slice_segment_in_pic_flag` and `slice_segment_address`) typically would need to have different values.

3.2. Partitioning of pictures in VVC

[0036] In VVC, A picture is divided into one or more tile rows and one or more tile columns. A tile is a sequence of CTUs that covers a rectangular region of a picture. The CTUs in a tile are scanned in raster scan order within that tile.

[0037] A slice consists of an integer number of complete tiles or an integer number of consecutive complete CTU rows within a tile of a picture.

[0038] Two modes of slices are supported, namely the raster-scan slice mode and the rectangular slice mode. In the raster-scan slice mode, a slice contains a sequence of complete tiles in a tile raster scan of a picture. In the rectangular slice mode, a slice contains either a number of complete tiles that collectively form a rectangular region of the picture or a number of consecutive complete CTU rows of one tile that collectively form a rectangular region of the picture. Tiles within a rectangular slice are scanned in tile raster scan order within the rectangular region corresponding to that slice.

[0039] A subpicture contains one or more slices that collectively cover a rectangular region of a picture.

[0040] FIG. 1 shows an example of raster-scan slice partitioning of a picture, where the picture is divided into 12 tiles and 3 raster-scan slices.

[0041] FIG. 2 shows an example of rectangular slice partitioning of a picture, where the picture is divided into 24 tiles (6 tile columns and 4 tile rows) and 9 rectangular slices.

[0042] FIG. 3 shows an example of a picture partitioned into tiles and rectangular slices, where the picture is divided into 4 tiles (2 tile columns and 2 tile rows) and 4 rectangular slices.

[0043] FIG. 4 shows an example of subpicture partitioning of a picture, where a picture is partitioned into 18 tiles, 12 on the left-hand side each covering one slice of 4 by 4 CTUs and 6 tiles on the right-hand side each covering 2 vertically-stacked slices of 2 by 2 CTUs, altogether resulting in 24 slices and 24 subpictures of varying dimensions (each slice is a subpicture).

3.3. Signalling of subpictures, tiles, and slices in VVC

[0044] In the latest VVC draft text, information of subpictures, includes subpicture layout (i.e., the number of subpictures for each picture and the position and size of each picture) and other

sequence-level subpicture information, is signalled in the SPS. The order of subpictures signalled in the SPS defines the subpicture index. A list of subpicture IDs, one for each subpicture, may be explicitly signalled, e.g., in the SPS or in the PPS.

[0045] Tiles in VVC are conceptually the same as in HEVC, i.e., each picture is partitioned into tile columns and tile rows, but with different syntax in the PPS for signalling of tiles.

[0046] In VVC, the slice mode is also signalled in the PPS. When the slice mode is the rectangular slice mode, the slice layout (i.e., the number of slices for each picture and the position and size of each slice) for each picture is signalled in the PPS. The order of the rectangular slices within a picture signalled in the PPS defines the picture-level slice index. The subpicture-level slice index is defined as the order of the slices within a subpicture in increasing order of their picture-level slice indices. The positions and sizes of the rectangular slices are signalled/derived based on either the subpicture positions and sizes that are signalled in the SPS (when each subpicture contains only one slice), or based on the tile positions and sizes that are signalled in the PPS (when a subpicture may contain more than one slice). When the slice mode is the raster-scan slice mode, similarly as in HEVC, the layout of slices within a picture is signalled in the slices themselves, with different details.

[0047] The SPS, PPS, and slice header syntax and semantics in the latest VVC draft text that are most relevant to the inventions herein are as follows.

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
...	
subpics_present_flag	u(1)
if(subpics_present_flag) {	
sps_num_subpics_minus1	u(8)
for(i = 0; i <= sps_num_subpics_minus1; i++) {	
subpic_ctu_top_left_x[i]	u(v)
subpic_ctu_top_left_y[i]	u(v)
subpic_width_minus1[i]	u(v)
subpic_height_minus1[i]	u(v)
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)

}	
}	
sps_subpic_id_present_flag	u(1)
if(sps_subpics_id_present_flag) {	
sps_subpic_id_signalling_present_flag	u(1)
if(sps_subpics_id_signalling_present_flag) {	
sps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= sps_num_subpics_minus1; i++)	
sps_subpic_id[i]	u(v)
}	
}	
...	
}	

7.4.3.3 Sequence parameter set RBSP semantics

...

subpics_present_flag equal to 1 specifies that subpicture parameters are present in in the SPS RBSP syntax. **subpics_present_flag** equal to 0 specifies that subpicture parameters are not present in the SPS RBSP syntax.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of **subpics_present_flag** equal to 1 in the RBSP of the SPSs.

sps_num_subpics_minus1 plus 1 specifies the number of subpictures. **sps_num_subpics_minus1** shall be in the range of 0 to 254. When not present, the value of **sps_num_subpics_minus1** is inferred to be equal to 0.

subpic_ctu_top_left_x[i] specifies horizontal position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_ctu_top_left_x[i]** is inferred to be equal to 0.

subpic_ctu_top_left_y[i] specifies vertical position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_ctu_top_left_y[i]** is inferred to be equal to 0.

subpic_width_minus1[i] plus 1 specifies the width of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_width_minus1[i]** is inferred to be equal to $\text{Ceil}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}) - 1$.

subpic_height_minus1[i] plus 1 specifies the height of the i-th subpicture in units of CtbSizeY. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}))$ bits. When not present, the value of **subpic_height_minus1[i]** is inferred to be equal to $\text{Ceil}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}) - 1$.

subpic_treated_as_pic_flag[i] equal to 1 specifies that the i-th subpicture of each coded picture in the CLVS is treated as a picture in the decoding process excluding in-loop filtering operations.

subpic_treated_as_pic_flag[i] equal to 0 specifies that the i-th subpicture of each coded picture in the CLVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of **subpic_treated_as_pic_flag[i]** is inferred to be equal to 0.

loop_filter_across_subpic_enabled_flag[i] equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the i-th subpicture in each coded picture in the CLVS.

loop_filter_across_subpic_enabled_flag[i] equal to 0 specifies that in-loop filtering operations are not performed across the boundaries of the i-th subpicture in each coded picture in the CLVS. When not present, the value of **loop_filter_across_subpic_enabled_pic_flag[i]** is inferred to be equal to 1.

It is a requirement of bitstream conformance that the following constraints apply:

- For any two subpictures subpicA and subpicB, when the subpicture index of subpicA is less than that of subpicB, any coded slice NAL unit of subPicA shall precede any coded slice NAL unit of subPicB in decoding order.
- The shapes of the subpictures shall be such that each subpicture, when decoded, shall have its entire left boundary and entire top boundary consisting of picture boundaries or consisting of boundaries of previously decoded subpictures.

sps_subpic_id_present_flag equal to 1 specifies that subpicture ID mapping is present in the SPS.

sps_subpic_id_present_flag equal to 0 specifies that subpicture ID mapping is not present in the SPS.

sps_subpic_id_signalling_present_flag equal to 1 specifies that subpicture ID mapping is signalled in the SPS. **sps_subpic_id_signalling_present_flag** equal to 0 specifies that subpicture ID mapping is not signalled in the SPS. When not present, the value of **sps_subpic_id_signalling_present_flag** is inferred to be equal to 0.

sps_subpic_id_len_minus1 plus 1 specifies the number of bits used to represent the syntax element **sps_subpic_id[i]**. The value of **sps_subpic_id_len_minus1** shall be in the range of 0 to 15, inclusive.

sps_subpic_id[i] specifies that subpicture ID of the i-th subpicture. The length of the **sps_subpic_id[i]** syntax element is **sps_subpic_id_len_minus1 + 1** bits. When not present, and when **sps_subpic_id_present_flag** equal to 0, the value of **sps_subpic_id[i]** is inferred to be equal to i, for each i in the range of 0 to **sps_num_subpics_minus1**, inclusive

...

7.3.2.4 Picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	Descriptor
...	
mixed_nalu_types_in_pic_flag	u(1)
pps_subpic_id_signalling_present_flag	u(1)
if(pps_subpics_id_signalling_present_flag) {	
pps_num_subpics_minus1	ue(v)
pps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= pps_num_subpic_minus1; i++)	
pps_subpic_id[i]	u(v)
}	
no_pic_partition_flag	u(1)
if(!no_pic_partition_flag) {	
pps_log2_ctu_size_minus5	u(2)
num_exp_tile_columns_minus1	ue(v)
num_exp_tile_rows_minus1	ue(v)
for(i = 0; i <= num_exp_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i <= num_exp_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
rect_slice_flag	u(1)
if(rect_slice_flag)	
single_slice_per_subpic_flag	u(1)
if(rect_slice_flag && !single_slice_per_subpic_flag) {	
num_slices_in_pic_minus1	ue(v)
tile_idx_delta_present_flag	u(1)
for(i = 0; i < num_slices_in_pic_minus1; i++) {	

slice_width_in_tiles_minus1[i]	ue(v)
slice_height_in_tiles_minus1[i]	ue(v)
if(slice_width_in_tiles_minus1[i] == 0 && slice_height_in_tiles_minus1[i] == 0) {	
num_slices_in_tile_minus1[i]	ue(v)
numSlicesInTileMinus1 = num_slices_in_tile_minus1[i]	
for(j = 0; j < numSlicesInTileMinus1; j++)	
slice_height_in_ctu_minus1[i++]	ue(v)
}	
if(tile_idx_delta_present_flag && i < num_slices_in_pic_minus1)	
tile_idx_delta[i]	se(v)
}	
}	
loop_filter_across_tiles_enabled_flag	u(1)
loop_filter_across_slices_enabled_flag	u(1)
}	
...	u(1)
}	

7.4.3.4 Picture parameter set RBSP semantics

...

pps_subpic_id_signalling_present_flag equal to 1 specifies that subpicture ID mapping is signalled in the PPS. **pps_subpic_id_signalling_present_flag** equal to 0 specifies that subpicture ID mapping is not signalled in the PPS. When **sps_subpic_id_present_flag** is 0 or **sps_subpic_id_signalling_present_flag** is equal to 1, **pps_subpic_id_signalling_present_flag** shall be equal to 0.

pps_num_subpics_minus1 plus 1 specifies the number of subpictures in the coded pictures referring to the PPS.

It is a requirement of bitstream conformance that the value of **pps_num_subpic_minus1** shall be equal to **sps_num_subpics_minus1**.

pps_subpic_id_len_minus1 plus 1 specifies the number of bits used to represent the syntax element **pps_subpic_id[i]**. The value of **pps_subpic_id_len_minus1** shall be in the range of 0 to 15, inclusive.

It is a requirement of bitstream conformance that the value of **pps_subpic_id_len_minus1** shall be the same for all PPSs that are referred to by coded pictures in a CLVS.

pps_subpic_id[i] specifies the subpicture ID of the i-th subpicture. The length of the pps_subpic_id[i] syntax element is pps_subpic_id_len_minus1 + 1 bits.

no_pic_partition_flag equal to 1 specifies that no picture partitioning applied to each picture referring to the PPS. no_pic_partition_flag equal to 0 specifies each picture referring to the PPS may be partitioned into more than one tile or slice.

It is a requirement of bitstream conformance that the value of no_pic_partition_flag shall be the same for all PPSs that are referred to by coded pictures within a CLVS.

It is a requirement of bitstream conformance that the value of no_pic_partition_flag shall not be equal to 1 when the value of sps_num_subpics_minus1 + 1 is greater than 1.

pps_log2_ctu_size_minus5 plus 5 specifies the luma coding tree block size of each CTU.

pps_log2_ctu_size_minus5 shall be equal to sps_log2_ctu_size_minus5.

num_exp_tile_columns_minus1 plus 1 specifies the number of explicitly provided tile column widths.

The value of num_exp_tile_columns_minus1 shall be in the range of 0 to PicWidthInCtbsY - 1, inclusive. When no_pic_partition_flag is equal to 1, the value of num_exp_tile_columns_minus1 is inferred to be equal to 0.

num_exp_tile_rows_minus1 plus 1 specifies the number of explicitly provided tile row heights. The value of num_exp_tile_rows_minus1 shall be in the range of 0 to PicHeightInCtbsY - 1, inclusive. When no_pic_partition_flag is equal to 1, the value of num_exp_tile_rows_minus1 is inferred to be equal to 0.

tile_column_width_minus1[i] plus 1 specifies the width of the i-th tile column in units of CTBs for i in the range of 0 to num_exp_tile_columns_minus1 - 1, inclusive.

tile_column_width_minus1[num_exp_tile_columns_minus1] is used to derive the width of the tile columns with index greater than or equal to num_exp_tile_columns_minus1 as specified in clause 6.5.1.

When not present, the value of tile_column_width_minus1[0] is inferred to be equal to PicWidthInCtbsY - 1.

tile_row_height_minus1[i] plus 1 specifies the height of the i-th tile row in units of CTBs for i in the range of 0 to num_exp_tile_rows_minus1 - 1, inclusive.

tile_row_height_minus1[num_exp_tile_rows_minus1] is used to derive the height of the tile rows with index greater than or equal to num_exp_tile_rows_minus1 as specified in clause 6.5.1. When not present, the value of tile_row_height_minus1[0] is inferred to be equal to PicHeightInCtbsY - 1.

rect_slice_flag equal to 0 specifies that tiles within each slice are in raster scan order and the slice information is not signalled in PPS. rect_slice_flag equal to 1 specifies that tiles within each slice cover a rectangular region of the picture and the slice information is signalled in the PPS. When not present,

rect_slice_flag is inferred to be equal to 1. When **subpics_present_flag** is equal to 1, the value of **rect_slice_flag** shall be equal to 1.

single_slice_per_subpic_flag equal to 1 specifies that each subpicture consists of one and only one rectangular slice. **single_slice_per_subpic_flag** equal to 0 specifies that each subpicture may consist one or more rectangular slices. When **subpics_present_flag** is equal to 0, **single_slice_per_subpic_flag** shall be equal to 0. When **single_slice_per_subpic_flag** is equal to 1, **num_slices_in_pic_minus1** is inferred to be equal to **sps_num_subpics_minus1**.

num_slices_in_pic_minus1 plus 1 specifies the number of rectangular slices in each picture referring to the PPS. The value of **num_slices_in_pic_minus1** shall be in the range of 0 to **MaxSlicesPerPicture** – 1, inclusive, where **MaxSlicesPerPicture** is specified in Annex A. When **no_pic_partition_flag** is equal to 1, the value of **num_slices_in_pic_minus1** is inferred to be equal to 0.

tile_idx_delta_present_flag equal to 0 specifies that **tile_idx_delta** values are not present in the PPS and that all rectangular slices in pictures referring to the PPS are specified in raster order according to the process defined in clause 6.5.1. **tile_idx_delta_present_flag** equal to 1 specifies that **tile_idx_delta** values may be present in the PPS and that all rectangular slices in pictures referring to the PPS are specified in the order indicated by the values of **tile_idx_delta**.

slice_width_in_tiles_minus1[i] plus 1 specifies the width of the i-th rectangular slice in units of tile columns. The value of **slice_width_in_tiles_minus1[i]** shall be in the range of 0 to **NumTileColumns** – 1, inclusive. When not present, the value of **slice_width_in_tiles_minus1[i]** is inferred as specified in clause 6.5.1.

slice_height_in_tiles_minus1[i] plus 1 specifies the height of the i-th rectangular slice in units of tile rows. The value of **slice_height_in_tiles_minus1[i]** shall be in the range of 0 to **NumTileRows** – 1, inclusive. When not present, the value of **slice_height_in_tiles_minus1[i]** is inferred as specified in clause 6.5.1.

num_slices_in_tile_minus1[i] plus 1 specifies the number of slices in the current tile for the case where the i-th slice contains a subset of CTU rows from a single tile. The value of **num_slices_in_tile_minus1[i]** shall be in the range of 0 to **RowHeight[tileY]** – 1, inclusive, where **tileY** is the tile row index containing the i-th slice. When not present, the value of **num_slices_in_tile_minus1[i]** is inferred to be equal to 0.

slice_height_in_ctu_minus1[i] plus 1 specifies the height of the i-th rectangular slice in units of CTU rows for the case where the i-th slice contains a subset of CTU rows from a single tile. The value of **slice_height_in_ctu_minus1[i]** shall be in the range of 0 to **RowHeight[tileY]** – 1, inclusive, where **tileY** is the tile row index containing the i-th slice.

tile_idx_delta[i] specifies the difference in tile index between the i-th rectangular slice and the (i + 1)-th rectangular slice. The value of **tile_idx_delta[i]** shall be in the range of $-\text{NumTilesInPic} + 1$ to $\text{NumTilesInPic} - 1$, inclusive. When not present, the value of **tile_idx_delta[i]** is inferred to be equal to 0. In all other cases, the value of **tile_idx_delta[i]** shall not be equal to 0.

loop_filter_across_tiles_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across tile boundaries in pictures referring to the PPS. **loop_filter_across_tiles_enabled_flag** equal to 0 specifies that in-loop filtering operations are not performed across tile boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **loop_filter_across_tiles_enabled_flag** is inferred to be equal to 1.

loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across slice boundaries in pictures referring to the PPS. **loop_filter_across_slice_enabled_flag** equal to 0 specifies that in-loop filtering operations are not performed across slice boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter, sample adaptive offset filter, and adaptive loop filter operations. When not present, the value of **loop_filter_across_slices_enabled_flag** is inferred to be equal to 0.

...

7.3.7.1 General slice header syntax

slice_header() {	Descriptor
...	
if(subpics_present_flag)	
slice_subpic_id	u(v)
if(rect_slice_flag NumTilesInPic > 1)	
slice_address	u(v)
if(!rect_slice_flag && NumTilesInPic > 1)	
num_tiles_in_slice_minus1	ue(v)
slice_type	ue(v)
...	
}	

7.4.8.1 General slice header semantics

...

slice_subpic_id specifies the subpicture identifier of the subpicture that contains the slice. If **slice_subpic_id** is present, the value of the variable **SubPicIdx** is derived to be such that **SubpicIdList[SubPicIdx]** is equal to **slice_subpic_id**. Otherwise (**slice_subpic_id** is not present), the variable **SubPicIdx** is derived to be equal to 0. The length of **slice_subpic_id**, in bits, is derived as follows:

- If **sps_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **sps_subpic_id_len_minus1 + 1**.
- Otherwise, if **ph_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **ph_subpic_id_len_minus1 + 1**.
- Otherwise, if **pps_subpic_id_signalling_present_flag** is equal to 1, the length of **slice_subpic_id** is equal to **pps_subpic_id_len_minus1 + 1**.
- Otherwise, the length of **slice_subpic_id** is equal to $\text{Ceil}(\text{Log}_2(\text{sps\_num\_subpics\_minus1} + 1))$.

slice_address specifies the slice address of the slice. When not present, the value of **slice_address** is inferred to be equal to 0.

If **rect_slice_flag** is equal to 0, the following applies:

- The slice address is the raster scan tile index.
- The length of **slice_address** is $\text{Ceil}(\text{Log}_2(\text{NumTilesInPic}))$ bits.
- The value of **slice_address** shall be in the range of 0 to **NumTilesInPic** – 1, inclusive.

Otherwise (**rect_slice_flag** is equal to 1), the following applies:

- The slice address is the slice index of the slice within the **SubPicIdx**-th subpicture.
- The length of **slice_address** is $\text{Ceil}(\text{Log}_2(\text{NumSlicesInSubpic[SubPicIdx]}))$ bits.
- The value of **slice_address** shall be in the range of 0 to **NumSlicesInSubpic[SubPicIdx]** – 1, inclusive.

It is a requirement of bitstream conformance that the following constraints apply:

- If **rect_slice_flag** is equal to 0 or **subpics_present_flag** is equal to 0, the value of **slice_address** shall not be equal to the value of **slice_address** of any other coded slice NAL unit of the same coded picture.
- Otherwise, the pair of **slice_subpic_id** and **slice_address** values shall not be equal to the pair of **slice_subpic_id** and **slice_address** values of any other coded slice NAL unit of the same coded picture.
- When **rect_slice_flag** is equal to 0, the slices of a picture shall be in increasing order of their **slice_address** values.

- The shapes of the slices of a picture shall be such that each CTU, when decoded, shall have its entire left boundary and entire top boundary consisting of a picture boundary or consisting of boundaries of previously decoded CTU(s).

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.

The variable **NumCtuInCurrSlice**, which specifies the number of CTUs in the current slice, and the list **CtbAddrInCurrSlice[i]**, for **i** ranging from 0 to **NumCtuInCurrSlice** – 1, inclusive, specifying the picture raster scan address of the **i**-th CTB within the slice, are derived as follows:

```

if( rect_slice_flag ) {
    picLevelSliceIdx = SliceSubpicToPicIdx[ SubPicIdx ][ slice_address ]
    NumCtuInCurrSlice = NumCtuInSlice[ picLevelSliceIdx ]
    for( i = 0; i < NumCtuInCurrSlice; i++ )
        CtbAddrInCurrSlice[ i ] = CtbAddrInSlice[ picLevelSliceIdx ][ i ]
} else {
    NumCtuInCurrSlice = 0
    for( tileIdx = slice_address; tileIdx <= slice_address + num_tiles_in_slice_minus1[ i ];
        tileIdx++ ) {
        tileX = tileIdx % NumTileColumns
        tileY = tileIdx / NumTileColumns
        for( ctbY = tileRowBd[ tileY ]; ctbY < tileRowBd[ tileY + 1 ]; ctbY++ ) {
            for( ctbX = tileColBd[ tileX ]; ctbX < tileColBd[ tileX + 1 ]; ctbX++ ) {
                CtbAddrInCurrSlice[ NumCtuInCurrSlice ] = ctbY * PicWidthInCtb + ctbX
                NumCtuInCurrSlice++
            }
        }
    }
}

```

(115)

The variables **SubPicLeftBoundaryPos**, **SubPicTopBoundaryPos**, **SubPicRightBoundaryPos**, and **SubPicBotBoundaryPos** are derived as follows:

```

if( subpic_treated_as_pic_flag[ SubPicIdx ] ) {
    SubPicLeftBoundaryPos = subpic_ctu_top_left_x[ SubPicIdx ] * CtbSizeY
    SubPicRightBoundaryPos = Min( pic_width_max_in_luma_samples – 1,

```

$$\begin{aligned}
 & (\text{subpic\_ctu\_top\_left\_x}[\text{SubPicIdx}] + \text{subpic\_width\_minus1}[\text{SubPicIdx}] + 1) * \text{CtbSizeY} \\
 & - 1) \\
 & \text{SubPicTopBoundaryPos} = \text{subpic\_ctu\_top\_left\_y}[\text{SubPicIdx}] * \text{CtbSizeY} \quad (116) \\
 & \text{SubPicBotBoundaryPos} = \text{Min}(\text{pic\_height\_max\_in\_luma\_samples} - 1, \\
 & (\text{subpic\_ctu\_top\_left\_y}[\text{SubPicIdx}] + \text{subpic\_height\_minus1}[\text{SubPicIdx}] + 1) * \text{CtbSizeY} \\
 & - 1) \\
 & \}
 \end{aligned}$$

...

4. Examples of technical problems solved by solutions herein

[0048] The existing designs for signaling of subpictures, tiles, and slices in VVC have the following problems:

- 1) The coding of `sps_num_subpics_minus1` is `u(8)`, which disallows more than 256 subpictures per picture. However, in certain applications, the maximum number of subpictures per picture may need to be greater than 256.
- 2) It is allowed to have `subpics_present_flag` equal to 0 and `sps_subpic_id_present_flag` equal to 1. However, this does not make sense as `subpics_present_flag` equal to 0 means that the CLVS has no information on subpictures at all.
- 3) A list of subpicture IDs may be signalled in picture headers (PHs), one for each of the subpictures. However, when the list of subpicture IDs is signalled in PHs, and when a subset of the subpictures is extracted from the bitstream, all the PHs would all need to be changed. This is undesirable.
- 4) Currently when subpicture IDs are indicated to be explicitly signalled, by `sps_subpic_id_present_flag` (or the name of the syntax element is changed to `subpic_ids_explicitly_signalled_flag`) equal to 1, subpicture IDs may be not signalled anywhere. This is problematic as subpicture IDs need to be explicitly signalled in either the SPS or the PPS when subpicture IDs are indicated to be explicitly signalled.
- 5) When subpicture IDs are not explicitly signalled, the slice header syntax element `slice_subpic_id` still need to be signalled as long as `subpics_present_flag` is equal to 1, including when `sps_num_subpics_minus1` is equal to 0. However, the length of

- slice_subpic_id is currently specified as $\text{Ceil}(\text{Log2}(\text{sps\_num\_subpics\_minus1} + 1))$ bits, which would be 0 bits when `sps_num_subpics_minus1` is equal to 0. This is problematic, as the length of any present syntax elements cannot be 0 bits.
- 6) The subpicture layout, including the number of subpictures and their sizes and positions, keeps unchanging for the entire CLVS. Even when the subpicture IDs are not explicitly signalled in the SPS or the PPS, the subpicture ID length still needs to be signalled, for the subpicture ID syntax element in slice headers.
 - 7) Whenever `rect_slice_flag` is equal to 1, the syntax element `slice_address` is signalled in the slice header and specifies the slice index within the subpicture containing the slice, including when the number of slices within the subpicture (i.e., `NumSlicesInSubpic[ SubPicIdx ]`) is equal to 1. However, currently, when `rect_slice_flag` is equal to 1, the length of `slice_address` is specified to be $\text{Ceil}(\text{Log2}(\text{NumSlicesInSubpic[SubPicIdx]}))$ bits, which would be 0 bits when `NumSlicesInSubpic[ SubPicIdx ]` is equal to 1. This is problematic, as the length of any present syntax elements cannot be 0 bits.
 - 8) There is redundancy between the syntax elements `no_pic_partition_flag` and `pps_num_subpics_minus1`, although the latest VVC text has the following constraint: When `sps_num_subpics_minus1` is greater than 0, the value of `no_pic_partition_flag` shall be equal to 1.
 - 9) Within a CLVS, the subpicture ID value for a particular subpicture position or index may change from picture to picture. When this happens, in principle, the subpicture cannot use inter prediction by referring to a reference picture in the same layer. However, currently, there lacks a constraint to prohibit this in the current VVC specification.
 - 10) In current VVC design, a reference picture could be a picture in a different layer to support multiple applications, e.g., scalable video coding and multi-view video coding. If subpicture is present in different layers, whether to allow or disallow the inter-layer prediction needs to be studied.

5. Example techniques and embodiments

[0049] To solve the above problems, and others, methods as summarized below are disclosed.

The inventions should be considered as examples to explain the general concepts and should not

be interpreted in a narrow way. Furthermore, these inventions can be applied individually or combined in any manner.

[0050] To solve the first problem, change the coding of `sps_num_subpics_minus1` from `u(8)` to `ue(v)`, to enable more than 256 subpictures per picture.

- a. Furthermore, the value of `sps_num_subpics_minus1` is restricted to be in the range of 0 to $\text{Ceil}(\text{pic\_width\_max\_in\_luma\_samples} \div \text{CtbSizeY}) * \text{Ceil}(\text{pic\_height\_max\_in\_luma\_samples} \div \text{CtbSizeY}) - 1$, inclusive.
 - b. Furthermore, the number of subpictures per picture is further restricted in the definition of levels.
- 2) To solve the second problem, condition the signalling of the syntax element `sps_subpic_id_present_flag` on "`if( subpics_present_flag )`", i.e., the syntax element `sps_subpic_id_present_flag` is not signalled when `subpics_present_flag` is equal to 0, and infer the value of `sps_subpic_id_present_flag` to be equal to 0 when it is not present.
- a. Alternatively, the syntax element `sps_subpic_id_present_flag` is still signalled when `subpics_present_flag` is equal to 0, but the value is required to be equal to 0 when `subpics_present_flag` is equal to 0.
 - b. Furthermore, the names of the syntax elements `subpics_present_flag` and `sps_subpic_id_present_flag` are changed to be `subpic_info_present_flag` and `subpic_ids_explicitly_signalled_flag`, respectively.
- 3) To solve the third problem, the signalling of subpicture IDs in the PH syntax is removed. Consequently, the list `SubpicIdList[ i ]`, for `i` in the range of 0 to `sps_num_subpics_minus1`, inclusive, is derived as follows:
- ```

for(i = 0; i <= sps_num_subpics_minus1; i++)
 if(subpic_ids_explicitly_signalled_flag)
 SubpicIdList[i] = subpic_ids_in_pps_flag ? pps_subpic_id[i] :
sps_subpic_id[i]
 else
 SubpicIdList[i] = i

```
- 4) To solve the fourth problem, subpicture IDs are signalled either in the SPS or in the PPS when subpictures are indicated to be explicitly signalled.

- a. This is realized by add the following constraint: If `subpic_ids_explicitly_signalled_flag` is 0 or `subpic_ids_in_sps_flag` is equal to 1, `subpic_ids_in_pps_flag` shall be equal to 0. Otherwise (`subpic_ids_explicitly_signalled_flag` is 1 and `subpic_ids_in_sps_flag` is equal to 0), `subpic_ids_in_pps_flag` shall be equal to 1.
- 5) To solve the fifth and sixth problems, the length of subpicture IDs is signalled in the SPS regardless of the value of the SPS flag `sps_subpic_id_present_flag` (or renamed to `subpic_ids_explicitly_signalled_flag`), although the length may also be signalled in the PPS when subpicture IDs are explicitly signalled in the PPS to avoid parsing dependency of PPS on SPS. In this case, the length also specifies the length of the subpicture IDs in the slice headers, even subpicture IDs are not explicitly signalled in the SPS or PPS. Consequently, the length of the `slice_subpic_id`, when present, is also specified by the subpicture ID length signalled in the SPS.
- 6) Alternatively, to solve the fifth and sixth problems, a flag is added to the SPS syntax, the value of 1 for which to specify the existence of the subpicture ID length in the SPS syntax. This flag is present is independent of the value of the flag indicating whether subpicture IDs are explicitly signalled in the SPS or PPS. The value of this flag may be equal to either 1 or 0 when `subpic_ids_explicitly_signalled_flag` is equal to 0, but the value of flag must be equal to 1 when `subpic_ids_explicitly_signalled_flag` is equal to 1. When this flag is equal to 0, i.e., the subpicture length is not present, the length of `slice_subpic_id` is specified to be  $\text{Max}(\text{Ceil}(\text{Log}_2(\text{sps\_num\_subpics\_minus1} + 1)), 1)$  bits (as opposed to be  $\text{Ceil}(\text{Log}_2(\text{sps\_num\_subpics\_minus1} + 1))$  bits in the latest VVC draft text).
  - a. Alternatively, this flag is present only when `subpic_ids_explicitly_signalled_flag` is equal to 0, and when `subpic_ids_explicitly_signalled_flag` is equal to 1 the value of this flag is inferred to be equal to 1.
- 7) To solve the seventh problem, when `rect_slice_flag` is equal to 1, the length of `slice_address` is specified to be  $\text{Max}(\text{Ceil}(\text{Log}_2(\text{NumSlicesInSubpic}[\text{SubPicIdx}])), 1)$  bits.

- a. Alternatively, further more, when `rect_slice_flag` is equal to 0, the length of `slice_address` is specified to  $\text{Max}(\text{Ceil}(\text{Log2}(\text{NumTilesInPic})), 1)$  bits, as opposed to be  $\text{Ceil}(\text{Log2}(\text{NumTilesInPic}))$  bits.
- 8) To solve the eighth problem, condition the signalling of `no_pic_partition_flag` on "`if( subpic_ids_in_pps_flag && pps_num_subpics_minus1 > 0 )`", and add the following inference: When not present, the value of `no_pic_partition_flag` is inferred to be equal to 1.
- a. Alternatively, move the subpicture ID syntax (all the four syntax elements) after the tile and slice syntax in the PPS, e.g., immediately before the syntax element `entropy_coding_sync_enabled_flag`, and then condition the signalling of `pps_num_subpics_minus1` on "`if( no_pic_partition_flag )`".
- 9) To solve the ninth problem, the following constraint is specified: For each particular subpicture index (or equivalently, subpicture position), when the subpicture ID value changes at a picture `picA` compared to that in the previous picture in decoding order in the same layer as `picA`, unless `picA` is the first picture of the CLVS, the subpicture at `picA` shall only contain coded slice NAL units with `nal_unit_type` equal to `IDR_W_RADL`, `IDR_N_LP`, or `CRA_NUT`.
- a. Alternatively, the above constraint applies only for subpicture indices for which the value of the `subpic_treated_as_pic_flag[ i ]` is equal to 1.
  - b. Alternatively, for both item 9 and 9a, change "`IDR_W_RADL`, `IDR_N_LP`, or `CRA_NUT`" to "`IDR_W_RADL`, `IDR_N_LP`, `CRA_NUT`, `RSV_IRAP_11`, or `RSV_IRAP_12`".
  - c. Alternatively, the subpicture at `picA` may contain other types of coded slice NAL units, however, these coded slice NAL unit only use one or more of intra prediction, Intra Block Copy (IBC) prediction, and palette mode prediction.
  - d. Alternatively, a first video unit (such as a slice, a tile, a block, etc.) in a subpicture of `picA` may refer to a second video unit in a previous picture. It is constrained that the second video unit and the first video unit may be in subpictures with the same subpicture index, although the subpicture IDs of the two may be different. A

subpicture index is a unique number assigned to a subpicture, which cannot be changed in a CLVS.

- 10) For a particular subpicture index (or equivalently, subpicture position), indications of which of subpictures, identified by layer ID values together with subpicture indices or subpicture ID values, are allowed to be used as reference pictures may be signaled in the bitstream.
- 11) For multiple-layer cases, when certain conditions (e.g., which may depend on the number of subpictures, location of subpictures) are satisfied, inter-layer prediction (ILR) from a subpicture in a different layer is allowed, while when the certain conditions are not satisfied, ILR is disabled.
  - a. In one example, even when two subpictures in two layers are with the same subpicture index value but different subpicture ID values, the inter-layer prediction may be still allowed when certain conditions are satisfied.
    - i. In one example, the certain conditions are “if the two layers are associated with different view order index/view order ID values”.
  - b. It may be constrained that a first subpicture in a first layer and second subpicture in a second layer must be at the collocated positions and/or rational width/height if the two subpictures have the same subpicture index.
  - c. It may be constrained that a first subpicture in a first layer and second subpicture in a second layer must be at the collocated positions and/or rational width/height if the first subpicture can refer to the second reference subpicture.
- 12) An indication of whether a current subpicture may use Inter-Layer Prediction (ILP) from sample values and/or other values, e.g., motion information and/or coding modes information, associated with regions or subpictures of reference layers is signaled in the bitstream, such as in VPS/DPS/SPS/PPS/APS/sequence header/picture header.
  - a. In one example, the reference regions or subpictures of reference layers are the ones that contains at least one collocated sample of a sample within the current subpicture.
  - b. In one example, the reference regions or subpictures of reference layers are outside the collocated region of the current subpicture.

- c. In one example, such an indication is signaled in one or more SEI messages.
- d. In one example, such an indication is signaled regardless of whether the reference layers have multiple subpictures or not, and, when multiple subpictures are present in one or more of the reference layers, regardless of whether the partitioning of the pictures into subpictures aligned with the current picture such that each subpicture in the current picture a corresponding subpicture in a reference picture that covers the collocated region, and furthermore, regardless of whether the corresponding/collocated subpicture has the same subpicture ID value as the current subpicture.

6. Embodiments

*Below are some example embodiments for all the invention aspects except item 8 summarized above in Section 5, which can be applied to the VVC specification. The changed texts are based on the latest VVC text in JVET-P2001-v14. Most relevant parts that have been added or modified are shown in **underline, bolded and italicized text**, and the most relevant removed parts are highlighted in enclosed in bolded double brackets, e.g., **[[a]]** indicates that “a” has been removed.. There are some other changes that are editorial in nature and thus not highlighted.*

6.1. First embodiment

7.3.2.3 Sequence parameter set RBSP syntax

|                                                  |                     |
|--------------------------------------------------|---------------------|
| seq_parameter_set_rbsp( ) {                      | <b>Descriptor</b>   |
| ...                                              |                     |
| sps_log2_ctu_size_minus5                         | u(2)                |
| <b><u>subpic info present flag</u></b>           | <b><u>u(1)</u></b>  |
| if( <b><u>subpic info present flag</u></b> ) {   |                     |
| sps_num_subpics_minus1                           | <b><u>ue(v)</u></b> |
| for( i = 0; i <= sps_num_subpics_minus1; i++ ) { |                     |
| subpic_ctu_top_left_x[ i ]                       | u(v)                |
| subpic_ctu_top_left_y[ i ]                       | u(v)                |
| subpic_width_minus1[ i ]                         | u(v)                |
| subpic_height_minus1[ i ]                        | u(v)                |
| subpic_treated_as_pic_flag[ i ]                  | u(1)                |

|                                                     |              |
|-----------------------------------------------------|--------------|
| loop_filter_across_subpic_enabled_flag[ i ]         | u(1)         |
| }                                                   |              |
| <u>sps_subpic_id_len_minus1</u>                     | <u>ue(v)</u> |
| <u>subpic_ids_explicitly_signalled_flag</u>         | u(1)         |
| if( <u>subpic_ids_explicitly_signalled_flag</u> ) { |              |
| <u>subpic_ids_in_sps_flag</u>                       | u(1)         |
| if( <u>subpic_ids_in_sps_flag</u> )                 |              |
| for( i = 0; i <= sps_num_subpics_minus1; i++ )      |              |
| sps_subpic_id[ i ]                                  | u(v)         |
| }                                                   |              |
| }                                                   |              |
| ...                                                 |              |
| }                                                   |              |

#### 7.4.3.3 Sequence parameter set RBSP semantics

...

subpic\_info\_present\_flag equal to 1 specifies that subpicture information is present.

subpic\_info\_present\_flag equal to 0 specifies that subpicture information is not present. The values of sps\_ref\_pic\_resampling\_enabled\_flag and subpic\_info\_present\_flag shall not be both equal to 1.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of subpic\_info\_present\_flag equal to 1 in the SPSs.

sps\_num\_subpics\_minus1 plus 1 specifies the number of subpictures. The value of

sps\_num\_subpics\_minus1 shall be in the range of 0 to

$\text{Ceil}(\text{pic\_width\_max\_in\_luma\_samples} \div \text{CtbSizeY}) *$

$\text{Ceil}(\text{pic\_height\_max\_in\_luma\_samples} \div \text{CtbSizeY}) - 1$ , inclusive. Noet that the maximum number of subpictures can be further restricted in the level definitions. When not present, the value of

sps\_num\_subpics\_minus1 is inferred to be equal to 0.

subpic\_ctu\_top\_left\_x[ i ] specifies horizontal position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}))$  bits. When not present, the value of subpic\_ctu\_top\_left\_x[ i ] is inferred to be equal to 0.

**subpic\_ctu\_top\_left\_y[ i ]** specifies vertical position of top left CTU of i-th subpicture in unit of CtbSizeY. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}))$  bits. When not present, the value of **subpic\_ctu\_top\_left\_y[ i ]** is inferred to be equal to 0.

**subpic\_width\_minus1[ i ]** plus 1 specifies the width of the i-th subpicture in units of CtbSizeY. The length of the syntax element is  $\text{Ceil}(\text{Log2}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}))$  bits. When not present, the value of **subpic\_width\_minus1[ i ]** is inferred to be equal to  $\text{Ceil}(\text{pic\_width\_max\_in\_luma\_samples} / \text{CtbSizeY}) - 1$ .

**subpic\_height\_minus1[ i ]** plus 1 specifies the height of the i-th subpicture in units of CtbSizeY. The length of the syntax element is  $\text{Ceil}(\text{Log2}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}))$  bits. When not present, the value of **subpic\_height\_minus1[ i ]** is inferred to be equal to  $\text{Ceil}(\text{pic\_height\_max\_in\_luma\_samples} / \text{CtbSizeY}) - 1$ .

**subpic\_treated\_as\_pic\_flag[ i ]** equal to 1 specifies that the i-th subpicture of each coded picture in the CLVS is treated as a picture in the decoding process excluding in-loop filtering operations.

**subpic\_treated\_as\_pic\_flag[ i ]** equal to 0 specifies that the i-th subpicture of each coded picture in the CLVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of **subpic\_treated\_as\_pic\_flag[ i ]** is inferred to be equal to 0.

**loop\_filter\_across\_subpic\_enabled\_flag[ i ]** equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the i-th subpicture in each coded picture in the CLVS.

**loop\_filter\_across\_subpic\_enabled\_flag[ i ]** equal to 0 specifies that in-loop filtering operations are not performed across the boundaries of the i-th subpicture in each coded picture in the CLVS. When not present, the value of **loop\_filter\_across\_subpic\_enabled\_pic\_flag[ i ]** is inferred to be equal to 1.

It is a requirement of bitstream conformance that the following constraints apply:

- For any two subpictures subpicA and subpicB, when the subpicture index of subpicA is less than that of subpicB, any coded slice NAL unit of subPicA shall precede any coded slice NAL unit of subPicB in decoding order.
- The shapes of the subpictures shall be such that each subpicture, when decoded, shall have its entire left boundary and entire top boundary consisting of picture boundaries or consisting of boundaries of previously decoded subpictures.

**sps\_subpic\_id\_len\_minus1 plus 1 specifies the number of bits used to represent the explicitly signalled subpicture IDs as well as the slice\_subpic\_id syntax element in the slice header. The value of sps\_subpic\_id\_len\_minus1 shall be in the range of 0 to 15, inclusive.**

subpic\_ids\_explicitly\_signalled\_flag equal to 1 specifies that a set of subpicture ID, one for each subpicture, is explicitly signalled for each subpicture either in the SPS or the PPS.

subpic\_ids\_explicitly\_signalled\_flag equal to 0 specifies that no subpicture IDs are explicitly signalled in the SPS or the PPS. When not present, the value of subpic\_ids\_explicitly\_signalled\_flag is inferred to be equal to 0.

subpic\_ids\_in\_sps\_flag equal to 1 specifies that a subpicture ID for each subpicture is explicitly signalled in the SPS. subpic\_ids\_in\_sps\_flag equal to 0 specifies that no subpicture IDs are signalled in the SPS. When not present, the value of subpic\_ids\_in\_sps\_flag is inferred to be equal to 0.

sps\_subpic\_id[ i ] specifies the subpicture ID of the i-th subpicture. The length of the sps\_subpic\_id[ i ] syntax element is sps\_subpic\_id\_len\_minus1 + 1 bits.

...

#### 7.3.2.4 Picture parameter set RBSP syntax

| pic_parameter_set_rbsp( ) {                         | Descriptor |
|-----------------------------------------------------|------------|
| ...                                                 |            |
| <b>mixed_nalu_types_in_pic_flag</b>                 | u(1)       |
| <b><u>subpic_ids_in_pps_flag</u></b>                | u(1)       |
| if( <b><u>subpic_ids_in_pps_flag</u></b> ) {        |            |
| <b>pps_num_subpics_minus1</b>                       | ue(v)      |
| <b>pps_subpic_id_len_minus1</b>                     | ue(v)      |
| for( i = 0; i <= pps_num_subpic_minus1; i++ )       |            |
| <b>pps_subpic_id[ i ]</b>                           | u(v)       |
| }                                                   |            |
| <b>no_pic_partition_flag</b>                        | u(1)       |
| if( !no_pic_partition_flag ) {                      |            |
| <b>pps_log2_ctu_size_minus5</b>                     | u(2)       |
| <b>num_exp_tile_columns_minus1</b>                  | ue(v)      |
| <b>num_exp_tile_rows_minus1</b>                     | ue(v)      |
| for( i = 0; i <= num_exp_tile_columns_minus1; i++ ) |            |
| <b>tile_column_width_minus1[ i ]</b>                | ue(v)      |
| for( i = 0; i <= num_exp_tile_rows_minus1; i++ )    |            |
| <b>tile_row_height_minus1[ i ]</b>                  | ue(v)      |
| if( NumTilesInPic > 1 )                             |            |

|                                                                                            |       |
|--------------------------------------------------------------------------------------------|-------|
| <b>rect_slice_flag</b>                                                                     | u(1)  |
| if( rect_slice_flag )                                                                      |       |
| <b>single_slice_per_subpic_flag</b>                                                        | u(1)  |
| if( rect_slice_flag && !single_slice_per_subpic_flag ) {                                   |       |
| <b>num_slices_in_pic_minus1</b>                                                            | ue(v) |
| <b>tile_idx_delta_present_flag</b>                                                         | u(1)  |
| for( i = 0; i < num_slices_in_pic_minus1; i++ ) {                                          |       |
| <b>slice_width_in_tiles_minus1[ i ]</b>                                                    | ue(v) |
| <b>slice_height_in_tiles_minus1[ i ]</b>                                                   | ue(v) |
| if( slice_width_in_tiles_minus1[ i ] == 0 &&<br>slice_height_in_tiles_minus1[ i ] == 0 ) { |       |
| <b>num_slices_in_tile_minus1[ i ]</b>                                                      | ue(v) |
| for( j = 0; j < num_slices_in_tile_minus1[ i ]; j++ )                                      |       |
| <b>slice_height_in_ctu_minus1[ i++ ]</b>                                                   | ue(v) |
| }                                                                                          |       |
| if( tile_idx_delta_present_flag && i < num_slices_in_pic_minus1 )                          |       |
| <b>tile_idx_delta[ i ]</b>                                                                 | se(v) |
| }                                                                                          |       |
| }                                                                                          |       |
| <b>loop_filter_across_tiles_enabled_flag</b>                                               | u(1)  |
| <b>loop_filter_across_slices_enabled_flag</b>                                              | u(1)  |
| }                                                                                          |       |
| ...                                                                                        |       |
| }                                                                                          |       |

#### 7.4.3.4 Picture parameter set RBSP semantics

...

subpic\_ids\_in\_pps\_flag equal to 1 specifies that a subpicture ID for each subpicture is explicitly signalled in the PPS. subpic\_ids\_in\_pps\_flag equal to 0 specifies that no subpicture IDs are signalled in the PPS. If subpic\_ids\_explicitly\_signalled\_flag is 0 or subpic\_ids\_in\_sps\_flag is equal to 1, subpic\_ids\_in\_pps\_flag shall be equal to 0. Otherwise (subpic\_ids\_explicitly\_signalled\_flag is 1 and subpic\_ids\_in\_sps\_flag is equal to 0), subpic\_ids\_in\_pps\_flag shall be equal to 1.

pps\_num\_subpics\_minus1 shall be equal to sps\_num\_subpics\_minus1.

**pps\_subpic\_id\_len\_minus1** shall be equal to **sps\_subpic\_id\_len\_minus1**.

**pps\_subpic\_id[ i ]** specifies the subpicture ID of the i-th subpicture. The length of the pps\_subpic\_id[ i ] syntax element is  $\text{pps\_subpic\_id\_len} - \text{minus1} + 1$  bits.

The list *SubpicIdList*[*i*], for *i* in the range of 0 to *sps\_num\_subpics\_minus1*, inclusive, is derived as follows:

$$\begin{aligned} & \underline{\text{for}(i = 0; i \leq \text{sps\_num\_subpics\_minus1}; i++)} \\ & \quad \underline{\text{if}(\text{subpic\_ids\_explicitly\_signalled\_flag})} \\ & \quad \quad \underline{\text{SubpicIdList}[i] = \text{subpic\_ids\_in\_pps\_flag} ? \text{pps\_subpic\_id}[i] : \text{sps\_subpic\_id}[i]} \\ & \quad (76) \\ & \quad \underline{\text{else}} \\ & \quad \quad \underline{\text{SubpicIdList}[i] = i} \end{aligned}$$

It is a requirement of bitstream conformance that, for any i and j in the range of 0 to sps\_num\_subpics\_minus1, inclusive, when i is less than j, SubpicIdList[ i ] shall be less than SubpicIdList[ j ].

...

**rect\_slice\_flag** equal to 0 specifies that tiles within each slice are in raster scan order and the slice information is not signalled in PPS. **rect\_slice\_flag** equal to 1 specifies that tiles within each slice cover a rectangular region of the picture and the slice information is signalled in the PPS. When not present, **rect\_slice\_flag** is inferred to be equal to 1. When **subpic\_info\_present\_flag** is equal to 1, the value of **rect\_slice\_flag** shall be equal to 1.

**single\_slice\_per\_subpic\_flag** equal to 1 specifies that each subpicture consists of one and only one rectangular slice. **single\_slice\_per\_subpic\_flag** equal to 0 specifies that each subpicture may consist one or more rectangular slices. When **subpic\_info\_present\_flag** is equal to 0, **single\_slice\_per\_subpic\_flag** shall be equal to 0. When **single\_slice\_per\_subpic\_flag** is equal to 1, **num\_slices\_in\_pic\_minus1** is inferred to be equal to **sps\_num\_subpics\_minus1**.

...

### 7.3.7.1 General slice header syntax

|                                              |                   |
|----------------------------------------------|-------------------|
| slice_header( ) {                            | <b>Descriptor</b> |
| ...                                          |                   |
| if( <i><u>subpic info present flag</u></i> ) |                   |
| <b>slice_subpic_id</b>                       | u(v)              |
| if( rect_slice_flag    NumTilesInPic > 1 )   |                   |
| <b>slice_address</b>                         | u(v)              |

|                                             |       |
|---------------------------------------------|-------|
| if( !rect_slice_flag && NumTilesInPic > 1 ) |       |
| num_tiles_in_slice_minus1                   | ue(v) |
| slice_type                                  | ue(v) |
| ...                                         |       |
| }                                           |       |

#### 7.4.8.1 General slice header semantics

...

**slice\_subpic\_id** specifies the subpicture ID of the subpicture that contains the slice. *The length of slice\_subpic\_id is sps\_subpic\_id\_len\_minus1 + 1 bits.*

When not present, the value of slice\_subpic\_id is inferred to be equal to 0.

The variable SubPicIdx is derived to be the value such that SubPicIdList[ SubPicIdx ] is equal to slice\_subpic\_id.

**slice\_address** specifies the slice address of the slice. When not present, the value of slice\_address is inferred to be equal to 0.

If rect\_slice\_flag is equal to 0, the following applies:

- The slice address is the raster scan tile index.
- The length of slice\_address is  $\text{Ceil}(\text{Log}_2(\text{NumTilesInPic}))$  bits.
- The value of slice\_address shall be in the range of 0 to NumTilesInPic – 1, inclusive.

Otherwise (rect\_slice\_flag is equal to 1), the following applies:

- The slice address is the subpicture-level slice index of the slice.
- The length of slice\_address is *Max( Ceil( Log2( NumSlicesInSubpic[ SubPicIdx ] ) ), 1 )* bits.
- The value of slice\_address shall be in the range of 0 to NumSlicesInSubpic[ SubPicIdx ] – 1, inclusive.

It is a requirement of bitstream conformance that the following constraints apply:

- If rect\_slice\_flag is equal to 0 or *subpic\_info\_present\_flag* is equal to 0, the value of slice\_address shall not be equal to the value of slice\_address of any other coded slice NAL unit of the same coded picture.
- Otherwise, the pair of slice\_subpic\_id and slice\_address values shall not be equal to the pair of slice\_subpic\_id and slice\_address values of any other coded slice NAL unit of the same coded picture.
- When rect\_slice\_flag is equal to 0, the slices of a picture shall be in increasing order of their slice\_address values.

- The shapes of the slices of a picture shall be such that each CTU, when decoded, shall have its entire left boundary and entire top boundary consisting of a picture boundary or consisting of boundaries of previously decoded CTU(s).

...

**[0051]** FIG. 5 is a block diagram showing an example video processing system 500 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 500. The system 500 may include input 502 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 502 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

**[0052]** The system 500 may include a coding component 504 that may implement the various coding or encoding methods described in the present document. The coding component 504 may reduce the average bitrate of video from the input 502 to the output of the coding component 504 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 504 may be either stored, or transmitted via a communication connected, as represented by the component 506. The stored or communicated bitstream (or coded) representation of the video received at the input 502 may be used by the component 508 for generating pixel values or displayable video that is sent to a display interface 510. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as “coding” operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

**[0053]** Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various

electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

**[0054]** FIG. 6 is a block diagram of a video processing apparatus 600. The apparatus 600 may be used to implement one or more of the methods described herein. The apparatus 600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 600 may include one or more processors 602, one or more memories 604 and video processing hardware 606. The processor(s) 602 may be configured to implement one or more methods described in the present document. The memory (memories) 604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 606 may be used to implement, in hardware circuitry, some techniques described in the present document.

**[0055]** FIG. 7 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

**[0056]** As shown in FIG. 7, video coding system 100 may include a source device 110 and a destination device 120. Source device 110 generates encoded video data which may be referred to as a video encoding device. Destination device 120 may decode the encoded video data generated by source device 110 which may be referred to as a video decoding device.

**[0057]** Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

**[0058]** Video source 112 may include a source such as a video capture device, an interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources. The video data may comprise one or more pictures. Video encoder 114 encodes the video data from video source 112 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. I/O interface 116 may include a modulator/demodulator (modem) and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via I/O interface 116 through network 130a. The

encoded video data may also be stored onto a storage medium/server 130b for access by destination device 120.

**[0059]** Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

**[0060]** I/O interface 126 may include a receiver and/or a modem. I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/ server 130b. Video decoder 124 may decode the encoded video data. Display device 122 may display the decoded video data to a user. Display device 122 may be integrated with the destination device 120, or may be external to destination device 120 which be configured to interface with an external display device.

**[0061]** Video encoder 114 and video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding(VVM) standard and other current and/or further standards.

**[0062]** FIG. 8 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG. 7.

**[0063]** Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 8, video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

**[0064]** The functional components of video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra prediction unit 206, a residual generation unit 207, a transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

**[0065]** In other examples, video encoder 200 may include more, fewer, or different functional components. In an example, predication unit 202 may include an intra block copy(IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[0066] Furthermore, some components, such as motion estimation unit 204 and motion compensation unit 205 may be highly integrated, but are represented in the example of FIG. 8 separately for purposes of explanation.

[0067] Partition unit 201 may partition a picture into one or more video blocks. Video encoder 200 and video decoder 300 may support various video block sizes.

[0068] Mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra- or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some example, Mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication signal and an intra predication signal. Mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[0069] To perform inter prediction on a current video block, motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from buffer 213 to the current video block. Motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from buffer 213 other than the picture associated with the current video block.

[0070] Motion estimation unit 204 and motion compensation unit 205 may perform different operations for a current video block, for example, depending on whether the current video block is in an I slice, a P slice, or a B slice.

[0071] In some examples, motion estimation unit 204 may perform uni-directional prediction for the current video block, and motion estimation unit 204 may search reference pictures of list 0 or list 1 for a reference video block for the current video block. Motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. Motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video

block of the current block based on the reference video block indicated by the motion information of the current video block.

**[0072]** In other examples, motion estimation unit 204 may perform bi-directional prediction for the current video block, motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. Motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. Motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

**[0073]** In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

**[0074]** In some examples, motion estimation unit 204 may do not output a full set of motion information for the current video. Rather, motion estimation unit 204 may signal the motion information of the current video block with reference to the motion information of another video block. For example, motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

**[0075]** In one example, motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

**[0076]** In another example, motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[0077] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode signaling.

[0078] Intra prediction unit 206 may perform intra prediction on the current video block. When intra prediction unit 206 performs intra prediction on the current video block, intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[0079] Residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block(s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[0080] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and residual generation unit 207 may not perform the subtracting operation.

[0081] Transform processing unit 208 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[0082] After transform processing unit 208 generates a transform coefficient video block associated with the current video block, quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[0083] Inverse quantization unit 210 and inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. Reconstruction unit 212 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current block for storage in the buffer 213.

**[0084]** After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

**[0085]** Entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When entropy encoding unit 214 receives the data, entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

**[0086]** FIG. 9 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG. 7.

**[0087]** The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 8, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

**[0088]** In the example of FIG. 9, video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an inverse transformation unit 305, and a reconstruction unit 306 and a buffer 307. Video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200 (FIG. 8).

**[0089]** Entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). Entropy decoding unit 301 may decode the entropy coded video data, and from the entropy decoded video data, motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. Motion compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode.

**[0090]** Motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

**[0091]** Motion compensation unit 302 may use interpolation filters as used by video encoder 200 during encoding of the video block to calculate interpolated values for sub-integer pixels of a

reference block. Motion compensation unit 302 may determine the interpolation filters used by video encoder 200 according to received syntax information and use the interpolation filters to produce predictive blocks.

**[0092]** Motion compensation unit 302 may use some of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence.

**[0093]** Intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. Inverse quantization unit 303 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. Inverse transform unit 303 applies an inverse transform.

**[0094]** Reconstruction unit 306 may sum the residual blocks with the corresponding prediction blocks generated by motion compensation unit 202 or intra-prediction unit 303 to form decoded blocks. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in buffer 307, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

**[0095]** FIGS. 10-11 show example methods that can implement the technical solution described above in, for example, the embodiments shown in FIGS. 5-9.

**[0096]** FIG. 10 shows a flowchart for an example method 1000 of video processing. The method 1000 includes, at operation 1010, performing a conversion between a video comprising multiple pictures comprising one or more subpictures and a bitstream of the video, the bitstream conforming to a format rule that specifies whether, in case that a current subpicture in a current picture has a current identifier that is different from an identifier of a previous subpicture in a previous picture at a same position as the current subpicture, the current subpicture is disallowed to refer to the previous subpicture for inter prediction.

**[0097]** FIG. 11 shows a flowchart for an example method 1100 of video processing. The method 1100 includes, at operation 1110, performing a conversion between a video comprising a video region and a bitstream of the video comprising multiple coding layers, the bitstream conforming to a format rule that specifies whether an inter-layer prediction (ILP) between video regions in different coding layers of the multiple coding layers is allowed based on a condition.

**[0098]** FIG. 12 shows a flowchart for an example method 1200 of video processing. The method 1200 includes, at operation 1210, performing a conversion between a video comprising a current video region and a bitstream of the video comprising multiple coding layers, the bitstream conforming to a format rule that specifies the bitstream includes an indication of whether an inter-layer prediction (ILP) between the current video region and a video region in a reference layer is allowed.

**[0099]** A listing of solutions preferred by some embodiments is provided next.

**[00100]** 1. A method of video processing, comprising performing a conversion between a video comprising multiple pictures comprising one or more subpictures and a bitstream of the video, wherein the bitstream conforms to a format rule that specifies whether, in case that a current subpicture in a current picture has a current identifier that is different from an identifier of a previous subpicture in a previous picture at a same position as the current subpicture, the current subpicture is disallowed to refer to the previous subpicture for inter prediction.

**[00101]** 2. The method of solution 1, wherein the current identifier is a current subpicture identifier or a current subpicture position.

**[00102]** 3. The method of solution 2, wherein the current subpicture is disallowed to refer to the previous subpicture for inter prediction due to the current identifier being different from the identifier of the previous subpicture.

**[00103]** 4. The method of solution 3, wherein the current subpicture is coded using intra-coding due to the current identifier being different from the identifier of the previous subpicture.

**[00104]** 5. The method of solution 3 or 4, wherein the current subpicture comprises only one or more coded slices of an instantaneous decoding refresh (IDR) subpicture or a clean random access (CRA) subpicture.

**[00105]** 6. The method of solution 3 or 4, wherein the current subpicture comprises only one or more coded slices of an intra random access point (IRAP) subpicture.

- [00106] 7. The method of solution 3 or 4, wherein the current subpicture comprises only coded slice network abstraction layer (NAL) units with one or more of a set of predetermined NAL unit types.
- [00107] 8. The method of solution 7, wherein the set of predetermined NAL unit types comprises IDR\_W\_RADL, IDR\_N\_LP, and CRA\_NUT.
- [00108] 9. The method of solution 7, wherein the set of predetermined NAL unit types comprises IDR\_W\_RADL, IDR\_N\_LP, CRA\_NUT, RSV\_IRAP\_11, and RSV\_IRAP\_12.
- [00109] 10. The method of solution 3 or 4, wherein the bitstream includes a syntax element indicating that the subpicture is treated as a picture.
- [00110] 11. The method of solution 3 or 4, wherein the current subpicture comprises coded slice network abstraction layer (NAL) units that use one or more of intra prediction, intra block copy (IBC) prediction, and palette mode prediction.
- [00111] 12. The method of solution 2, wherein a first video unit in the current subpicture refers to a second video unit in the previous subpicture, wherein a subpicture index of the current subpicture is identical to a subpicture index of the previous subpicture, and wherein the subpicture index is a number assigned to a subpicture that cannot be changed in a coded layer video sequence (CLVS).
- [00112] 13. The method of solution 12, wherein the subpicture identifier of the current subpicture is identical to the subpicture identifier of the previous subpicture.
- [00113] 14. The method of solution 12, wherein the subpicture identifier of the current subpicture is different from the subpicture identifier of the previous subpicture.
- [00114] 15. The method of solution 2, wherein the current subpicture refers to the previous subpicture and an indication of the current subpicture is signaled in the bitstream due to the current subpicture being identified by a layer identifier value and a subpicture index or the subpicture identifier.
- [00115] 16. A method of video processing, comprising performing a conversion between a video comprising a video region and a bitstream of the video comprising multiple coding layers, wherein the bitstream conforms to a format rule, and wherein the format rule specifies whether an inter-layer prediction (ILP) between video regions in different coding layers of the multiple coding layers is allowed based on a condition.

[00116] 17. The method of solution 16, wherein the video region is a subpicture, and wherein the inter-layer prediction is allowed.

[00117] 18. The method of solution 17, wherein two subpictures in the different coding layers comprise an identical subpicture index value and different subpicture identifier values.

[00118] 19. The method of solution 18, wherein the condition specifies that the two layers are associated with different view order indices or different view order identifier values.

[00119] 20. The method of solution 17, wherein a first subpicture and a second subpicture in the different coding layers are at a collocated position or have a rational height or width due to the first subpicture and the second subpicture have an identical subpicture index.

[00120] 21. The method of solution 17, wherein a first subpicture and a second subpicture in the different coding layers are at a collocated position or have a rational height or width due to the first subpicture referring to the second subpicture.

[00121] 22. A method of video processing, comprising performing a conversion between a video comprising a current video region and a bitstream of the video comprising multiple coding layers, wherein the bitstream conforms to a format rule, and wherein the format rule specifies the bitstream includes an indication of whether an inter-layer prediction (ILP) between the current video region and a video region in a reference layer is allowed.

[00122] 23. The method of solution 22, wherein the video region is a subpicture.

[00123] 24. The method of solution 23, wherein the indication is signaled in a video parameter set (VPS), a decoding parameter set (DPS), a sequence parameter set (SPS), a picture parameter set (PPS), an adaptation parameter set (APS), a sequence header, or a picture header.

[00124] 25. The method of solution 23, wherein the video region in the reference layer comprises at least one sample that is collocated with a sample of the current video region.

[00125] 26. The method of solution 23, wherein the video region in the reference layer is outside a collocated region of the current video region.

[00126] 27. The method of solution 23, wherein the indication is signaled in one or more supplemental enhancement information (SEI) messages.

[00127] 28. The method of solution 23, wherein the indication is signaled regardless of whether the reference layer comprises multiple subpictures.

[00128] 29. The method of solution 23, wherein the reference layer comprises multiple subpictures, and wherein the indication is signaled regardless of whether a partitioning of a picture into multiple subpictures aligns with a current picture such that each subpicture in the reference layer is collocated with a corresponding subpicture in the current picture.

[00129] 30. The method of any of solutions 1 to 29, wherein the conversion comprises decoding the video from the bitstream.

[00130] 31. The method of any of solutions 1 to 29, wherein the conversion comprises encoding the video into the bitstream.

[00131] 32. A method of storing a bitstream representing a video to a computer-readable recording medium, comprising generating the bitstream from the video according to a method described in any one or more of solutions 1 to 29; and writing the bitstream to the computer-readable recording medium.

[00132] 33. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of solutions 1 to 32.

[00133] 34. A computer-readable medium having instructions stored thereon, the instructions, when executed, causing a processor to implement a method recited in one or more of solutions 1 to 32.

[00134] 35. A computer readable medium that stores the bitstream generated according to any one or more of solutions 1 to 32.

[00135] 36. A video processing apparatus for storing a bitstream, wherein the video processing apparatus is configured to implement a method recited in any one or more of solutions 1 to 32.

[00136] Another listing of solutions preferred by some embodiments is provided next.

[00137] P1. A method of video processing, comprising performing a conversion between a picture of a video and a coded representation of the video, wherein a number of subpictures in the picture is included in the coded representation as a field whose bitwidth is dependent on a value of the number of subpictures.

[00138] P2. The method of solution P1, wherein the field represents the number of subpictures using a codeword.

[00139] P3. The method of solution P2, wherein the codeword comprises a Golomb codeword.

**[00140]** P4. The method of any of solutions P1 to P3, wherein the value of the number of subpictures is restricted to be less than or equal to an integer number of coding tree blocks that fit within the picture.

**[00141]** P5. The method of any of solutions P1 to P4, wherein the field is dependent on a coding level associated with the coded representation.

**[00142]** P6. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to omit a syntax element indicative of subpicture identifiers due to the video region not comprising any subpictures.

**[00143]** P7. The method of solution P6, wherein the coded representation includes a field having a 0 value indicating that the video region is not comprising any subpictures.

**[00144]** P8. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to omit identifiers of subpictures in the video region at a video region header level in the coded representation.

**[00145]** P9. The method of solution P8, wherein the coded representation identifies subpictures numerically according to an order in which the subpictures are listed in the video region header.

**[00146]** P10. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include identifiers of subpictures and/or a length of the identifiers of subpictures in the video region at a sequence parameter set level or a picture parameter set level.

**[00147]** P11. The method of solution P10, wherein the length is included in the picture parameter set level.

**[00148]** P12. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, wherein the format rule specifies to include a field in the coded representation at a video sequence level to indicate whether a subpicture identifier length field is included in the coded representation at the video sequence level.

**[00149]** P13. The method of solution P12, wherein the format rule specifies to set the field to “1” in case that another field in the coded representation indicates that a length identifier for the video region is included in the coded representation.

**[00150]** P14. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, and wherein the format rule specifies to include an indication in the coded representation to indicate whether the video region can be used as a reference picture.

**[00151]** P15. The method of solution P14, wherein the indication comprises a layer ID and an index or an ID value associated with the video region.

**[00152]** P16. A method of video processing, comprising performing a conversion between a video region of a video and a coded representation of the video, wherein the coded representation conforms to a format rule, and wherein the format rule specifies to include an indication in the coded representation to indicate whether the video region may use inter-layer prediction (ILP) from a plurality of sample values associated with video regions of reference layers.

**[00153]** P17. The method of solution P16, wherein the indication is included at a sequence level, a picture level or a video level.

**[00154]** P18. The method of solution P16, wherein the video regions of the reference layers comprise at least one collocated sample of a sample within the video region.

**[00155]** P19. The method of solution P16, wherein the indication is included in one or more supplemental enhancement information (SEI) messages.

**[00156]** P20. The method of any of above claims, wherein the video region comprises a subpicture of the video.

**[00157]** P21. The method of any of above claims, wherein the conversion comprises parsing and decoding the coded representation to generate the video.

**[00158]** P22. The method of any of above claims, wherein the conversion comprises encoding the video to generate the coded representation.

**[00159]** P23. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of solutions P1 to P22.

**[00160]** P24. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of solutions P1 to P22.

**[00161]** P25. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of solutions P1 to P22.

**[00162]** In some embodiments, a bitstream generated according to an above-described method may be stored on a computer-readable medium.

**[00163]** The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

**[00164]** A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A

computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

**[00165]** The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

**[00166]** Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

**[00167]** While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features

from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

**[00168]** Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

**[00169]** Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

WHAT IS CLAIMED IS:

1. A method of video processing, comprising:  
performing a conversion between a video comprising multiple pictures comprising one or more subpictures and a bitstream of the video,  
wherein the bitstream conforms to a format rule that specifies whether, in case that a current subpicture in a current picture has a current identifier that is different from an identifier of a previous subpicture in a previous picture at a same position as the current subpicture, the current subpicture is disallowed to refer to the previous subpicture for inter prediction.
2. The method of claim 1, wherein the current identifier is a current subpicture identifier or a current subpicture position.
3. The method of claim 2, wherein the current subpicture is disallowed to refer to the previous subpicture for inter prediction due to the current identifier being different from the identifier of the previous subpicture.
4. The method of claim 3, wherein the current subpicture is coded using intra-coding due to the current identifier being different from the identifier of the previous subpicture.
5. The method of claim 3 or 4, wherein the current subpicture comprises only one or more coded slices of an instantaneous decoding refresh (IDR) subpicture or a clean random access (CRA) subpicture.
6. The method of claim 3 or 4, wherein the current subpicture comprises only one or more coded slices of an intra random access point (IRAP) subpicture.
7. The method of claim 3 or 4, wherein the current subpicture comprises only coded slice network abstraction layer (NAL) units with one or more of a set of predetermined NAL unit types.
8. The method of claim 7, wherein the set of predetermined NAL unit types comprises IDR\_W\_RADL, IDR\_N\_LP, and CRA\_NUT.

9. The method of claim 7, wherein the set of predetermined NAL unit types comprises IDR\_W\_RADL, IDR\_N\_LP, CRA\_NUT, RSV\_IRAP\_11, and RSV\_IRAP\_12.
10. The method of claim 3 or 4, wherein the bitstream includes a syntax element indicating that the subpicture is treated as a picture.
11. The method of claim 3 or 4, wherein the current subpicture comprises coded slice network abstraction layer (NAL) units that use one or more of intra prediction, intra block copy (IBC) prediction, and palette mode prediction.
12. The method of claim 2, wherein a first video unit in the current subpicture refers to a second video unit in the previous subpicture, wherein a subpicture index of the current subpicture is identical to a subpicture index of the previous subpicture, and wherein the subpicture index is a number assigned to a subpicture that cannot be changed in a coded layer video sequence (CLVS).
13. The method of claim 12, wherein the subpicture identifier of the current subpicture is identical to the subpicture identifier of the previous subpicture.
14. The method of claim 12, wherein the subpicture identifier of the current subpicture is different from the subpicture identifier of the previous subpicture.
15. The method of claim 2, wherein the current subpicture refers to the previous subpicture and an indication of the current subpicture is signaled in the bitstream due to the current subpicture being identified by a layer identifier value and a subpicture index or the subpicture identifier.
16. A method of video processing, comprising:
  - performing a conversion between a video comprising a video region and a bitstream of the video comprising multiple coding layers,
  - wherein the bitstream conforms to a format rule, and
  - wherein the format rule specifies whether an inter-layer prediction (ILP) between video regions in different coding layers of the multiple coding layers is allowed based on a condition.

17. The method of claim 16, wherein the video region is a subpicture, and wherein the inter-layer prediction is allowed.
18. The method of claim 17, wherein two subpictures in the different coding layers comprise an identical subpicture index value and different subpicture identifier values.
19. The method of claim 18, wherein the condition specifies that the two layers are associated with different view order indices or different view order identifier values.
20. The method of claim 17, wherein a first subpicture and a second subpicture in the different coding layers are at a collocated position or have a rational height or width due to the first subpicture and the second subpicture have an identical subpicture index.
21. The method of claim 17, wherein a first subpicture and a second subpicture in the different coding layers are at a collocated position or have a rational height or width due to the first subpicture referring to the second subpicture.
22. A method of video processing, comprising:  
performing a conversion between a video comprising a current video region and a bitstream of the video comprising multiple coding layers,  
wherein the bitstream conforms to a format rule, and  
wherein the format rule specifies the bitstream includes an indication of whether an inter-layer prediction (ILP) between the current video region and a video region in a reference layer is allowed.
23. The method of claim 22, wherein the video region is a subpicture.
24. The method of claim 23, wherein the indication is signaled in a video parameter set (VPS), a decoding parameter set (DPS), a sequence parameter set (SPS), a picture parameter set (PPS), an adaptation parameter set (APS), a sequence header, or a picture header.
25. The method of claim 23, wherein the video region in the reference layer comprises at least one sample that is collocated with a sample of the current video region.

26. The method of claim 23, wherein the video region in the reference layer is outside a collocated region of the current video region.
27. The method of claim 23, wherein the indication is signaled in one or more supplemental enhancement information (SEI) messages.
28. The method of claim 23, wherein the indication is signaled regardless of whether the reference layer comprises multiple subpictures.
29. The method of claim 23, wherein the reference layer comprises multiple subpictures, and wherein the indication is signaled regardless of whether a partitioning of a picture into multiple subpictures aligns with a current picture such that each subpicture in the reference layer is collocated with a corresponding subpicture in the current picture.
30. The method of any of claims 1 to 29, wherein the conversion comprises decoding the video from the bitstream.
31. The method of any of claims 1 to 29, wherein the conversion comprises encoding the video into the bitstream.
32. A method of storing a bitstream representing a video to a computer-readable recording medium, comprising:  
generating the bitstream from the video according to a method described in any one or more of claims 1 to 29; and  
writing the bitstream to the computer-readable recording medium.
33. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of claims 1 to 32.
34. A computer-readable medium having instructions stored thereon, the instructions, when executed, causing a processor to implement a method recited in one or more of claims 1 to 32.
35. A computer readable medium that stores the bitstream generated according to any one or more of claims 1 to 32.

36. A video processing apparatus for storing a bitstream, wherein the video processing apparatus is configured to implement a method recited in any one or more of claims 1 to 32.

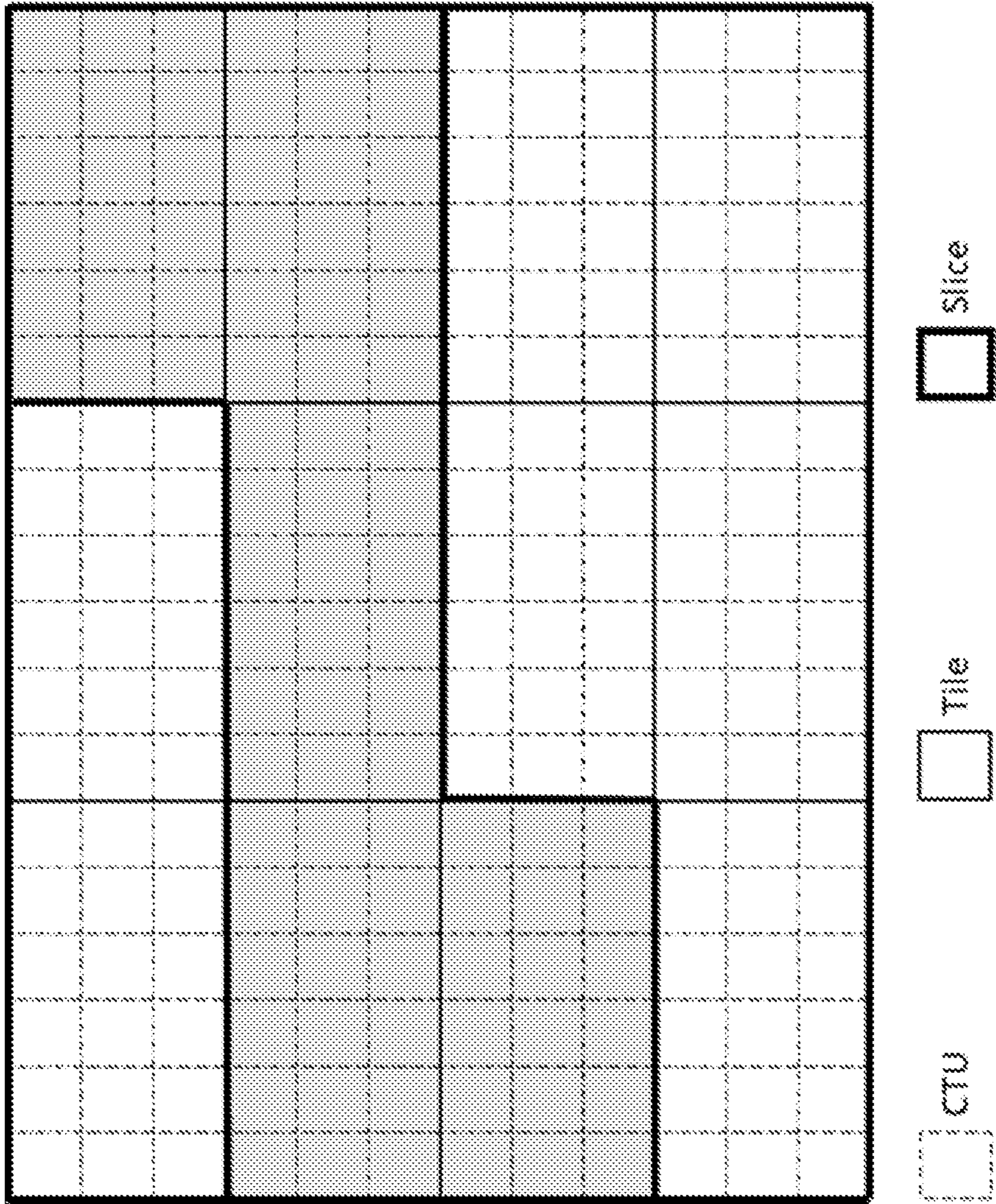


FIG. 1

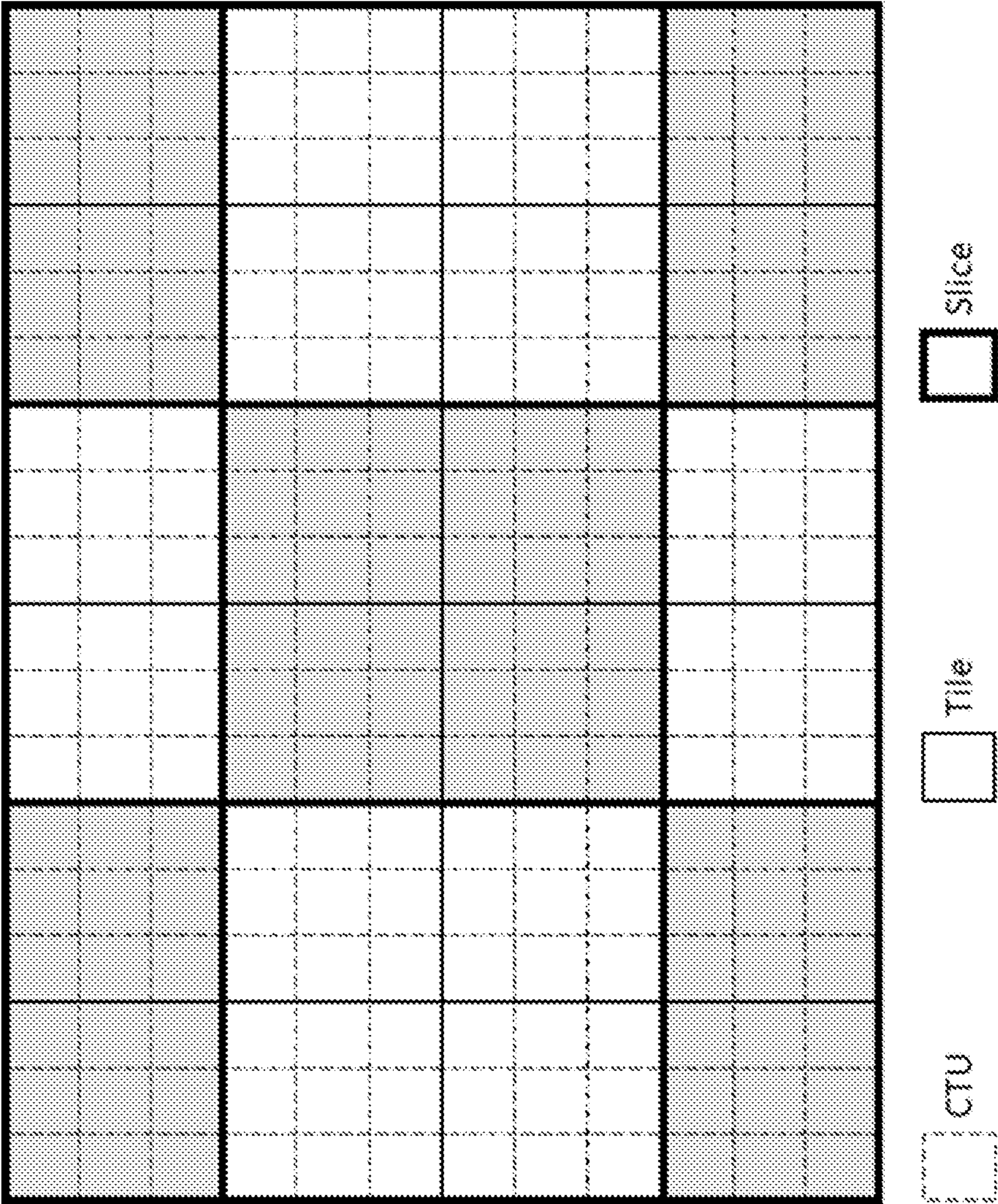


FIG. 2

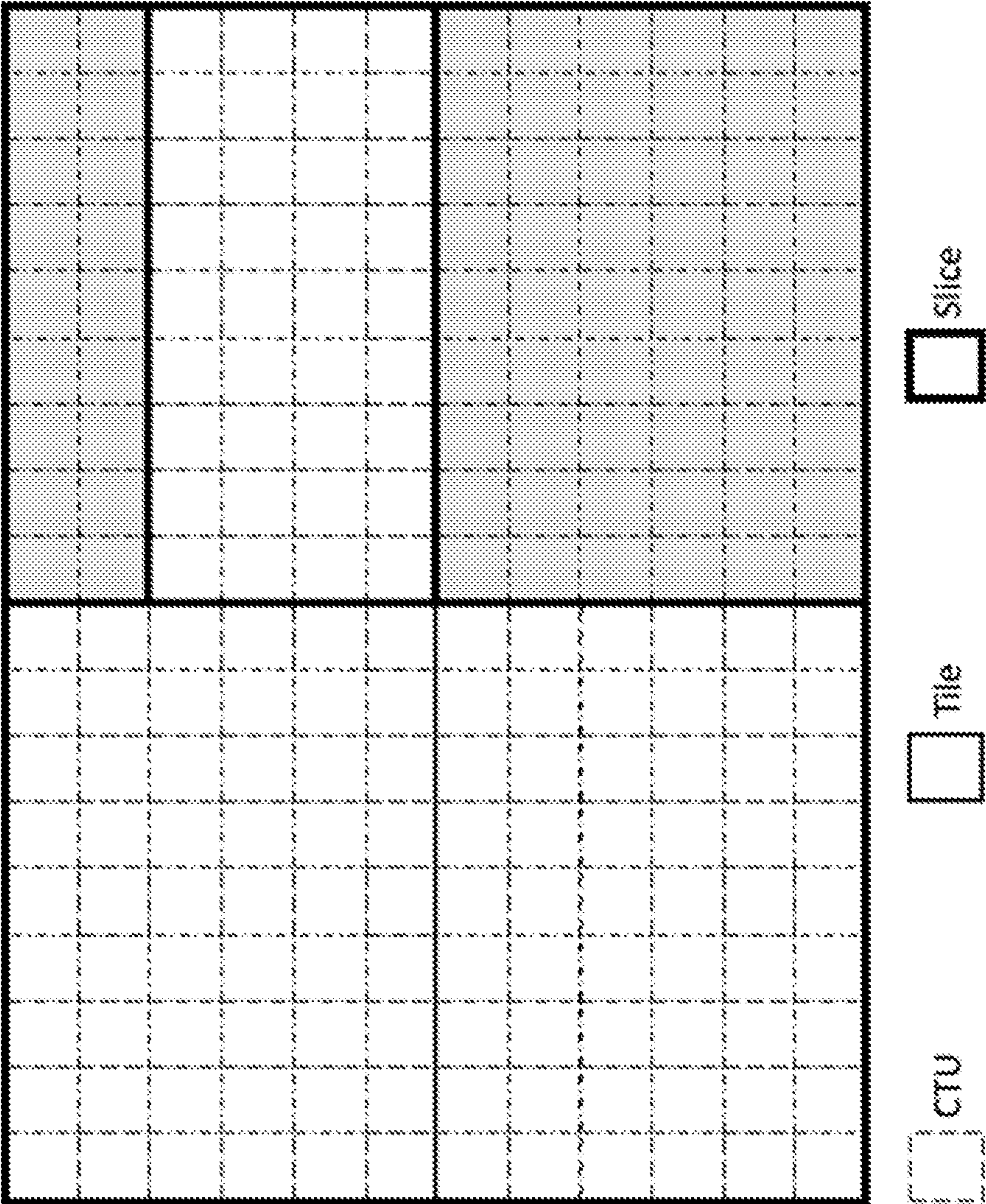


FIG. 3

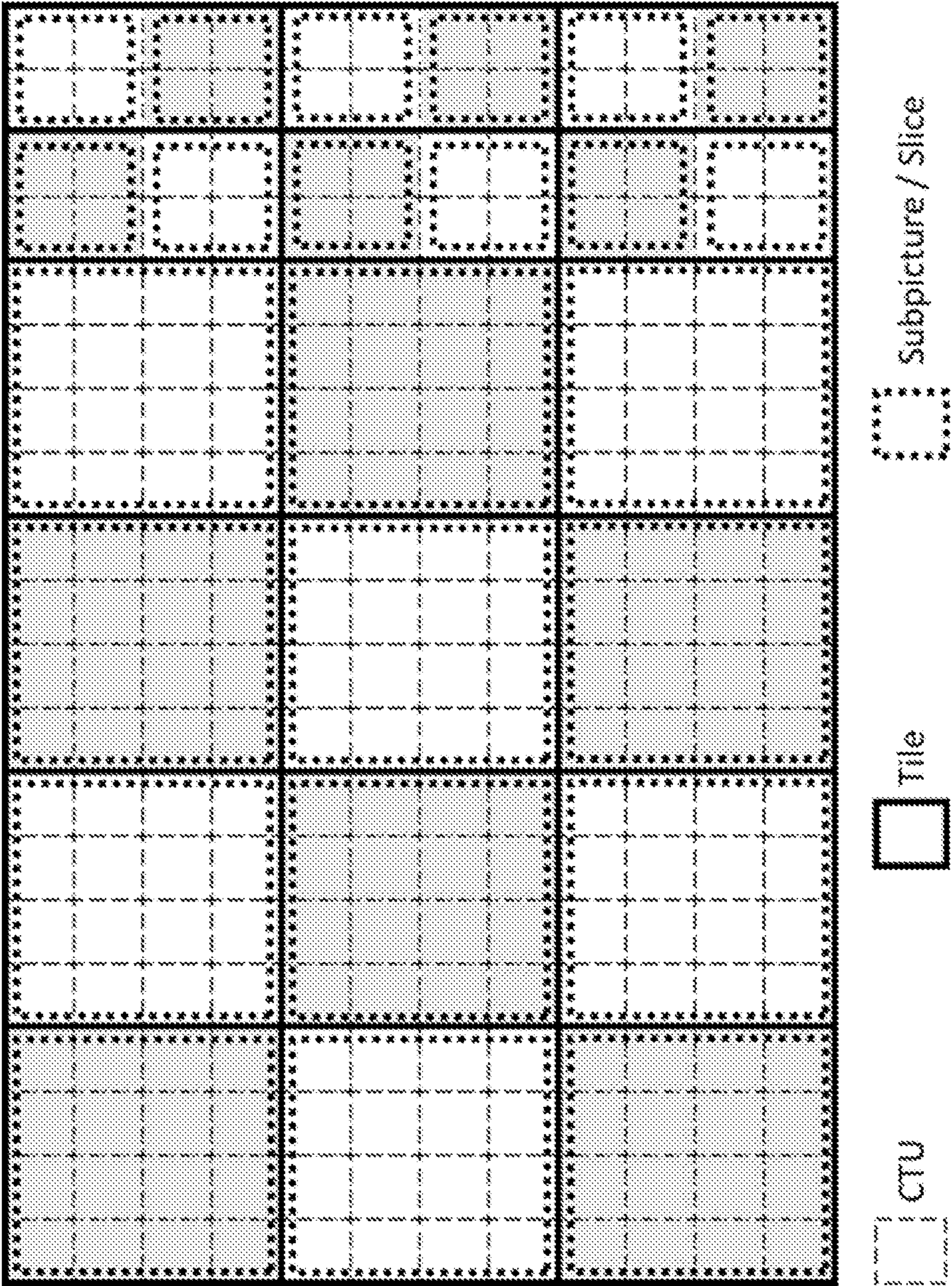


FIG. 4

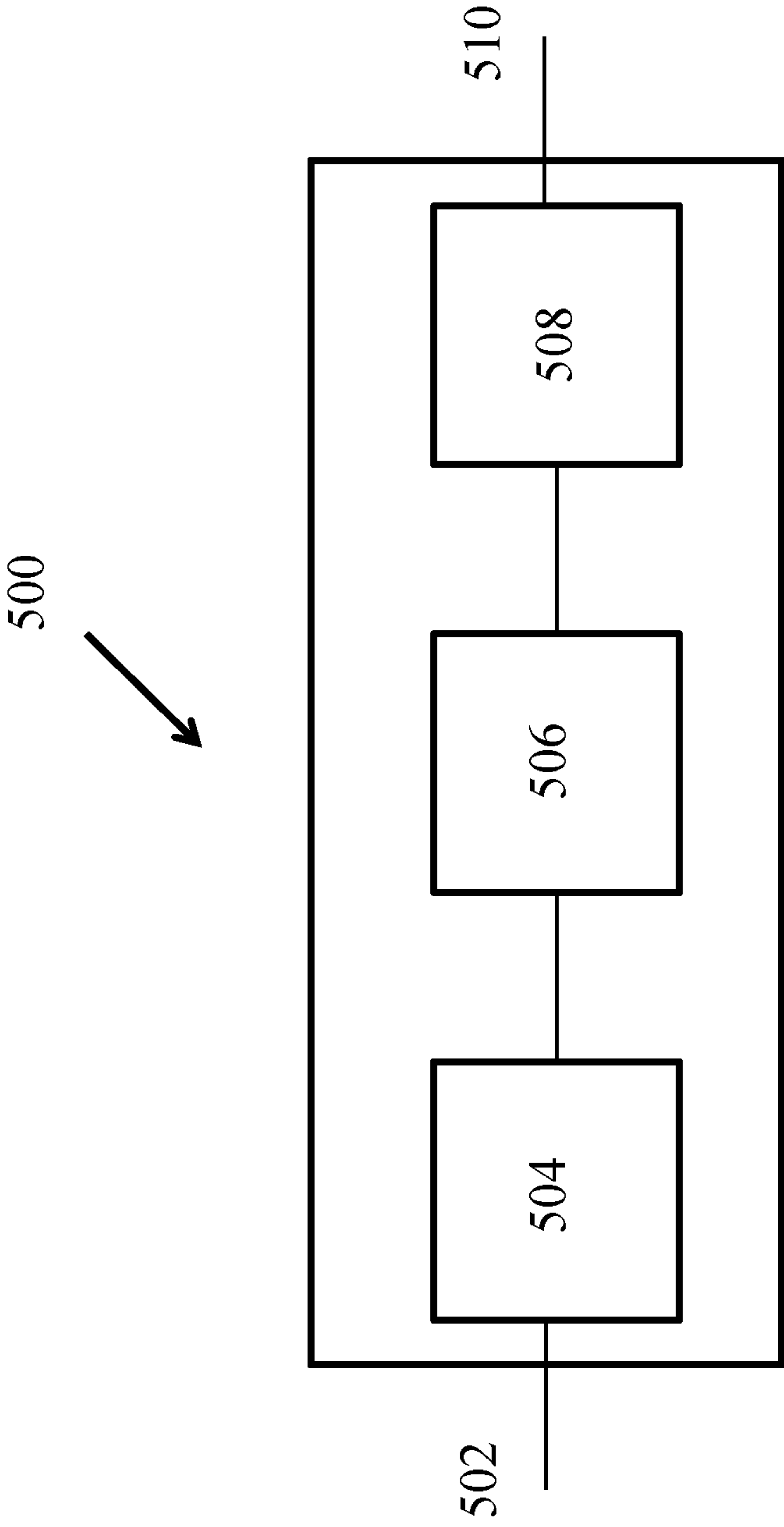


FIG. 5

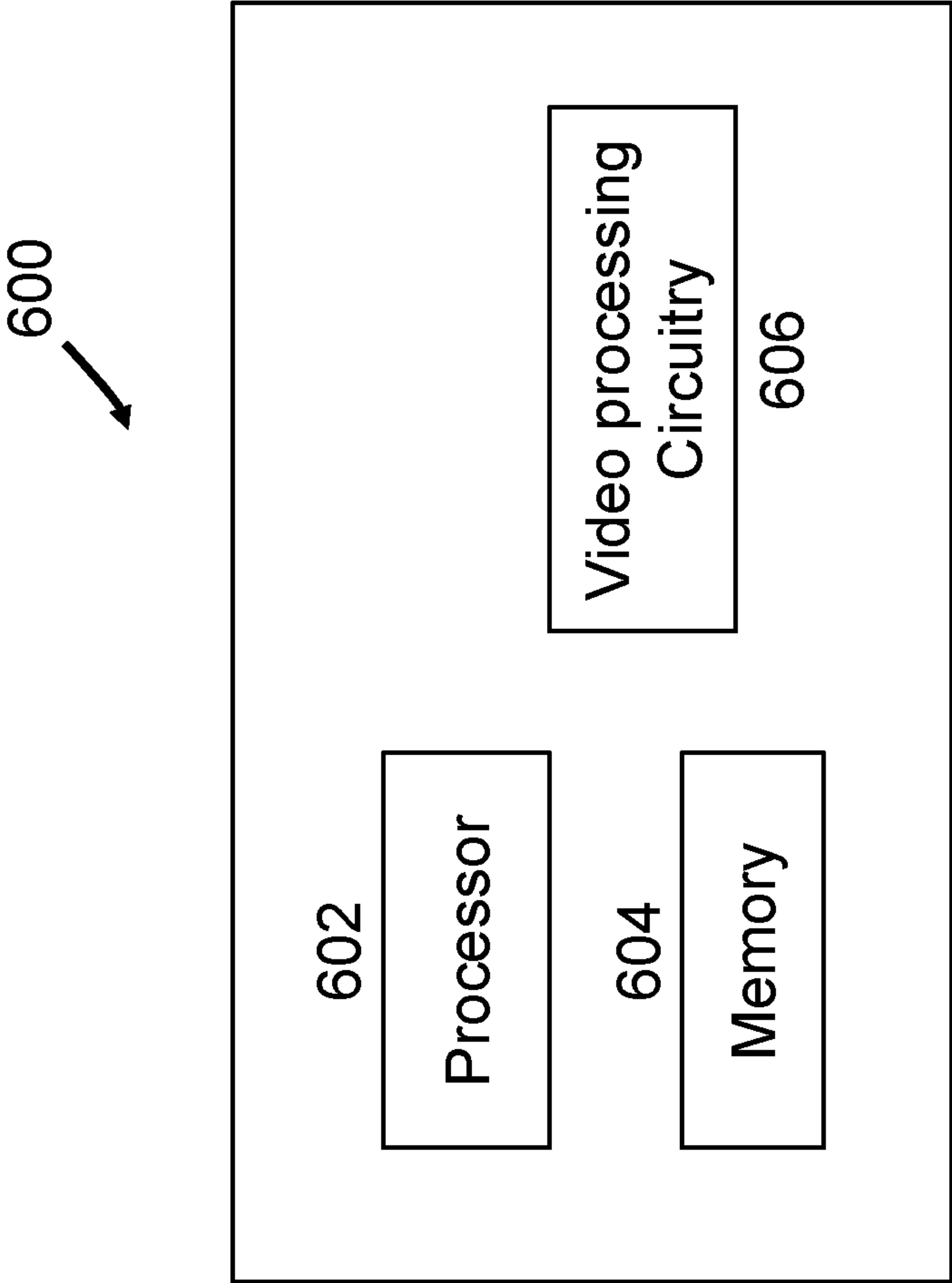


FIG. 6

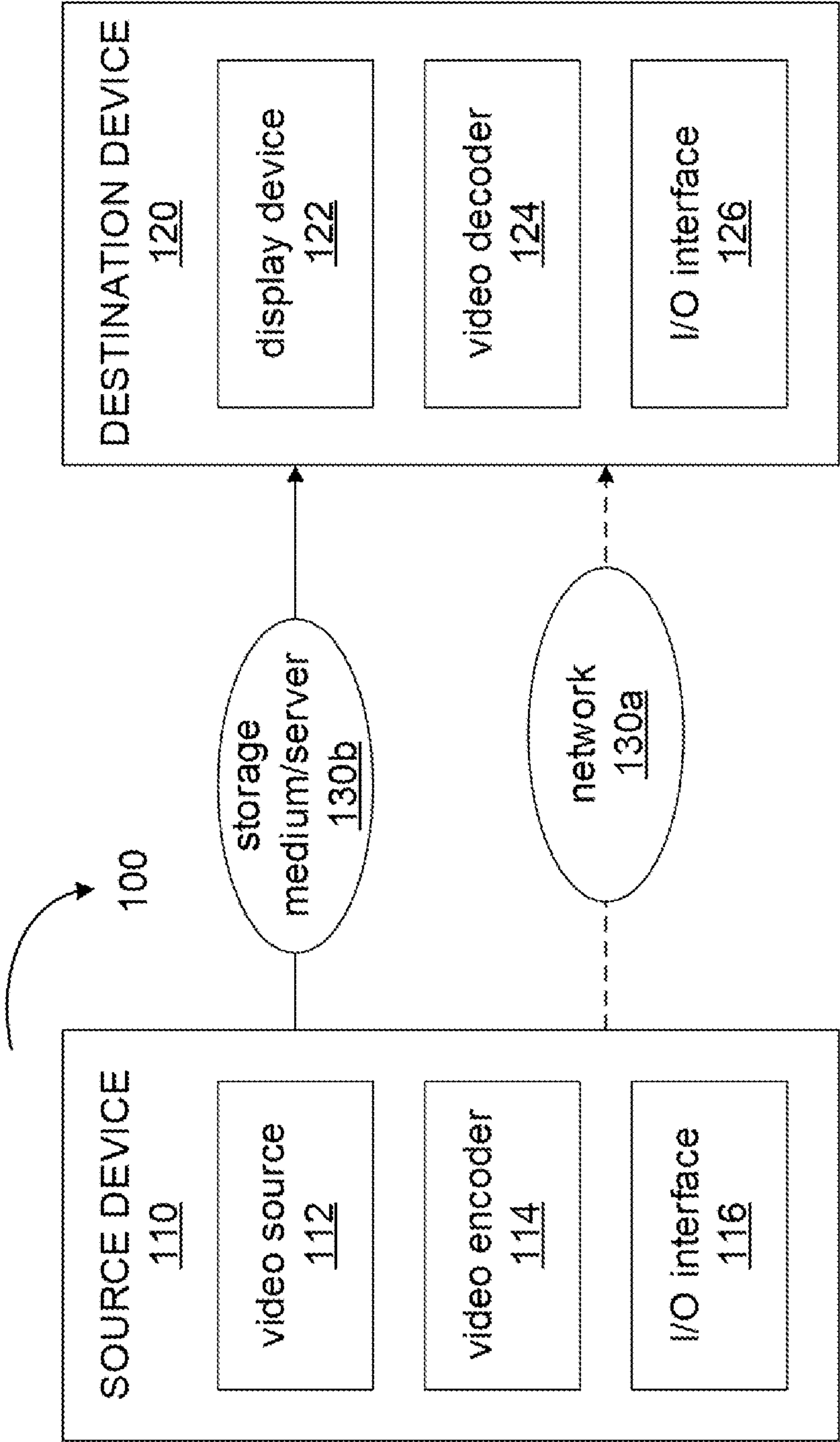


FIG. 7

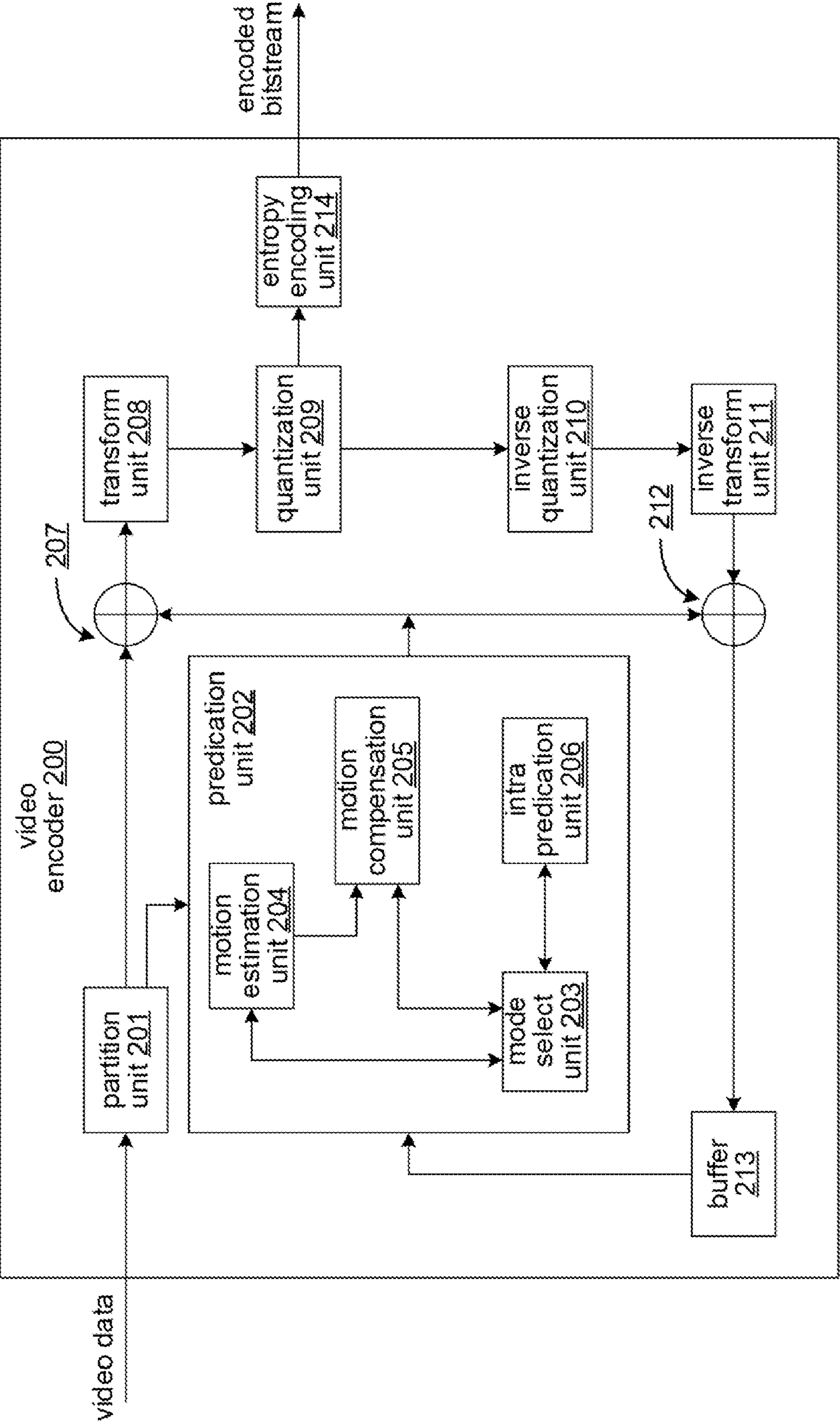


FIG. 8

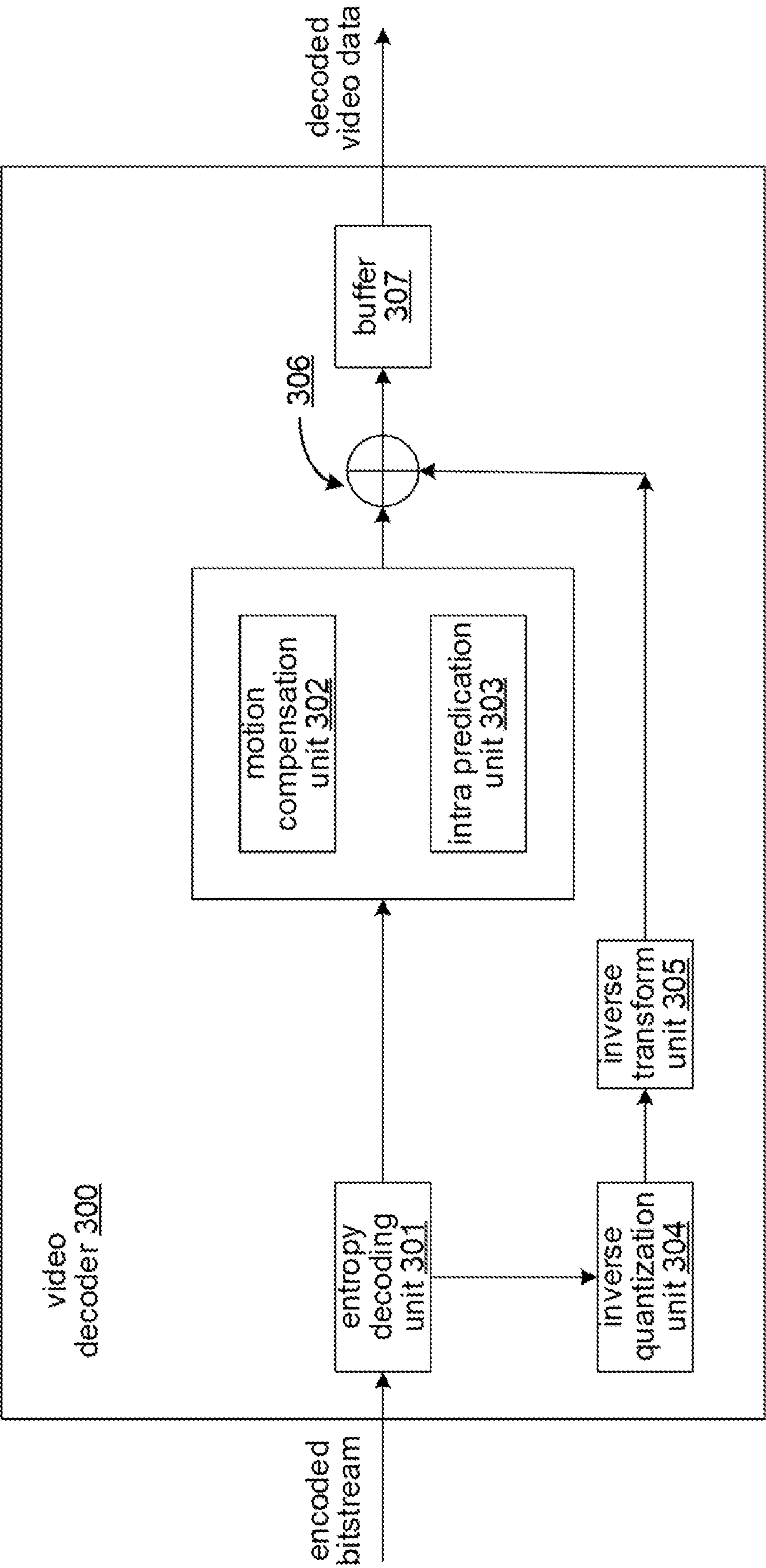


FIG. 9

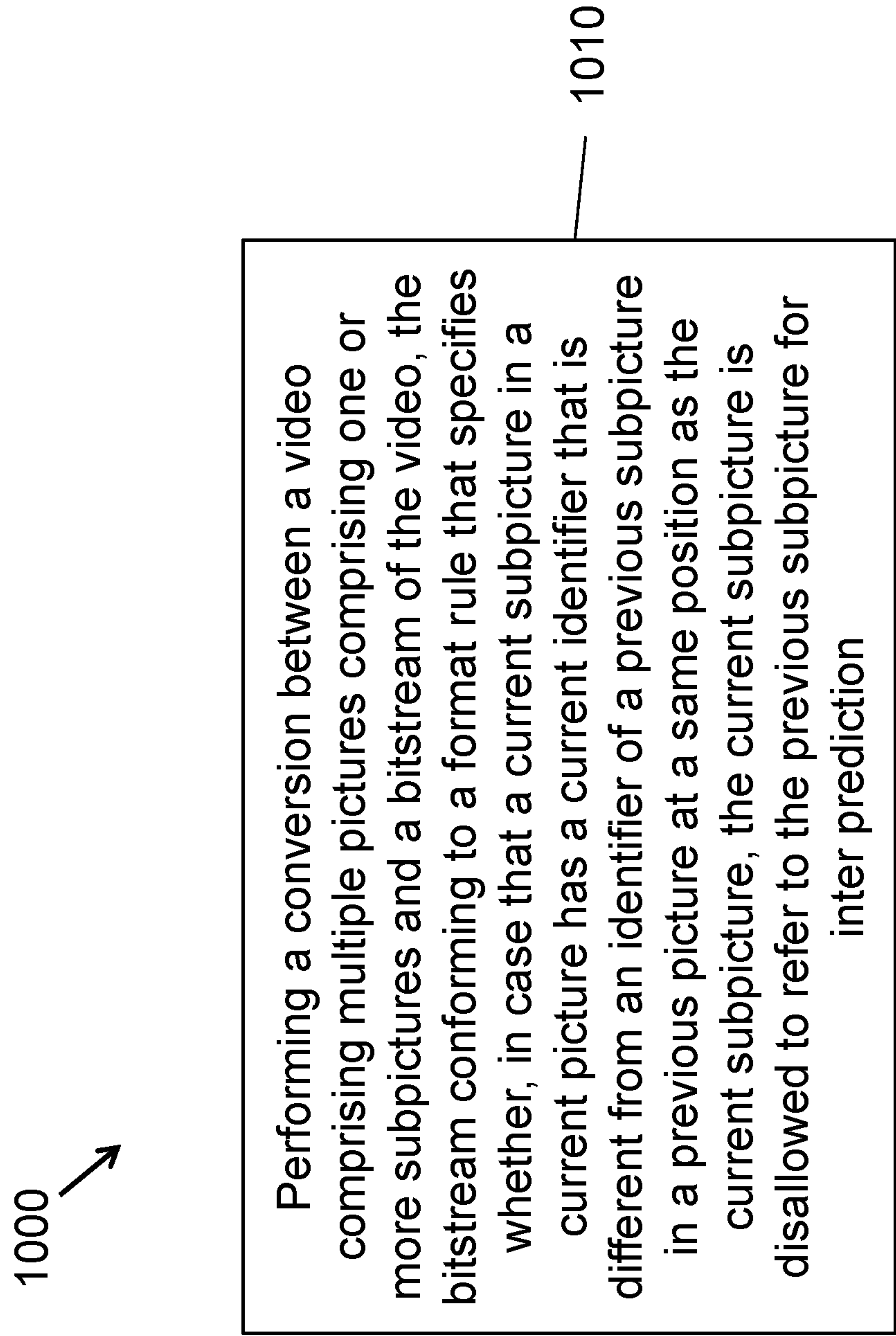


FIG. 10

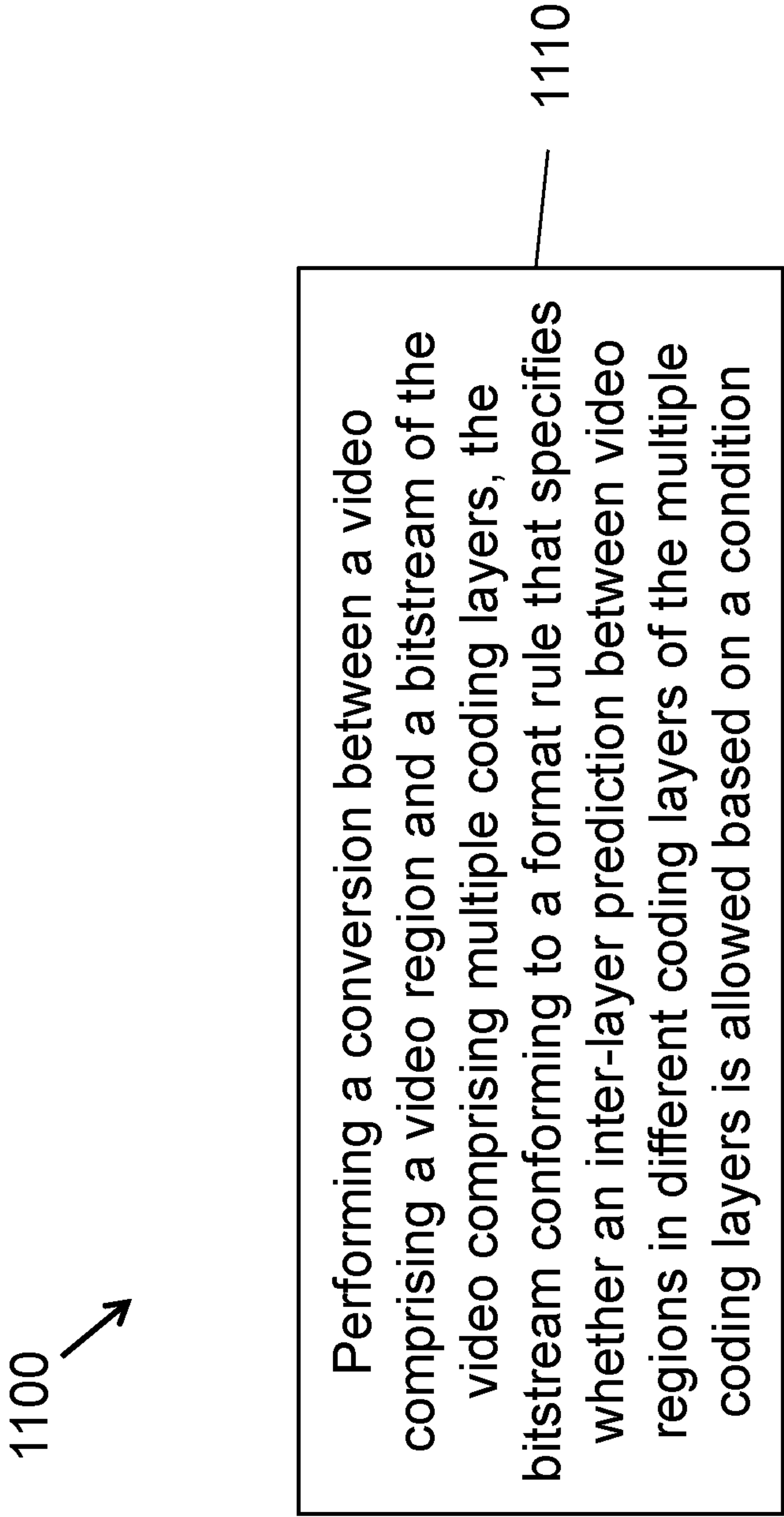


FIG. 11

1200 

Performing a conversion between a video comprising a current video region and a bitstream of the video comprising multiple coding layers, the bitstream conforming to a format rule that specifies the bitstream includes an indication of whether an inter-layer prediction between the current video region and a video region in a reference layer is allowed

1210 

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US 21/12035

Box No. II      Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:  
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:  
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☒ Claims Nos.: 30-36  
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III      Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 21/12035

## A. CLASSIFICATION OF SUBJECT MATTER

IPC - H04N 19/46 (2021.01)

CPC - H04N 19/103, H04N 19/169, H04N 19/187, H04N 19/70

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History document

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category*   | Citation of document, with indication, where appropriate, of the relevant passages                   | Relevant to claim No.                   |
|-------------|------------------------------------------------------------------------------------------------------|-----------------------------------------|
| X<br>—<br>Y | WO 2019/195035 A1 (FUTUREWEI TECHNOLOGIES, INC.) 10 October 2019 (10.10.2019); para [0017].          | 16-17, 20-26, 28<br>1-15, 18-19, 27, 29 |
| Y —         | CA 2884352 C (QUALCOMM INCORPORATED) 03 April 2014 (03.04.2014); paras [0075], [0159].               | 1-15, 18-19, 27, 29                     |
| Y           | US 2016/0165248 A1 (NOKIA TECHNOLOGIES, OY) 09 June 2016 (09.06.2016); paras [0185], [0196].         | 6, 8-10                                 |
| Y           | US 2017/0105014 A1 (QUALCOMM INCORPORATED) 13 April 2017 (13.04.2017); paras [0149], [0157], [0166]. | 12-14                                   |

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

## \* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“D” document cited by the applicant in the international application

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&amp;” document member of the same patent family

Date of the actual completion of the international search

24 February 2021 (24.02.2021)

Date of mailing of the international search report

MAR 24 2021

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents  
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Lee Young

Telephone No. PCT Helpdesk: 571-272-4300