(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 June 2021 (17.06.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/115362 A1

(51) International Patent Classification:
H04N 19/117 (2014.01) *H04N 19/82* (2014.01)

(21) International Application Number:
PCT/CN2020/135134

(22) International Filing Date:
10 December 2020 (10.12.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2019/124481
11 December 2019 (11.12.2019) CN

(71) Applicants: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No. 3 Building, No. 30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **WANG, Yang**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **DENG, Zhipin**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **WANG, Yue**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN).

(74) Agent: **LIU, SHEN & ASSOCIATES**; 10th Floor, Building 1, 10 Caihefang Road, Haidian District, Beijing 100080 (CN).

(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.

(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,

(54) Title: SAMPLE PADDING FOR CROSS-COMPONENT ADAPTIVE LOOP FILTERING

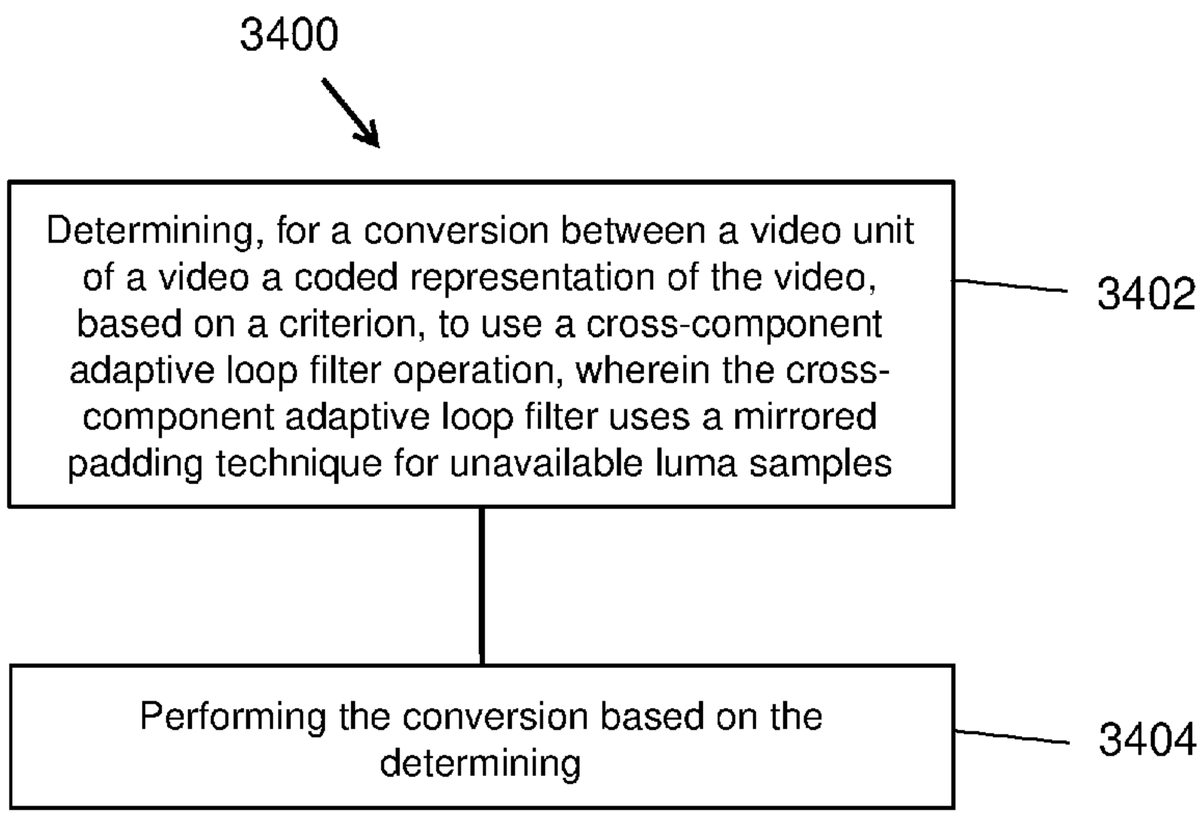


FIG. 34

(57) Abstract: A method for video processing is described. The method includes determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit; and performing the conversion based on the determining.

[Continued on next page]

WO 2021/115362 A1

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:

— *of inventorship (Rule 4.17(iv))*

Published:

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

SAMPLE PADDING FOR CROSS-COMPONENT ADAPTIVE LOOP FILTERING

CROSS REFERENCE TO RELATED APPLICATIONS

[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 International Patent Application No. PCT/CN2019/124481, filed on December 11, 2019. 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 to perform cross-component adaptive loop filtering during video encoding or decoding.

[005] In one example aspect, a method of video processing is disclosed. The method includes determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit; and performing the conversion based on the determining.

[006] In another example aspect, a method of video processing is disclosed. The method includes determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to apply a repetitive padding process and/or a mirrored padding process for padding a sample located at a virtual boundary based on coded information of the video unit; and performing the conversion based on the determining.

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

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

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

[0010] These, and other, features are described throughout the present document.

BRIEF DESCRIPTION OF DRAWINGS

[0011] FIG. 1 shows nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture.

[0012] FIG. 2 is an example of encoder block diagram.

[0013] FIG. 3 shows a picture with 18 by 12 luma CTUs that is partitioned into 12 tiles and 3 raster-scan slices.

[0014] FIG. 4 shows a picture with 18 by 12 luma CTUs that is partitioned into 24 tiles and 9 rectangular slices.

[0015] FIG. 5 shows a picture that is partitioned into 4 tiles and 4 rectangular slices.

[0016] FIG. 6 shows a picture that is partitioned into 15 tiles, 24 slices and 24 subpictures.

[0017] FIG. 7A-7C show: FIG. 7A - CTBs crossing the bottom picture border; FIG. 7B - CTBs crossing the right picture border; and FIG. 7C - CTBs crossing the right bottom picture border.

[0018] FIG. 8 is an illustration of picture samples and horizontal and vertical block boundaries on the 8×8 grid, and the nonoverlapping blocks of the 8×8 samples, which can be deblocked in parallel.

[0019] FIG. 9 shows pixels involved in filter on/off decision and strong/weak filter selection.

[0020] FIG. 10 shows four 1-D directional patterns for EO sample classification: horizontal (EO class = 0), vertical (EO class = 1), 135° diagonal (EO class = 2), and 45° diagonal (EO class = 3)

[0021] FIG. 11 shows examples of ALF filter shapes (chroma: 5×5 diamond, luma: 7×7 diamond).

[0022] FIG. 12A shows subsampled positions for vertical gradient, FIG. 12B shows subsampled positions for horizontal gradient, FIG. 12C shows subsampled positions for diagonal gradient, and FIG. 12D shows subsampled positions for diagonal gradient.

[0023] FIG. 13 shows an example of Loop filter line buffer requirement in VTM-4.0 for Luma component.

[0024] FIG. 14 illustrates an example of loop filter line buffer requirement in VTM-4.0 for Chroma component.

[0025] FIG. 15 shows an example of modified block classification at virtual boundaries.

[0026] FIG. 16 shows an example of modified ALF filtering for Luma component at virtual boundaries.

[0027] FIG. 17A shows one required line above/below VB need to be padded (per side), FIG. 17B shows 2 required lines above/below VB need to be padded (per side), and FIG. 17C shows 3 required lines above/below VB need to be padded (per side).

[0028] FIG. 18 shows examples of repetitive padding for luma ALF filtering at picture/subpicture/slice/tile boundary.

[0029] FIG. 19 shows an example of a Horizontal wrap around motion compensation in VVC.

[0030] FIG. 20 shows an image of HEC in 3x2 layout.

[0031] FIG. 21A shows a placement of CC-ALF with respect to other loop filters. FIG. 21B shows a diamond shaped filter.

[0032] FIG. 22 shows an example of repetitive padding at ALF virtual boundary for CC-ALF in JVET-P0080.

[0033] FIG. 23 shows a 3x4 Diamond Shape Filter with 8 unique coefficients.

[0034] FIG. 24 is an example of repetitive padding at ALF virtual boundary for CC-ALF in JVET-P1008.

[0035] FIG. 25 shows CC-ALF filter shape of 8 coefficients in JVET-P0106.

[0036] FIG. 26 shows CC-ALF filter shape of 6 coefficients in JVET-P0173.

[0037] FIG. 27 shows CC-ALF filter shape of 6 coefficients in JVET-P0251.

[0038] FIG. 28 shows an example of a JC-CCALF workflow.

[0039] FIG. 29 shows example locations of samples to be padded for the CC-ALF filtering method with 8-tap 4x3 filter shape.

[0040] FIG. 30 shows an example of mirrored padding method 1.

[0041] FIG. 31 shows an example of mirrored padding method 2.

[0042] FIG. 32 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[0043] FIG. 33 is a block diagram of an example hardware platform used for video processing.

[0044] FIG. 34 is a flowchart for an example method of video processing.

[0045] FIG. 35 is a block diagram that illustrates a video coding system in accordance with some embodiments of the present disclosure.

[0046] FIG. 36 is a block diagram that illustrates an encoder in accordance with some embodiments of the present disclosure.

[0047] FIG. 37 is a block diagram that illustrates a decoder in accordance with some embodiments of the present disclosure.

[0048] FIGS. 38A and 38B are flowcharts for example methods of video processing based on some implementations of the disclosed technology.

DETAILED DESCRIPTION

[0049] 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. Brief Summary

[0050] This document is related to video coding technologies. Specifically, it is related picture/subpicture/slice/tile boundary, and 360-degree video virtual boundary and ALF virtual boundary coding especially for cross component adaptive loop filter (CC-ALF) and other coding tools in image/video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

2. Video Coding Introduction

[0051] 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, 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). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

2.1. Color space and chroma subsampling

[0052] Color space, also known as the color model (or color system), is an abstract mathematical model which simply describes the range of colors as tuples of numbers, typically as 3 or 4 values

or color components (e.g. RGB). Basically speaking, color space is an elaboration of the coordinate system and sub-space.

[0053] For video compression, the most frequently used color spaces are YCbCr and RGB.

[0054] YCbCr, Y′CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y'CBCR, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y′ is the luma component and CB and CR are the blue-difference and red-difference chroma components. Y′ (with prime) is distinguished from Y, which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.

[0055] Chroma subsampling is the practice of encoding images by implementing less resolution for chroma information than for luma information, taking advantage of the human visual system's lower acuity for color differences than for luminance.

2.1.1. 4:4:4

[0056] Each of the three Y'CbCr components have the same sample rate, thus there is no chroma subsampling. This scheme is sometimes used in high-end film scanners and cinematic post production.

2.1.2. 4:2:2

[0057] The two chroma components are sampled at half the sample rate of luma: the horizontal chroma resolution is halved while the vertical chroma resolution is unchanged. This reduces the bandwidth of an uncompressed video signal by one-third with little to no visual difference. An example of nominal vertical and horizontal locations of 4:2:2 color format is depicted in FIG. 1 in VVC working draft.

2.1.3. 4:2:0

[0058] In 4:2:0, the horizontal sampling is doubled compared to 4:1:1, but as the Cb and Cr channels are only sampled on each alternate line in this scheme, the vertical resolution is halved. The data rate is thus the same. Cb and Cr are each subsampled at a factor of 2 both horizontally and vertically. There are three variants of 4:2:0 schemes, having different horizontal and vertical siting.

- In MPEG-2, Cb and Cr are cosited horizontally. Cb and Cr are sited between pixels in the vertical direction (sited interstitially).
- In JPEG/JFIF, H.261, and MPEG-1, Cb and Cr are sited interstitially, halfway between alternate luma samples.
- In 4:2:0 DV, Cb and Cr are co-sited in the horizontal direction. In the vertical direction, they are co-sited on alternating lines.

Table 2-1. SubWidthC and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

•

chroma_format_idc	separate_colour_plane_flag	Chroma format	SubWidthC	SubHeightC
0	0	Monochrome	1	1
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	1	1

2.2. Coding flow of a typical video codec

[0059] FIG. 2 shows an example of encoder block diagram of VVC, which contains three in-loop filtering blocks: deblocking filter (DF), sample adaptive offset (SAO) and ALF. Unlike DF, which uses predefined filters, SAO and ALF utilize the original samples of the current picture to reduce the mean square errors between the original samples and the reconstructed samples by adding an offset and by applying a finite impulse response (FIR) filter, respectively, with coded side information signalling the offsets and filter coefficients. ALF is located at the last processing stage of each picture and can be regarded as a tool trying to catch and fix artifacts created by the previous stages.

2.3. Example definitions of video units

[0060] 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.

[0061] 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.

[0062] 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.

[0063] A subpicture contains one or more slices that collectively cover a rectangular region of a picture.

[0064] FIG. 3 shows an example of raster-scan slice partitioning of a picture, where the picture is divided into 12 tiles and 3 raster-scan slices.

[0065] FIG. 4 in the VVC specification 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.

[0066] FIG. 4 A picture with 18 by 12 luma CTUs that is partitioned into 24 tiles and 9 rectangular slices (informative)

[0067] FIG. 5 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.

[0068] FIG. 6 shows an example of subpicture partitioning of a picture, where a picture is partitioned into 15 tiles covering 4 by 4 CTUs, 24 slices and 24 subpictures of varying dimensions.

2.3.1. CTU/CTB sizes

[0069] In VVC, the CTU size, signaled in SPS by the syntax element **log2_ctu_size_minus2**, could be as small as 4x4.

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_5bits	u(5)
profile_tier_level(sps_max_sub_layers_minus1)	
gra_enabled_flag	u(1)
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)

if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
long_term_ref_pics_flag	u(1)
sps_idr_rpl_present_flag	u(1)
rpl1_same_as_rpl0_flag	u(1)
for(i = 0; i < !rpl1_same_as_rpl0_flag ? 2 : 1; i++) {	
num_ref_pic_lists_in_sps[i]	ue(v)
for(j = 0; j < num_ref_pic_lists_in_sps[i]; j++)	
ref_pic_list_struct(i, j)	
}	
qtbtt_dual_tree_intra_flag	u(1)
log2_ctu_size_minus2	ue(v)
log2_min_luma_coding_block_size_minus2	ue(v)
partition_constraints_override_enabled_flag	u(1)
sps_log2_diff_min_qt_min_cb_intra_slice_luma	ue(v)
sps_log2_diff_min_qt_min_cb_inter_slice	ue(v)

sps_max_mtt_hierarchy_depth_inter_slice	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_luma	ue(v)
if(sps_max_mtt_hierarchy_depth_intra_slice_luma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_luma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_luma	ue(v)
}	
if(sps_max_mtt_hierarchy_depth_inter_slices != 0) {	
sps_log2_diff_max_bt_min_qt_inter_slice	ue(v)
sps_log2_diff_max_tt_min_qt_inter_slice	ue(v)
}	
if(qtbtt_dual_tree_intra_flag) {	
sps_log2_diff_min_qt_min_cb_intra_slice_chroma	ue(v)
sps_max_mtt_hierarchy_depth_intra_slice_chroma	ue(v)
if (sps_max_mtt_hierarchy_depth_intra_slice_chroma != 0) {	
sps_log2_diff_max_bt_min_qt_intra_slice_chroma	ue(v)
sps_log2_diff_max_tt_min_qt_intra_slice_chroma	ue(v)
}	
}	
…	
rbsp_trailing_bits( )	
}	

log2_ctu_size_minus2 plus 2 specifies the luma coding tree block size of each CTU.

log2_min_luma_coding_block_size_minus2 plus 2 specifies the minimum luma coding block size.

The variables CtbLog2SizeY, CtbSizeY, MinCbLog2SizeY, MinCbSizeY, MinTbLog2SizeY, MaxTbLog2SizeY, MinTbSizeY, MaxTbSizeY, PicWidthInCtbsY, PicHeightInCtbsY, PicSizeInCtbsY, PicWidthInMinCbsY, PicHeightInMinCbsY, PicSizeInMinCbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC are derived as follows:

$$CtbLog2SizeY = log2_ctu_size_minus2 + 2 \quad (7\text{-}9)$$

$$CtbSizeY = 1 \ll CtbLog2SizeY \quad (7\text{-}10)$$

$$MinCbLog2SizeY = log2_min_luma_coding_block_size_minus2 + 2 \quad (7\text{-}11)$$

$$MinCbSizeY = 1 \ll MinCbLog2SizeY \quad (7\text{-}12)$$

$$MinTbLog2SizeY = 2 \quad (7\text{-}13)$$

MaxTbLog2SizeY = 6 (7-14)

MinTbSizeY = 1 << MinTbLog2SizeY (7-15)

MaxTbSizeY = 1 << MaxTbLog2SizeY (7-16)

PicWidthInCtbsY = Ceil(pic_width_in_luma_samples ÷ CtbSizeY) (7-17)

PicHeightInCtbsY = Ceil(pic_height_in_luma_samples ÷ CtbSizeY) (7-18)

PicSizeInCtbsY = PicWidthInCtbsY * PicHeightInCtbsY (7-19)

PicWidthInMinCbsY = pic_width_in_luma_samples / MinCbSizeY (7-20)

PicHeightInMinCbsY = pic_height_in_luma_samples / MinCbSizeY (7-21)

PicSizeInMinCbsY = PicWidthInMinCbsY * PicHeightInMinCbsY (7-22)

PicSizeInSamplesY = pic_width_in_luma_samples * pic_height_in_luma_samples (7-23)

PicWidthInSamplesC = pic_width_in_luma_samples / SubWidthC (7-24)

PicHeightInSamplesC = pic_height_in_luma_samples / SubHeightC (7-25)

2.3.2. CTUs in a picture

[0070] Suppose the CTB/LCU size indicated by M×N (typically M is equal to N, as defined in HEVC/VVC), and for a CTB located at picture (or tile or slice or other kinds of types, picture border is taken as an example) border, K×L samples are within picture border wherein either K<M or L<N. For those CTBs as depicted in FIG. 7A-7C, **the CTB size is still equal to MxN, however, the bottom boundary/right boundary of the CTB is outside the picture.**

[0071] FIG. 7 shows examples of CTBs crossing picture borders, (a) K=M, L<N; (b) K<M, L=N; (c) K<M, L<N. Also, (a) CTBs crossing the bottom picture border (b) CTBs crossing the right picture border, (c) CTBs crossing the right bottom picture border

2.4. Deblocking filter (DB)

[0072] The input of DB is the reconstructed samples before in-loop filters.

[0073] The vertical edges in a picture are filtered first. Then the horizontal edges in a picture are filtered with samples modified by the vertical edge filtering process as input. The vertical and horizontal edges in the CTBs of each CTU are processed separately on a coding unit basis. The vertical edges of the coding blocks in a coding unit are filtered starting with the edge on the left-hand side of the coding blocks proceeding through the edges towards the right-hand side of the coding blocks in their geometrical order. The horizontal edges of the coding blocks in a coding unit are filtered starting with the edge on the top of the coding blocks proceeding through the edges towards the bottom of the coding blocks in their geometrical order.

[0074] FIG. 8 is an Illustration of picture samples and horizontal and vertical block boundaries on the 8×8 grid, and the nonoverlapping blocks of the 8×8 samples, which can be deblocked in parallel.

2.4.1. Boundary decision

[0075] Filtering is applied to 8×8 block boundaries. In addition, it must be a transform block boundary or a coding subblock boundary (e.g., due to usage of Affine motion prediction, ATMVP). For those which are not such boundaries, filter is disabled.

2.4.2. Boundary strength calculation

[0076] For a transform block boundary/coding subblock boundary, if it is located in the 8×8 grid, it may be filterd and the setting of bS[xDi][yDj] (wherein [xDi][yDj] denotes the coordinate) for this edge is defined in Table 2-2 and Table 2-3, respectively.

Table 2-2. Boundary strength (when SPS IBC is disabled)

Priority	Conditions	Y	U	V
5	At least one of the adjacent blocks is intra	2	2	2
4	TU boundary and at least one of the adjacent blocks has non-zero transform coefficients	1	1	1
3	Reference pictures or number of MVs (1 for uni-prediction, 2 for bi-prediction) of the adjacent blocks are different	1	N/A	N/A
2	Absolute difference between the motion vectors of same reference picture that belong to the adjacent blocks is greater than or equal to one integer luma sample	1	N/A	N/A
1	Otherwise	0	0	0

Table 2-3. Boundary strength (when SPS IBC is enabled)

Priority	Conditions	Y	U	V
8	At least one of the adjacent blocks is intra	2	2	2
7	TU boundary and at least one of the adjacent blocks has non-zero transform coefficients	1	1	1
6	Prediction mode of adjacent blocks is different (e.g., one is IBC, one is inter)	1		

5	Both IBC and absolute difference between the motion vectors that belong to the adjacent blocks is greater than or equal to one integer luma sample	1	N/A	N/A
4	Reference pictures or number of MVs (1 for uni-prediction, 2 for bi-prediction) of the adjacent blocks are different	1	N/A	N/A
3	Absolute difference between the motion vectors of same reference picture that belong to the adjacent blocks is greater than or equal to one integer luma sample	1	N/A	N/A
1	Otherwise	0	0	0

2.4.3. Deblocking decision for luma component

[0077] The deblocking decision process is described in this sub-section.

[0078] Wider-stronger luma filter is filters are used only if all the Condition1, Condition2 and Condition 3 are TRUE.

The condition 1 is the “large block condition”. This condition detects whether the samples at P-side and Q-side belong to large blocks, which are represented by the variable bSidePisLargeBlk and bSideQisLargeBlk respectively. The bSidePisLargeBlk and bSideQisLargeBlk are defined as follows.

bSidePisLargeBlk = ((edge type is vertical and p_0 belongs to CU with width >= 32) | | (edge type is horizontal and p_0 belongs to CU with height >= 32))? TRUE: FALSE

bSideQisLargeBlk = ((edge type is vertical and q_0 belongs to CU with width >= 32) | | (edge type is horizontal and q_0 belongs to CU with height >= 32))? TRUE: FALSE

Based on bSidePisLargeBlk and bSideQisLargeBlk, the condition 1 is defined as follows.

Condition1 = (bSidePisLargeBlk || bSidePisLargeBlk) ? TRUE: FALSE

Next, if **Condition 1** is true, the condition 2 will be further checked. First, the following variables are derived:

- dp0, dp3, dq0, dq3 are first derived as in HEVC
- if (p side is greater than or equal to 32)

  $dp0 = (dp0 + Abs(p5_0 - 2 * p4_0 + p3_0) + 1) >> 1$

  $dp3 = (dp3 + Abs(p5_3 - 2 * p4_3 + p3_3) + 1) >> 1$
- if (q side is greater than or equal to 32)

  $dq0 = (dq0 + Abs(q5_0 - 2 * q4_0 + q3_0) + 1) >> 1$

  $dq3 = (dq3 + Abs(q5_3 - 2 * q4_3 + q3_3) + 1) >> 1$

Condition2 = (d < β) ? TRUE: FALSE

where d= dp0 + dq0 + dp3 + dq3.

If Condition1 and Condition2 are valid, whether any of the blocks uses sub-blocks is further checked:

```
If (bSidePisLargeBlk)
{
   If (mode block P == SUBBLOCKMODE)
      Sp =5
   else
      Sp =7
}
else
 Sp = 3

If (bSideQisLargeBlk)
 {
   If (mode block Q == SUBBLOCKMODE)
      Sq =5
   else
      Sq =7
  }
else
  Sq = 3
```

Finally, if both the **Condition 1** and **Condition 2** are valid, the proposed deblocking method will check the condition 3 (the large block strong filter condition), which is defined as follows.

In the Condition3 StrongFilterCondition, the following variables are derived:

dpq is derived as in HEVC.

sp_3 = Abs(p_3 − p_0), derived as in HEVC

if (p side is greater than or equal to 32)

if(Sp==5)

sp_3 = (sp_3 + Abs(p_5 − p_3) + 1) >> 1

else

sp_3 = (sp_3 + Abs(p_7 − p_3) + 1) >> 1

sq_3 = Abs(q_0 − q_3), derived as in HEVC

if (q side is greater than or equal to 32)

If(Sq==5)

sq_3 = (sq_3 + Abs(q_5 − q_3) + 1) >> 1

else

sq_3 = (sq_3 + Abs(q_7 − q_3) + 1) >> 1

As in HEVC, StrongFilterCondition = (dpq is less than (β >> 2), sp_3 + sq_3 is less than (3*β >> 5), and Abs(p_0 − q_0) is less than (5 * t_C + 1) >> 1) ? TRUE : FALSE.

2.4.4. Stronger deblocking filter for luma (designed for larger blocks)

[0079] Bilinear filter is used when samples at either one side of a boundary belong to a large block. A sample belonging to a large block is defined as when the width >= 32 for a vertical edge, and when height >= 32 for a horizontal edge.

[0080] The bilinear filter is listed below.

[0081] Block boundary samples p_i for i=0 to Sp-1 and q_i for j=0 to Sq-1 (pi and qi are the i-th sample within a row for filtering vertical edge, or the i-th sample within a column for filtering horizontal edge) in HEVC deblocking described above) are then replaced by linear interpolation as follows:

— $p_i' = (f_i * Middle_{s,t} + (64 - f_i) * P_s + 32) \gg 6), clipped\ to\ p_i \pm tcPD_i$

— $q_j' = (g_j * Middle_{s,t} + (64 - g_j) * Q_s + 32) \gg 6), clipped\ to\ q_j \pm tcPD_j$

where $tcPD_i$ and $tcPD_j$ term is a position dependent clipping described in Section 2.4.7 and g_j, f_i, $Middle_{s,t}$, P_s and Q_s are given below:

2.4.5. Deblocking control for chroma

[0082] The chroma strong filters are used on both sides of the block boundary. Here, the chroma filter is selected when both sides of the chroma edge are greater than or equal to 8 (chroma position), and the following decision with three conditions are satisfied: the first one is for decision of boundary strength as well as large block. The proposed filter can be applied when the block width or height which orthogonally crosses the block edge is equal to or larger than 8 in chroma sample domain. The second and third one is basically the same as for HEVC luma deblocking decision, which are on/off decision and strong filter decision, respectively.

[0083] In the first decision, boundary strength (bS) is modified for chroma filtering and the conditions are checked sequentially. If a condition is satisfied, then the remaining conditions with lower priorities are skipped.

[0084] Chroma deblocking is performed when bS is equal to 2, or bS is equal to 1 when a large block boundary is detected.

[0085] The second and third condition is basically the same as HEVC luma strong filter decision as follows.

In the second condition:

d is then derived as in HEVC luma deblocking.

The second condition will be TRUE when d is less than β.

In the third condition StrongFilterCondition is derived as follows:

dpq is derived as in HEVC.

sp_3 = Abs($p_3 - p_0$), derived as in HEVC

sq_3 = Abs($q_0 - q_3$), derived as in HEVC

[0086] As in HEVC design, StrongFilterCondition = (dpq is less than (β >> 2), $sp_3 + sq_3$ is less than (β >> 3), and Abs($p_0 - q_0$) is less than (5 * t_C + 1) >> 1)

2.4.6. Strong deblocking filter for chroma

[0087] The following strong deblocking filter for chroma is defined:

$p_2' = (3*p_3+2*p_2+p_1+p_0+q_0+4) >> 3$

$p_1' = (2*p_3+p_2+2*p_1+p_0+q_0+q_1+4) >> 3$

$p_0' = (p_3+p_2+p_1+2*p_0+q_0+q_1+q_2+4) >> 3$

[0088] The proposed chroma filter performs deblocking on a 4x4 chroma sample grid.

2.4.7. Position dependent clipping

[0089] The position dependent clipping tcPD is applied to the output samples of the luma filtering process involving strong and long filters that are modifying 7, 5 and 3 samples at the boundary. Assuming quantization error distribution, it is proposed to increase clipping value for samples which are expected to have higher quantization noise, thus expected to have higher deviation of the reconstructed sample value from the true sample value.

[0090] For each P or Q boundary filtered with asymmetrical filter, depending on the result of decision-making process in section 2.4.2, position dependent threshold table is selected from two tables (i.e., Tc7 and Tc3 tabulated below) that are provided to decoder as a side information:

Tc7 = { 6, 5, 4, 3, 2, 1, 1}; Tc3 = { 6, 4, 2 };

tcPD = (Sp == 3) ? Tc3 : Tc7;

tcQD = (Sq == 3) ? Tc3 : Tc7;

[0091] For the P or Q boundaries being filtered with a short symmetrical filter, position dependent threshold of lower magnitude is applied:

Tc3 = { 3, 2, 1 };

[0092] Following defining the threshold, filtered p'_i and q'_i sample values are clipped according to tcP and tcQ clipping values:

$p''_i = Clip3(p'_i + tcP_i, p'_i - tcP_i, p'_i);$

$q''_j = Clip3(q'_j + tcQ_j, q'_j - tcQ_j, q'_j);$

[0093] where p'_i and q'_i are filtered sample values, p''_i and q''_j are output sample value after the clipping and tcP_i tcP_i are clipping thresholds that are derived from the VVC tc parameter and *tcPD* and *tcQD*. The function Clip3 is a clipping function as it is specified in VVC.

2.4.8. Sub-block deblocking adjustment

[0094] To enable parallel friendly deblocking using both long filters and sub-block deblocking the long filters is restricted to modify at most 5 samples on a side that uses sub-block deblocking (AFFINE or ATMVP or DMVR) as shown in the luma control for long filters. Additionally, the sub-block deblocking is adjusted such that that sub-block boundaries on an 8x8 grid that are close to a CU or an implicit TU boundary is restricted to modify at most two samples on each side.

Following applies to sub-block boundaries that not are aligned with the CU boundary.

```
If (mode block Q == SUBBLOCKMODE && edge !=0) {
  if (!(implicitTU && (edge == (64 / 4))))
    if (edge == 2 || edge == (orthogonalLength - 2) || edge == (56 / 4) || edge == (72 / 4))
      Sp = Sq = 2;
    else
      Sp = Sq = 3;
  else
    Sp = Sq = bSideQisLargeBlk ? 5:3
}
```

[0095] Where edge equal to 0 corresponds to CU boundary, edge equal to 2 or equal to orthogonalLength-2 corresponds to sub-block boundary 8 samples from a CU boundary etc. Where implicit TU is true if implicit split of TU is used.

2.5. Sample adaptive offset (SAO)

[0096] The input of SAO is the reconstructed samples after DB. The concept of SAO is to reduce mean sample distortion of a region by first classifying the region samples into multiple categories with a selected classifier, obtaining an offset for each category, and then adding the offset to each sample of the category, where the classifier index and the offsets of the region are

coded in the bitstream. In HEVC and VVC, the region (the unit for SAO parameters signaling) is defined to be a CTU.

[0097] Two SAO types that can satisfy the requirements of low complexity are adopted in HEVC. Those two types are edge offset (EO) and band offset (BO), which are discussed in further detail below. An index of an SAO type is coded (which is in the range of [0, 2]). For EO, the sample classification is based on comparison between current samples and neighboring samples according to 1-D directional patterns: horizontal, vertical, 135° diagonal, and 45° diagonal.

[0098] FIG. 10A-10D show four 1-D directional patterns for EO sample classification: horizontal (EO class = 0), vertical (EO class = 1), 135° diagonal (EO class = 2), and 45° diagonal (EO class = 3)

[0099] For a given EO class, each sample inside the CTB is classified into one of five categories. The current sample value, labeled as “c,” is compared with its two neighbors (labeled as “a” and “b”) along the selected 1-D pattern. The classification rules for each sample are summarized in Table 2-4. Categories 1 and 4 are associated with a local valley and a local peak along the selected 1-D pattern, respectively. Categories 2 and 3 are associated with concave and convex corners along the selected 1-D pattern, respectively. If the current sample does not belong to EO categories 1–4, then it is category 0 and SAO is not applied.

Table 2-4. Sample Classification Rules for Edge Offset

Category	Condition
1	c<a and c<b
2	(c < a && c==b) ||(c == a && c < b)
3	(c > a && c==b) ||(c == a && c > b)
4	c > a && c > b
5	None of above

2.6. Adaptive Loop Filter (ALF)

[00100] In VVC, an Adaptive Loop Filter (ALF) with block-based filter adaption is applied. For the luma component, one among 25 filters is selected for each 4×4 block, based on the direction and activity of local gradients.

2.6.1. Filter shape

[00101] Two diamond filter shapes (as shown in FIG. 11) are used. The 7×7 diamond shape is applied for luma component and the 5×5 diamond shape is applied for chroma components.

[00102] FIG. 11 shows ALF filter shapes (chroma: 5×5 diamond, luma: 7×7 diamond).

2.6.2. Block classification

[00103] For luma component, each 4×4 block is categorized into one out of 25 classes. The classification index C is derived based on its directionality D and a quantized value of activity $\hat{A}$, as follows:

$$C = 5D + \hat{A} \tag{2-1}$$

[00104] To calculate D and $\hat{A}$, gradients of the horizontal, vertical and two diagonal direction are first calculated using 1-D Laplacian:

$$g_v = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} V_{k,l}, \; V_{k,l} = |2R(k,l) - R(k,l-1) - R(k,l+1)| \tag{2-2}$$

$$g_h = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} H_{k,l}, \; H_{k,l} = |2R(k,l) - R(k-1,l) - R(k+1,l)| \tag{2-3}$$

$$g_{d1} = \sum_{k=i-2}^{i+3} \sum_{l=j-3}^{j+3} D1_{k,l}, D1_{k,l} = |2R(k,l) - R(k-1,l-1) - R(k+1,l+1)| \tag{2-4}$$

$$g_{d2} = \sum_{k=i-2}^{i+3} \sum_{j=j-2}^{j+3} D2_{k,l}, \; D2_{k,l} = |2R(k,l) - R(k-1,l+1) - R(k+1,l-1)| \tag{2-5}$$

[00105] Where indices i and j refer to the coordinates of the upper left sample within the 4×4 block and $R(i,j)$ indicates a reconstructed sample at coordinate (i,j).

[00106] To reduce the complexity of block classification, the subsampled 1-D Laplacian calculation is applied. FIG. 12 shows subsampled Laplacian calculation. *FIG. 12A shows* subsampled positions for vertical gradient, FIG. 12B shows subsampled positions for horizontal gradient, FIG. 12C shows subsampled positions for diagonal gradient, and FIG. 12D shows subsampled positions for diagonal gradient.

[00107] Then D maximum and minimum values of the gradients of horizontal and vertical directions are set as:

$$g_{h,v}^{max} = max(g_h, g_v), \;\; g_{h,v}^{min} = min(g_h, g_v) \tag{2-6}$$

[00108] The maximum and minimum values of the gradient of two diagonal directions are set as:

$$g_{d0,d1}^{max} = max(g_{d0}, g_{d1}), \;\; g_{d0,d1}^{min} = min(g_{d0}, g_{d1}) \tag{2-7}$$

[00109] To derive the value of the directionality D, these values are compared against each other and with two thresholds t_1 and t_2:

Step 1. If both $g_{h,v}^{max} \leq t_1 \cdot g_{h,v}^{min}$ and $g_{d0,d1}^{max} \leq t_1 \cdot g_{d0,d1}^{min}$ are true, D is set to 0.

Step 2. If $g_{h,v}^{max} / g_{h,v}^{min} > g_{d0,d1}^{max} / g_{d0,d1}^{min}$, continue from Step 3; otherwise continue from Step 4.

Step 3. If $g_{h,v}^{max} > t_2 \cdot g_{h,v}^{min}$, D is set to 2; otherwise D is set to 1.

Step 4. If $g_{d0,d1}^{max} > t_2 \cdot g_{d0,d1}^{min}$, D is set to 4; otherwise D is set to 3.

The activity value A is calculated as:

$$A = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} (V_{k,l} + H_{k,l}) \quad (2\text{-}8)$$

A is further quantized to the range of 0 to 4, inclusively, and the quantized value is denoted as $\hat{A}$.

[00110] For chroma components in a picture, no classification method is applied, i.e. a single set of ALF coefficients is applied for each chroma component.

2.6.3. Geometric transformations of filter coefficients and clipping values

[00111] Before filtering each 4×4 luma block, geometric transformations such as rotation or diagonal and vertical flipping are applied to the filter coefficients $f(k,l)$ and to the corresponding filter clipping values $c(k,l)$ depending on gradient values calculated for that block. This is equivalent to applying these transformations to the samples in the filter support region. The idea is to make different blocks to which ALF is applied more similar by aligning their directionality.

Three geometric transformations, including diagonal, vertical flip and rotation are introduced:

$$\text{Diagonal: } f_D(k,l) = f(l,k),\ c_D(k,l) = c(l,k), \quad (2\text{-}9)$$

$$\text{Vertical flip: } f_V(k,l) = f(k,K-l-1),\ c_V(k,l) = c(k,K-l-1) \quad (2\text{-}10)$$

$$\text{Rotation: } f_R(k,l) = f(K-l-1,k),\ c_R(k,l) = c(K-l-1,k) \quad (2\text{-}11)$$

[00112] where K is the size of the filter and $0 \le k,l \le K-1$ are coefficients coordinates, such that location $(0,0)$ is at the upper left corner and location $(K-1,K-1)$ is at the lower right corner. The transformations are applied to the filter coefficients $f(k, l)$ and to the clipping values $c(k,l)$ depending on gradient values calculated for that block. The relationship between the transformation and the four gradients of the four directions are summarized in the following table.

Table 2-5. Mapping of the gradient calculated for one block and the transformations

Gradient values	Transformation
$g_{d2} < g_{d1}$ and $g_h < g_v$	No transformation
$g_{d2} < g_{d1}$ and $g_v < g_h$	Diagonal
$g_{d1} < g_{d2}$ and $g_h < g_v$	Vertical flip
$g_{d1} < g_{d2}$ and $g_v < g_h$	Rotation

2.6.4. Filter parameters signalling

[00113] ALF filter parameters are signalled in Adaptation Parameter Set (APS). In one APS, up to 25 sets of luma filter coefficients and clipping value indexes, and up to eight sets of chroma filter coefficients and clipping value indexes could be signalled. To reduce bits overhead, filter

coefficients of different classification for luma component can be merged. In slice header, the indices of the APSs used for the current slice are signaled.

[00114] Clipping value indexes, which are decoded from the APS, allow determining clipping values using a table of clipping values for both luma and Chroma components. These clipping values are dependent of the internal bitdepth. More precisely, the clipping values are obtained by the following formula:

$$\text{AlfClip}= \{\text{round}(2^{B-\alpha*n}) \text{ for } n \in [0..N-1]\} \quad (2\text{-}12)$$

[00115] with B equal to the internal bitdepth, α is a pre-defined constant value equal to 2.35, and N equal to 4 which is the number of allowed clipping values in VVC.

[00116] In slice header, up to 7 APS indices can be signaled to specify the luma filter sets that are used for the current slice. The filtering process can be further controlled at CTB level. A flag is always signalled to indicate whether ALF is applied to a luma CTB. A luma CTB can choose a filter set among 16 fixed filter sets and the filter sets from APSs. A filter set index is signaled for a luma CTB to indicate which filter set is applied. The 16 fixed filter sets are pre-defined and hard-coded in both the encoder and the decoder.

[00117] For chroma component, an APS index is signaled in slice header to indicate the chroma filter sets being used for the current slice. At CTB level, a filter index is signaled for each chroma CTB if there is more than one chroma filter set in the APS.

[00118] The filter coefficients are quantized with norm equal to 128. In order to restrict the multiplication complexity, a bitstream conformance is applied so that the coefficient value of the non-central position shall be in the range of -2^7 to $2^7 - 1$, inclusive. The central position coefficient is not signalled in the bitstream and is considered as equal to 128.

2.6.5. Filtering process

[00119] At decoder side, when ALF is enabled for a CTB, each sample $R(i,j)$ within the CU is filtered, resulting in sample value $R'(i,j)$ as shown below,

$$R'(i,j) = R(i,j) + \left(\left(\sum_{k\neq 0}\sum_{l\neq 0} f(k,l) \times K\big(R(i+k,j+l) - R(i,j), c(k,l)\big) + 64\right) \gg 7\right) \quad (2\text{-}13)$$

[00120] where $f(k,l)$ denotes the decoded filter coefficients, $K(x,y)$ is the clipping function and $c(k,l)$ denotes the decoded clipping parameters. The variable k and l varies between $-\frac{L}{2}$ and $\frac{L}{2}$ where L denotes the filter length. The clipping function $K(x,y) = \min(y, \max(-y,x))$ which corresponds to the function $Clip3\ (-y,y,x)$.

2.6.6. Virtual boundary filtering process for line buffer reduction

[00121] In hardware and embedded software, picture-based processing is practically unacceptable due to its high picture buffer requirement. Using on-chip picture buffers is very expensive and using off-chip picture buffers significantly increases external memory access, power consumption, and data access latency. Therefore, DF, SAO, and ALF will be changed from picture-based to LCU-based decoding in real products. When LCU-based processing is used for DF, SAO, and ALF, the entire decoding process can be done LCU by LCU in a raster scan with an LCU-pipelining fashion for parallel processing of multiple LCUs. In this case, line buffers are required for DF, SAO, and ALF because processing one LCU row requires pixels from the above LCU row. If off-chip line buffers (e.g. DRAM) are used, the external memory bandwidth and power consumption will be increased; if on-chip line buffers (e.g. SRAM) are used, the chip area will be increased. Therefore, although line buffers are already much smaller than picture buffers, it is still desirable to reduce line buffers.

[00122] In VTM-4.0, as shown in FIG. 13, the total number of line buffers required is 11.25 lines for the Luma component. The explanation of the line buffer requirement is as follows: The deblocking of horizontal edge overlapping with CTU edge cannot be performed as the decisions and filtering require lines K, L, M, M from the first CTU and Lines O, P from the bottom CTU. Therefore, the deblocking of the horizontal edges overlapping with the CTU boundary is postponed until the lower CTU comes. Therefore, for the lines K, L, M, N reconstructed luma samples have to be stored in the line buffer (4 lines). Then the SAO filtering can be performed for lines A till J. The line J can be SAO filtered as deblocking does not change the samples in line K. For SAO filtering of line K, the edge offset classification decision is only stored in the line buffer (which is 0.25 Luma lines). The ALF filtering can only be performed for lines A-F. As shown in FIG. 13, the ALF classification is performed for each 4×4 block. Each 4×4 block classification needs an activity window of size 8×8 which in turn needs a 9×9 window to compute the 1d Laplacian to determine the gradient.

[00123] Therefore, for the block classification of the 4×4 block overlapping with lines G, H, I, J needs, SAO filtered samples below the Virtual boundary. In addition, the SAO filtered samples of lines D, E, F are required for ALF classification. Moreover, the ALF filtering of Line G needs three SAO filtered lines D, E, F from above lines. Therefore, the total line buffer requirement is as follows:

- Lines K-N (Horizontal DF pixels): 4 lines
- Lines D-J (SAO filtered pixels): 7 lines
- SAO Edge offset classifier values between line J and line K: 0.25 line

[00124] Therefore, the total number of luma lines required is 7 + 4 + 0.25 = 11.25.

[00125] Similarly, the line buffer requirement of the Chroma component is illustrated in FIG. 14. The line buffer requirement for Chroma component is evaluated to be 6.25 lines.

[00126] FIG. 13 shows a loop filter line buffer requirement in VTM-4.0 for Luma component.

[00127] FIG. 14 shows a Loop filter line buffer requirement in VTM-4.0 for Chroma component.

[00128] In order to eliminate the line buffer requirements of SAO and ALF, the concept of virtual boundary (VB) is introduced to reduce the line buffer requirement of ALF in the latest VVC. Modified block classification and filtering are employed for the samples near horizontal CTU boundaries. As shown in FIG. 13, VBs are upward shifted horizontal LCU boundaries by N pixels. For each LCU, SAO and ALF can process pixels above the VB before the lower LCU comes but cannot process pixels below the VB until the lower LCU comes, which is caused by DF. With consideration of the hardware implementation cost, the space between the proposed VB and the horizontal LCU boundary is set as four pixels for luma component (i.e. N=4 in FIG. 13 or FIG. 15) and two pixels for chroma component (i.e. N=2).

[00129] FIG. 15 shows a modified block classification at virtual boundaries

[00130] Modified block classification is applied for the Luma component as depicted in FIG. 16. For the 1D Laplacian gradient calculation of the 4×4 block above the virtual boundary, only the samples above the virtual boundary are used. Similarly, for the 1D Laplacian gradient calculation of the 4×4 block below the virtual boundary, only the samples below the virtual boundary are used. The quantization of activity value A is accordingly scaled by taking into account the reduced number of samples used in 1D Laplacian gradient calculation.

[00131] For filtering processing, mirrored (symmetric) padding operation at the virtual boundaries are used for both Luma and Chroma components. As shown in FIG. 16 when the sample being filtered is located below the virtual boundary, the neighboring samples that are located above the virtual boundary are padded. Meanwhile, the corresponding samples at the other sides are also padded, symmetrically.

[00132] FIG. 16 shows a modified ALF filtering for Luma component at virtual boundaries

[00133] For another example, if one sample located at (i, j) (e.g., the P0A with dash line in FIG. 17B is padded, then the corresponding sample located at (m, n) (e.g., the P3B with dash line in FIG. 17B which share the same filter coefficient is also padded even the sample is available, as depicted in FIGS. 17A-17C.

[00134] FIG. 17A shows one required line above/below VB need to be padded (per side).

[00135] FIG. 17B shows 2 required lines above/below VB need to be padded (per side).

[00136] FIG. 17C shows 3 required lines above/below VB need to be padded (per side).

[00137] FIG. 27 shows examples of modified luma ALF filtering at virtual boundary

[00138] Different to the mirrored (symmetric) padding method used at horizontal CTU boundaries, repetitive (one-side) padding process is applied for slice, tile and subpicture boundaries when filter across the boundaries is disabled. The repetitive (one-side) padding process is also applied at picture boundary. The padded samples are used for both classification and filtering process. FIG. 18 depicts an example of repetitive padding method for luma ALF filtering at picture/subpicture/slice/tile boundary.

[00139] FIG. 18 shows examples of repetitive padding for luma ALF filtering at picture/subpicture/slice/tile boundary

2.7. 360-degree video coding

[00140] The horizontal wrap around motion compensation in the VTM5 is a 360-specific coding tool designed to improve the visual quality of reconstructed 360-degree video in the equi-rectangular (ERP) projection format. In conventional motion compensation, when a motion vector refers to samples beyond the picture boundaries of the reference picture, repetitive padding is applied to derive the values of the out-of-bounds samples by copying from those nearest neighbors on the corresponding picture boundary. For 360-degree video, this method of repetitive padding is not suitable, and could cause visual artefacts called “seam artefacts” in a reconstructed viewport video. Because a 360-degree video is captured on a sphere and inherently has no “boundary,” the reference samples that are out of the boundaries of a reference picture in the projected domain can always be obtained from neighboring samples in the spherical domain. For a general projection format, it may be difficult to derive the corresponding neighboring samples in the spherical domain, because it involves 2D-to-3D and 3D-to-2D coordinate conversion, as well as sample interpolation for fractional sample positions. This problem is much simpler for the left and right boundaries of the ERP projection format, as the spherical neighbors outside of the left picture boundary can be obtained from samples inside the right picture boundary, and vice versa. FIG. 19 shows an example of horizontal wrap around motion compensation in VVC

[00141] The horizontal wrap around motion compensation process is as depicted in FIG. 19 When a part of the reference block is outside of the reference picture’s left (or right) boundary in the projected domain, instead of repetitive padding, the “out-of-boundary” part is taken from the corresponding spherical neighbors that are located within the reference picture toward the right (or left) boundary in the projected domain. **Repetitive padding is only used for the top and bottom picture boundaries**. As depicted in FIG. 19, the horizontal wrap around motion

compensation can be combined with the non-normative padding method often used in 360-degree video coding. In VVC, this is achieved by signaling a high-level syntax element to indicate the wrap-around offset, which should be set to the ERP picture width before padding; this syntax is used to adjust the position of horizontal wrap around accordingly. This syntax is not affected by the specific amount of padding on the left and right picture boundaries, and therefore naturally supports asymmetric padding of the ERP picture, i.e., when left and right padding are different. The horizontal wrap around motion compensation provides more meaningful information for motion compensation when the reference samples are outside of the reference picture's left and right boundaries.

[00142] For projection formats composed of a plurality of faces, no matter what kind of compact frame packing arrangement is used, discontinuities appear between two or more adjacent faces in the frame packed picture. For example, considering the 3×2 frame packing configuration depicted in FIG. 20, the three faces in the top half are continuous in the 3D geometry, the three faces in the bottom half are continuous in the 3D geometry, but the top and bottom halves of the frame packed picture are discontinuous in the 3D geometry. If in-loop filtering operations are performed across this discontinuity, face seam artifacts may become visible in the reconstructed video.

[00143] To alleviate face seam artifacts, in-loop filtering operations may be disabled across discontinuities in the frame-packed picture. A syntax was proposed to signal **vertical and/or horizontal virtual boundaries** across which the in-loop filtering operations are disabled. Compared to using two tiles, one for each set of continuous faces, and to disable in-loop filtering operations across tiles, the proposed signaling method is more flexible as it does not require the face size to be a multiple of the CTU size.

[00144] FIG. 20 shows an in image of HEC in 3×2 layout.

2.8. JVET-P0080: CE5-2.1, CE5-2.2: Cross Component Adaptive Loop Filter

[00145] FIG. 21A illustrates the placement of CC-ALF [1] with respect to the other loop filters. CC-ALF operates by applying a linear, diamond shaped filter FIG. 21B to the luma channel for each chroma component, which is expressed as

$$\Delta I_i(x,y) = \sum_{(x_0,y_0)\in S_i} I_0(x_C + x_0, y_C + y_0) c_i(x_0, y_0),$$

where

(x, y) is chroma component i location being refined

(x_C, y_C) is the luma location based on (x, y)

S_i is filter support in luma for chroma component i

$c_i(x_0, y_0)$ represents the filter coefficients

(2-14)

[00146] FIG. 21A shows an example placement of CC-ALF with respect to other loop filters. FIG. 21B shows a Diamond shaped filter.

[00147] The luma location (x_C, y_C), around which the support region is centered, is computed based on the spatial scaling factor between the luma and chroma planes. All filter coefficients are transmitted in the APS and have 8-bit dynamic range. An APS may be referenced in the slice header. CC-ALF coefficients used for each chroma component of a slice are also stored in a buffer corresponding to a temporal sublayer. Reuse of these sets of temporal sublayer filter coefficients is facilitated using slice-level flags. The application of the CC-ALF filters is controlled on a variable block size (i.e. 16×16, 32×32, 64×64, 128×128) and signalled by a context-coded flag received for each block of samples. The block size along with an CC-ALF enabling flag is received at the slice-level for each chroma component. Boundary padding for the horizontal virtual boundaries makes use of repetition. For the remaining boundaries the same type of padding is used as for regular ALF.

2.8.1. Specification on CC-ALF in JVET-P0080

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

- a reconstructed luma picture sample array recPicture_L prior to the luma adaptive loop filtering process,
- a filtered reconstructed chroma picture sample array alfPicture_C,
- a chroma location (xC, yC) specifying the top left sample of the current block of chroma samples relative to the top left sample of the current picture,
- a width ccAlfWidth of block of chroma samples
- a height ccAlfHeight of block of chroma samples
- cross component filter coefficients CcAlfCoeff[j], with j = 0..13

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xC + x][yC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xC + x][yC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

– The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xC + x, yC + y) is set equal to ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
– The luma locations (h_{xL+i}, v_{yL+j}) with i = −2..2, j = −2..3 inside the array $recPicture_L$ are derived as follows:
– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)

– Otherwise, the following applies:

h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)

– Otherwise, the following applies:

$$v_{y+j} = \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, yL + j) \quad (8\text{-}1234)$$

– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.
– The vertical sample position offsets yM2, yM1, yP1, yP2 and yP3 are specified in Table 2-6 according to the vertical luma sample position yL, clipLeftPos and clipRightPos.
– The horizontal sample position offsets xM1, xM2, xP1 and xP2 are specified in Table 2-7 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.
– The variable curr is derived as follows:

$$\text{curr} = \text{alfPicture}_C[xC + x, yC + y] \quad (8\text{-}1286)$$

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..13:

$$f[j] = \text{CcAlfCoeff}[j] \quad (8\text{-}1287)$$

– The variable sum is derived as follows:

$$\begin{aligned}\text{sum} = {} & f[0] * \text{recPicture}_L[h_x, v_{y+yM2}] + \\ & f[1] * \text{recPicture}_L[h_{x+xM1}, v_{y+yM1}] + \\ & f[2] * \text{recPicture}_L[h_x, v_{y+yM1}] + \\ & f[3] * \text{recPicture}_L[h_{x+xP1}, v_{y+yM1}] + \\ & f[4] * \text{recPicture}_L[h_{x+xM2}, v_y] + \\ & f[5] * \text{recPicture}_L[h_{x+xM1}, v_y] + \\ & f[6] * \text{recPicture}_L[h_x, v_y] + \\ & f[7] * \text{recPicture}_L[h_{x+xP1}, v_y] + \\ & f[4] * \text{recPicture}_L[h_{x+xP2}, v_y] + \\ & f[4] * \text{recPicture}_L[h_{x+xM2}, v_{y+yP1}] + \\ & f[8] * \text{recPicture}_L[h_{x+xM1}, v_{y+yP1}] + \\ & f[9] * \text{recPicture}_L[h_x, v_{y+yP1}] + \\ & f[10] * \text{recPicture}_L[h_{x+xP1}, v_{y+yP1}] + \\ & f[4] * \text{recPicture}_L[h_{x+xP2}, v_{y+yP1}] + \\ & f[11] * \text{recPicture}_L[h_{x+xM1}, v_{y+yP2}] + \\ & f[12] * \text{recPicture}_L[h_x, v_{y+yP2}] + \\ & f[13] * \text{recPicture}_L[h_{x+xP1}, v_{y+yP2}] + \\ & f[0] * \text{recPicture}_L[h_x, v_{y+yP3}]\end{aligned} \quad (8\text{-}1289)$$

$$\text{sum} = \text{curr} + (\text{sum} + 64) >> 7) \quad (8\text{-}1290)$$

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xC + x][yC + y] is derived as follows:

ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << $BitDepth_C$) − 1, sum) (8-1291)

Table 2-6. Specification of yM1, yM2, yP1, yP2 and yP3 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Condition	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 1	−1	−1	1	2	3
yL = = clipTopPos	0	0	1	2	3
yL = = clipBottomPos − 1	−2	−1	0	0	0
yL = = clipBottomPos − 2	−2	−1	1	1	1
yL = = clipBottomPos − 3	−2	−1	1	2	2
Otherwise	−2	−1	1	2	3

Table 2-7. Specification of xM1, xM2, xP1, and xP2 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM2	xM1	xP1	xP2
xL = = clipLeftPos + 1	–1	–1	1	2
xL = = clipLeftPos	0	0	1	2
xL = = clipRightPos − 1	–2	–1	0	0
xL = = clipRightPos – 2	–2	–1	1	1
Otherwise	–2	–1	1	2

2.8.2. Padding method at virtual boundary in JVET-P0080

[00148] Similar to luma ALF/chroma ALF, repetitive padding is utilized at ALF virtual boundary for CC-ALF in JVET-P0080. As shown in FIG. 22, if the luma samples above or below the ALF virtual boundary are unavailable, the nearest sample line is utilized for padding. The detailed padding method is also shown in Table 2-6.

2.9. JVET-P1008: CE5-related: On the design of CC-ALF

[00149] In JVET-O0636 [1] and CE5-2.1 [2], the Cross Component Adaptive Loop Filter (CC-ALF) was introduced and studied. The filter uses a linear filter to filter luma sample values and generate a residual correction for the chroma channels from the co-located filtered output. The filter is designed to operate in parallel with the existing luma ALF.

[00150] A CC-ALF design is proposed that is asserted to be both simplified and better aligned with the existing ALF. The design uses a 3x4 diamond shape with 8 unique coefficients. This reduces the number of multiplies by 43% compared to the 5x6 design studied in CE5-2.1. When

a restriction is placed that enables either chroma ALF or CC-ALF for chroma component of a CTU we limit the per-pixel multiplier count to 16 (current ALF is 15). The filter coefficient dynamic range is limited to 6-bit signed. An illustration of the filters for both the proposed and CE5-2.1 solution are shown in FIG. 23.

[00151] To be better aligned with the existing ALF design, the filter coefficients are signaled in the APS. Up to four filters are supported, and filter selection is indicated at the CTU-level. Symmetric line selection is used at the virtual boundary to further harmonize with ALF. Finally, to limit the amount of storage needed by the correction output, the CC-ALF residual output is clipped to $-2^{BitDepthC-1}$ to $2^{BitDepthC-1}-1$, inclusive.

[00152] Specification on CC-ALF in JVET-P1008.

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

- a reconstructed luma picture sample array $recPicture_L$ prior to the luma adaptive loop filtering process,
- a filtered reconstructed chroma picture sample array $alfPicture_C$,
- a chroma location (xCtbC, yCtbC) specifying the top-left sample of the current chroma coding tree block relative to the top left sample of the current picture,
- a width ccAlfWidth of block of chroma samples
- a height ccAlfHeight of block of chroma samples
- cross component filter coefficients CcAlfCoeff[j], with j = 0..7

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xCtbC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yCtbC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY(8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xCtbC + x][yCtbC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xCtbC + x][yCtbC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

- The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xCtbC + x, yCtbC + y) is set equal to ((xCtbC + x) * SubWidthC, (yCtbC + y) * SubHeightC)

– The luma locations (h_{xL+i}, v_{yL+j}) with i = −1..1, j = −1..2 inside the array $recPicture_L$ are derived as follows:
  – If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

    h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)
  – Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

    h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)
  – Otherwise, the following applies:

    h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)
  – If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

    v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)
  – Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

    v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)
  – Otherwise, the following applies:

    v_{y+j} = Clip3(0, pic_height_in_luma_samples − 1, yL + j) (8-1234)
– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.
– The vertical sample position offsets yM1, yP1 and yP2 are specified in Table 2-8 according to the vertical luma sample position yL, clipLeftPos and clipRightPos.

– The horizontal sample position offsets xM1 and xP1 are specified in Table 2-9 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.
– The variable curr is derived as follows:

$$curr = alfPicture_C[xCtbC + x, yCtbC + y] \quad (8\text{-}1286)$$

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..7:

$$f[j] = CcAlfCoeff[j] \quad (8\text{-}1287)$$

– The variable sum is derived as follows:

$$\begin{aligned} sum = {} & f[0] * recPicture_L[h_x, v_{y + yM1}] + \\ & f[1] * recPicture_L[h_{x+xM1}, v_y] + \\ & f[2] * recPicture_L[h_x, v_y] + \\ & f[3] * recPicture_L[h_{x+xP1}, v_y] + \\ & f[4] * recPicture_L[h_{x+xM1}, v_{y+yP1}] + \\ & f[5] * recPicture_L[h_x, v_{y+yP1}] + \\ & f[6] * recPicture_L[h_{x+xP1}, v_{y+yP1}] + \\ & f[7] * recPicture_L[h_x, v_{y + yP2}] \end{aligned} \quad (8\text{-}1289)$$

$$sum = Clip3(-(1 << (BitDepth_C - 1)), (1 << (BitDepth_C - 1)) - 1, sum) \quad (8\text{-}1290)$$

$$sum = curr + (sum + 64) >> (7 + (BitDepth_Y - BitDepth_C)) \quad (8\text{-}1290)$$

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xCtbC + x][yCtbC + y] is derived as follows:

$$ccAlfPicture[xCtbC + x][yCtbC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum) \quad (8\text{-}1291)$$

Table 2-8. Specification of yM1, yP1 and yP2 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Condition	yM1	yP1	yP2
yL = = clipTopPos + 1	−1	1	1
yL = = clipTopPos	0	0	1
yL = = clipBottomPos − 1	0	0	1
yL = = clipBottomPos − 2	−1	1	1
Otherwise	−1	1	2

Table 2-9. Specification of xM1 and xP1 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM1	xP1
xL = = clipLeftPos	0	0
xL = = clipRightPos − 1	0	0
xL = = clipRightPos – 2	–1	1
Otherwise	–1	1

2.9.1. Padding method at virtual boundary in JVET-P1008

[00153] Mirrored (symmetric) padding is utilized at ALF virtual boundary for CC-ALF in JVET-P1008. As shown in FIG. 24, if the luma samples above or below the ALF virtual boundary are unavailable, the nearest sample line is utilized for padding, and the corresponding samples also need to be padded. The detailed padding method is also shown in Table 2-9.

2.10. Simplified methods of CC-ALF in JVET-P2025

2.10.1. Alternative filter shapes

[00154] The CC-ALF filter shape is modified to have 8 or 6 coefficients as shown in the figure below.

[00155] FIG. 25 shows CC-ALF filter shape of 8 coefficients in JVET-P0106.

[00156] FIG. 26 shows CC-ALF filter shape of 6 coefficients in JVET-P0173.

[00157] FIG. 27 shows CC-ALF filter shape of 6 coefficients in JVET-P0251.

2.10.2. Joint chroma cross-component adaptive filtering

[00158] Joint Chroma Cross-Component Adaptive Loop Filter (JC-CCALF) uses only one set of CCALF filter coefficients trained at the encoder to generate one filtered output as the refinement signal, which will be added directly to the Cb component, and be properly weighted and then added to the Cr component. Filters are indicated at the CTU-level or indicated with a block size, which is signalled per slice.

[00159] The supported such chroma block sizes range from the minimum chroma CTU size to the current chroma CTU size. The minimum chroma CTU size is the minimum between the smallest possible width and height of a chroma CTU, i.e. Min(32 / SubWidthC, 32 / SubHeightC), while the current chroma CTU size is the minimum between the width and height of the current chroma CTU, i.e. Min(CtbWidthC, CtbHeightC). For example, if CTU size is set to the maximal 128×128, the JC-CCALF chroma block size of a slice will be one from 32×32, 64×64 and 128×128 for 4:4:4 video, or one from 16×16, 32×32 and 64×64 for 4:2:0 and 4:2:2 video.

[00160] FIG. 28 shows a JC-CCALF workflow.

3. Technical problems solved by technical solutions described herein

[00161] The current design of boundary padding for CC-ALF has the following problems:

1. The padding method at ALF virtual boundary in CC-ALF may be sub-optimal, since padded samples are utilized which may be less efficient.
2. Different ways for handling ALF virtual boundary and video unit boundary (e.g., picture/subpicture/slice/tile boundary) and 360-degree virtual boundary, i.e., different padding methods are existing.
3. In ALF, the mirror padding is applied wherein the distance to the current sample is calculated to determine which corresponding sample needs to be padded. However, in CC-ALF, especially, for 4:2:0, to filter one chroma sample, multiple luma samples are involved. How to determine which corresponding sample needs to be padded is unknown.

4. Example listing of techniques and embodiments

[00162] The listing below should be considered as examples to explain general concepts. These items should not be interpreted in a narrow way. Furthermore, these items can be combined in any manner.

[00163] In some embodiments described in this disclosure, the term 'CC-ALF' represents a coding tool that utilizes the sample values in a second color component (e.g., Y) or multiple color components (e.g., both Y and Cr) to refine the samples in a first color component (e.g., Cb). It is not limited to the CC-ALF technologies described in [1]-[4]. "corresponding filtering sample set" may be used to represent those samples included in a filter support, e.g., for CC-ALF, the "corresponding filtering sample set" may be used to represent the collocated luma sample and neighboring luma samples of the collocated luma sample of a chroma sample which are utilized to derive the refinement/offset of the chroma sample.

[00164] The padding method used for ALF virtual boundaries may be denoted as 'Mirrored Padding' wherein for a first unavailable sample located at (i, j), is padded, and a second sample, defined by 'corresponding sample of the first sample' in the filter support (e.g., the corresponding sample located at (m, n) which share the same distance from the current luma sample) in ALF is also padded even if the second sample is available.

[00165] In one example, vertical padding is utilized, such as the sample to be padded located at (x, y1) is set equal to the sample located at (x, y2), wherein y1 denotes the y-coordinate of the sample or the corresponding sample and y2 denotes the y-coordinate of the sample utilized for padding.

[00166] In one example, horizontal padding is utilized, such as the sample to be padded located at (x1, y) is set equal to the sample located at (x2, y), wherein x1 denotes the x-coordinate of the sample or the corresponding sample and x2 denotes the x-coordinate of the sample utilized for padding.

[00167] The padding method used for picture/subpicture/slice/tile boundaries/360-degree video virtual boundaries, normal boundaries (e.g, top and bottom boundaries) may be denoted as 'Repetitive Padding' wherein if one sample to be used is outside the boundaries, it is copied from an available one inside the boundary.

[00168] In the disclosure, a neighbouring (adjacent or non-adjacent) sample is "unavailable" if it is located in a different video processing unit (e.g., out of: the current picture, or current sub-picture, or current tile, or current slice, or current brick, or current CTU, or current processing unit (such as ALF processing unit or narrow ALF processing unit), or any other current video unit) or not reconstructed or cross-filtering video processing unit is disallowed.

<u>Handling ALF virtual boundary for CC-ALF</u>

1. For an unavailable luma sample to be padded at the ALF virtual boundary, mirrored padding may be utilized to derive the unavailable luma sample and one or multiple corresponding luma samples of the unavailable luma sample, for filtering in CC-ALF. That is, at least one corresponding luma sample of the unavailable sample needs to be padded as well even it is available.
    a. In one example, a luma sample that is determined as a corresponding sample of an unavailable luma sample may be padded using the mirrored padding method.
    b. In one example, whether a luma sample (in the corresponding filtering sample set) is determined as a corresponding sample of an unavailable sample may be dependent on the distance of the sample relative to a representative luma sample or/and the distance of the unavailable sample relative to the representative luma sample. Denote the center row wherein the representative luma sample is located by *C*. Suppose *K*×*L* filter shape which makes use of *K* rows of samples and *L* columns of samples is used in CC-ALF.
        i. In one example, the representative luma sample is defined as the collocated luma sample of current chroma sample to be filtered.
            1) In one example, the position of the collocated luma sample of current chroma sample may depend on the color format.

a) In one example, the collocated luma sample of a chroma sample located at (x, y) is defined as the one located at (2x, 2y) in 4:2:0 chroma format.
b) In one example, the collocated luma sample of a chroma sample located at (x, y) is defined as the one located at (2x, y) in 4:2:2 chroma format.
c) In one example, the collocated luma sample of a chroma sample located at (x, y) is defined as the one located at (x, y) in 4:4:4 chroma format.

ii. In one example, the distance may refer to the vertical distance between a row containing a luma sample and the row containing the representative luma sample. For example, the distance may be calculated as the absolute y-coordinate difference between a luma sample and the representative luma sample.
    1) As shown in FIG. 29, denote the center row wherein the representative luma sample is located, the row of an unavailable sample, and the row of the corresponding samples as *C*, *M*, and *N* respectively, and *M* is not equal to *N*. Deonte d(x, y) as the absolute y-coordinate difference between x and y, meaning that the distance between row x and row y.

iii. In one example, the determination of the corresponding samples to be padded in mirrored padding may be dependent on how many rows of samples would be utilized by the filter shape.

iv. In one example, if the unavailable sample is located at row *M* (e.g., $M < C < N$ or $M > C > N$), then samples located at row *N* are determined as the corresponding samples to be padded, when $d(C, M) = d(N, C)$.
    1) In one example, If the value *K* (e.g., *K*×*L* CC-ALF filter shape) is odd, the mirrored padding method for ALF (e.g., FIG. 16) may be utilized for CC-ALF wherein the center luma sample is selected as the representative luma sample.
        a) In one example, suppose $K = 5$ and denote yM2 = -2, yM1 = -1, yL = 0, yP1 = 1, yP2 = 2 as the y-coordinator of the

five sample rows respectively, shown in Table 4-5. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.
    1. In one example, when yL is equal to CtbSizeY – 3 and the row yM2 is unavailable, the samples (x, yM2) at the row yM2 may be padded using samples (x, yM1) at the row yM1. Meanwhile, the samples (x, yP2) at the corresponding row yP2 may be padded using samples (x, yP1) at the row yP1.
    2. In one example, when yL is equal to CtbSizeY – 4 and the rows yM2 and yM1 are unavailable, the samples (x, yM2) and (x, yM1) at the row yM2 and yM1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yP2) and (x, yP1) at the corresponding row yP2 and yP1 may be padded using samples (x, yL) at the row yL.
ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 6 and the row yP2 is unavailable, the samples (x, yP2) at the row yP2 may be padded using samples (x, yP1) at the row yP1. Meanwhile, the samples (x, yM2) at the corresponding row yM2 may be padded using samples(x, yM1) ath the row yM1.
2. In one example, when yL is equal to CtbSizeY – 5 and the rows yP2 and yP1 are unavailable, the samples (x, yP2) and (x, yP1) at the row yP2 and yP1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yM2) and (x, yM1) at the corresponding row yM2 and yM1 may be padded using samples (x, yL) at the row yL.

2) In one example, If the value *K* (e.g., *K*×*L* CC-ALF filter shape) is even, the mirrored padding method defined in FIG. 30 may be utilized. When the unavailable samples located at row *M* (*N*) above (below) the ALF virtual boundary and they are padded from the nearest sample row below (above) the ALF virtual boundary. It is proposed that the corresponding samples located at row *N* (*M*) below (above) the ALF virtual boundary may be padded from the nearest sample row above (below) row *N* (*M*).

a) In one example, suppose *K* = 2 and denote yL = 0 and yP1 = 1 as the y-coordinator of the two sample rows, shown in Table 4-1. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary.

Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 4 and the row above yL is unavailable, the samples (x, yP1) at the corresponding row yP1 may be padded using samples (x, yL) at the row yL.

ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 5 and the row yP1 is unavailable, the samples (x, yP1) at the row yP1 may be padded using samples (x, yL) at the row yL.

b) In one example, suppose *K* = 4 and denote yM1 = -1, yL = 0, yP1 = 1, yP2 = 2 as the y-coordinator of the four sample rows respectively, shown in Table 4-3. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 3 and the row above yM1 is

unavailable, the samples (x, yP2) at the corresponding row yP2 may be padded using samples (x, yP1) at the row yP1.

2. In one example, when yL is equal to CtbSizeY – 4 and the row above yM1 and yM1 are unavailable, the samples (x, yM1) at the row yM1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yP2) and (x, yP1) at the corresponding row yP2 and yP1 may be padded using samples (x, yL) at the row yL.

ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 6 and the row yP2 is unavailable, the samples (x, yP2) at the row yP2 may be padded using samples (x, yP1) at the row yP1.
2. In one example, when yL is equal to CtbSizeY – 5 and the rows yP2 and yP1 are unavailable, the samples (x, yP2) and (x, yP1) at the row yP2 and yP1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yM1) at the corresponding row yM1 may be padded using samples (x, yL) at the row yL.

c) In one example, suppose $K = 6$ and denote yM2 = -2, yM1 = -1, yL = 0, yP1 = 1, yP2 = 2, yP3 =3 as the y-coordinator

of the six sample rows respectively, shown in Table 4-6. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.
    1. In one example, when yL is equal to CtbSizeY – 2 and the row above yM2 is unavailable, the samples (x, yP3) at the corresponding row yP3 may be padded using samples (x, yP2) at the row yP2.
    2. In one example, when yL is equal to CtbSizeY – 3 and the row above yM2 and yM2 are unavailable, the samples (x, yM2) at the row yM2 may be padded using samples (x, yM1) at the row yM1. Meanwhile, the samples (x, yP3) and (x, yP2) at the corresponding row yP3 and yP2 may be padded using samples (x, yP1) at the row yP1.
    3. In one example, when yL is equal to CtbSizeY – 4 and the row above yM2, yM2, and yM1 are unavailable, the samples (x, yM2) and (x, yM1) at the row yM2 and yM1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yP3), (x, yP2) and (x, yP1) at the corresponding row yP3, yP2 and yP1 may be padded using samples (x, yL) at the row yL.

ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 7 and the row yP3 is unavailable, the samples (x, yP3) at the row yP3 may be padded using samples (x, yP2) at the row yP2.
2. In one example, when yL is equal to CtbSizeY – 6 and the rows yP3 and yP2 are unavailable, the samples (x, yP3) and (x, yP2) at the row yP3 and yP2 may be padded using samples (x, yP1) at the row. Meanwhile, the samples (x, yM2) at the corresponding row yM2 may be padded using samples (x, yM1) at the row yM1.
3. In one example, when yL is equal to CtbSizeY – 5 and the rows yP3, yP2 and yP1 are unavailable, the samples (x, yP3), (x, yP2) and (x, yP1) at the row yP3, yP2 and yP1 may be padded using samples(x, yL) at the row yL. Meanwhile, the samples (x, yM2) and (x, yM1) at the corresponding row yM2 and yM1 may be padded using samples(x, yL) at the row.

v. In one example, if the unavailable sample is located at row *M* (e.g., *M* < *C*), then samples located at row *N* are determined as the corresponding samples to be padded, when d(*C, M*) = d(*N*, *C*) – *offset* (wherein offset is an integer value, e.g., equal to 1) or d(*C, M*) < d(*N*, *C*).

1) In one example, if the unavailable sample is located at row *M* (e.g., *M* > *C*), then samples located at row *N* are treated as the corresponding samples to be padded, when d(*M, C*) = d(*C, N*) – *offset* (wherein offset is an integer value, e.g., equal to 1) or d(*C, M*) < d(*N, C*).
2) In one example, the mirrored padding method defined in FIG. 31 may be utilized. When the unavailable samples located at row *M* (*N*) above (below) the ALF virtual boundary and they are padded from the nearest sample row below (above) the ALF virtual boundary. It is proposed that the corresponding samples located row *N* (*M*) below (above) the ALF virtual boundary may be padded from the nearest sample row above (below) row *N* (*M*).
    a) In one example, suppose *K* = 2 and denote yL = 0 and yP1 = 1 as the y-coordinator of the two sample rows, shown in Table 4-2. The ALF virtual boundary is equal to CtbSizeY – 4.
        i. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.
            1. In one example, when yL is equal to CtbSizeY – 5 and the row yP1 is unavailable, the samples (x, yP1) at the row yP1 may be padded using samples (x, yL) at the row yL.
    b) In one example, suppose *K* = 4 and denote yM1 = -1, yL = 0, yP1 = 1, yP2 = 2 as the y-coordinator of the four sample rows respectively, shown in Table 4-4. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.
    1. In one example, when yL is equal to CtbSizeY – 4 and the row above yM1 and yM1 are unavailable, the samples (x, yM1) at the row yM1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yP2) at the corresponding row yP2 may be padded using samples (x, yP1) at the row yP1.
ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.
    1. In one example, when yL is equal to CtbSizeY – 6 and the row yP2 is unavailable, the samples (x, yP2) at the row yP2 may be padded using samples (x, yP1) at the row yP1.
    2. In one example, when yL is equal to CtbSizeY – 5 and the rows yP2 and yP1 are unavailable, the samples (x, yP2) and (x, yP1) at the row yP2 and yP1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yM1) at the

corresponding row yM1 may be padded using samples (x, yL) at the row yL.

c) In one example, suppose *K* = 6 and denote yM2 = -2, yM1 = -1, yL = 0, yP1 = 1, yP2 = 2, yP3 = 3 as the y-coordinator of the six sample rows respectively, shown in Table 4-7. The ALF virtual boundary is equal to CtbSizeY – 4.

i. In one example, when the ALF virtual boundary is above the representative luma sample, the unavailable samples may be padded using the nearest row below the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row above the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 3 and the row above yM2 and yM2 are unavailable, the samples (x, yM2) at the row yM2 may be padded using samples (x, yM1) at the row yM1. Meanwhile, the samples (x, yP3) at the corresponding row yP3 may be padded using samples (x, yP2) at the row yP2.
2. In one example, when yL is equal to CtbSizeY – 4 and the row above yM2, yM2, and yM1 are unavailable, the samples (x, yM2) and (x, yM1) at the row yM2 and yM1 may be padded using samples (x, yL) at the row yL. Meanwhile, the samples (x, yP3) and (x, yP2) at the corresponding row yP3 and yP2 may be padded using samples (x, yP1) at the row yP1.

ii. In one example, when the ALF virtual boundary is below the representative luma sample, the unavailable samples may be padded using the

nearest row above the ALF virtual boundary. Meanwhile, the corresponding samples may be padded using the nearest row below the row where the corresponding samples are located at.

1. In one example, when yL is equal to CtbSizeY – 7 and the row yP3 is unavailable, the samples (x, yP3) at the row yP3 may be padded using samples (x, yP2) at the row yP2. Meanwhile, the samples (x, yM2) at the corresponding row yM2 may be padded using samples (x, yM1) at the row yM1.
2. In one example, when yL is equal to CtbSizeY – 6 and the rows yP3 and yP2 are unavailable, the samples (x, yP3) and (x, yP2) at the row yP3 and yP2 may be padded using samples (x, yP1) at the row yP1. Meanwhile, the samples (x, yM2) and (x, yM1) at the corresponding row yM2 and yM1 may be padded using samples (x, yL) at the row yL.
3. In one example, when yL is equal to CtbSizeY – 5 and the rows yP3, yP2 and yP1 are unavailable, the samples (x, yP3), (x, yP2) and (x, yP1) at the row yP3, yP2 and yP1 may be padded using samples (x, yL) at the yL. Meanwhile, the samples (x, yM2) and (x, yM1) at the corresponding row yM2 and yM1 may be padded using samples (x, yL) at the yL.

c. FIG. 29 depicts an example of the location of an unavailable sample (above the ALF virtual boundary, denoted by C0) and its corresponding sample (denoted by C7) when filtering current chroma sample located at (X_c, Y_c).

d. In one example, whether to enable or disable mirrored padding at ALF virtual boundary for CC-ALF/chroma ALF/luma ALF/other kinds of filtering methods may be signalled at sequence level/picture level/slice level/tile group level, such as in sequence header/picture header/SPS/VPS/DPS/PPS/APS/slice header/tile group header.
e. In one example, whether to enable or disable repetitive padding and/or mirrored padding at ALF virtual boundary may be dependent on coded information.
    i. In one example, the coded information may refer to block size, such as CTU/CTB size.
        1) In one example, mirrored padding may be utilized at ALF virtual boundary when the CTU/CTB size is larger than or equal to T, such as T = 32/64/128.
        2) In one example, repetitive padding may be utilized at ALF virtual boundary when the CTU/CTB size is smaller than or equal to T, such as T = 4/8/16.

2. In above bullet, the vertical padding may be replaced by horizontal padding.
    a. Alternatively, furthermore, which padding direction (vertical or horizontal) to be used may depend on whether the boundary is a horizontal boundary or vertical boundary.
    b. Alternatively, furthermore, the vertical distance may be replaced by the horizontal distance.
3. The mirrored padding method in Bullet 1 may be utilized for picture/subpicture/slice/tile boundary and/or 360-degree boundary.

<u>General solutions</u>

4. Whether to and/or how to apply the disclosed methods above may be signalled at sequence level/picture level/slice level/tile group level, such as in sequence header/picture header/SPS/VPS/DPS/PPS/APS/slice header/tile group header.
5. Whether to and/or how to apply the disclosed methods above may be dependent on coded information, such as color format, single/dual tree partitioning, the position of a sample (e.g., relative to a CU/CTU).

Table 4-1. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-2. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-3. Specification of yM1, yP1 and yP2 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yM1	yP1	yP2
(yL = = CtbSizeY − 3) && (applyAlfLineBufBoundary = = 1)	-1	1	1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0	0	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0	0	0
(yL = = CtbSizeY − 6) && (applyAlfLineBufBoundary = = 1)	-1	1	1
Otherwise	−1	1	2

Table 4-4. Specification of yM1, yP1 and yP2 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yM1	yP1	yP2
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0	1	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0	0	0
(yL = = CtbSizeY − 6) && (applyAlfLineBufBoundary = = 1)	0	1	1
Otherwise	−1	1	2

Table 4-5. Specification ofyM2, yM1, yP1 and yP2 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yM2	yM1	yP1	yP2
(yL = = CtbSizeY − 3) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0	0	0	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0	0	0	0
(yL = = CtbSizeY − 6) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	1
Otherwise	−2	-1	1	2

Table 4-6. Specification of yM2, yM1, yP1, yP2, and yP3 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yM2	yM1	yP1	yP2	yP3
(yL = = CtbSizeY − 2) && (applyAlfLineBufBoundary = = 1)	-2	-1	1	2	2
(yL = = CtbSizeY − 3) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	1	1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0	0	0	0	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0	0	0	0	0
(yL = = CtbSizeY − 6) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	1	1
(yL = = CtbSizeY − 7) && (applyAlfLineBufBoundary = = 1)	-2	-1	1	2	2
Otherwise	−2	-1	1	2	3

Table 4-7. Specification of yM2, yM1, yP1, yP2, and yP3 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yM2	yM1	yP1	yP2	yP3
(yL = = CtbSizeY − 3) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	2	2
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0	0	1	1	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0	0	0	0	0
(yL = = CtbSizeY − 6) && (applyAlfLineBufBoundary = = 1)	0	0	1	1	1
(yL = = CtbSizeY − 7) && (applyAlfLineBufBoundary = = 1)	-1	-1	1	2	2
Otherwise	−2	-1	1	2	3

5. Embodiments

The changes are highlighted by showing deletion and additions.

5.1. Embodiment #1

The working draft specified in JVET-P0080 may be changed as below.

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

- a reconstructed luma picture sample array $recPicture_L$ prior to the luma adaptive loop filtering process,
- a filtered reconstructed chroma picture sample array $alfPicture_C$,
- a chroma location (xC, yC) specifying the top left sample of the current block of chroma samples relative to the top left sample of the current picture,
- a width ccAlfWidth of block of chroma samples
- a height ccAlfHeight of block of chroma samples
- cross component filter coefficients CcAlfCoeff[j], with j = 0..13

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xC + x][yC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xC + x][yC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

- The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xC + x, yC + y) is set equal to ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
- The luma locations (h_{xL+i}, v_{yL+j}) with i = −2..2, j = −2..3 inside the array $recPicture_L$ are derived as follows:
  - If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

    h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)
  - Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)

– Otherwise, the following applies:

h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)

– Otherwise, the following applies:

v_{y+j} = Clip3(0, pic_height_in_luma_samples − 1, yL + j) (8-1234)

– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.

– The vertical sample position offsets yM2, yM1, yP1, yP2 and yP3 are specified in Table 4-1. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	**yP1**
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-2. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

– according to the vertical luma sample position yL, clipLeftPos and clipRightPos.

– The horizontal sample position offsets xM1, xM2, xP1 and xP2 are specified in Table y-yyyy according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.
– The variable curr is derived as follows:

$$curr = alfPicture_C[xC + x, yC + y] \quad (8\text{-}1286)$$

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..13:

$$f[j] = CcAlfCoeff[j] \quad (8\text{-}1287)$$

– The variable sum is derived as follows:

$$\begin{aligned} sum = {} & f[0] * recPicture_L[h_x, v_{y + yM2}] + \\ & f[1] * recPicture_L[h_{x+xM1}, v_{y + yM1}] + \\ & f[2] * recPicture_L[h_x, v_{y + yM1}] + \\ & f[3] * recPicture_L[h_{x+xP1}, v_{y + yM1}] + \\ & f[4] * recPicture_L[h_{x+xM2}, v_y] + \\ & f[5] * recPicture_L[h_{x+xM1}, v_y] + \\ & f[6] * recPicture_L[h_x, v_y] + \\ & f[7] * recPicture_L[h_{x+xP1}, v_y] + \\ & f[4] * recPicture_L[h_{x+xP2}, v_y] + \quad (8\text{-}1289) \\ & f[4] * recPicture_L[h_{x+xM2}, v_{y+yP1}] + \\ & f[8] * recPicture_L[h_{x+xM1}, v_{y+yP1}] + \\ & f[9] * recPicture_L[h_x, v_{y+yP1}] + \\ & f[10] * recPicture_L[h_{x+xP1}, v_{y+yP1}] + \\ & f[4] * recPicture_L[h_{x+xP2}, v_{y+yP1}] + \\ & f[11] * recPicture_L[h_{x+xM1}, v_{y + yP2}] + \\ & f[12] * recPicture_L[h_x, v_{y + yP2}] + \\ & f[13] * recPicture_L[h_{x+xP1}, v_{y + yP2}] + \\ & f[0] * recPicture_L[h_x, v_{y + yP3}] + \end{aligned}$$

$$sum = curr + (sum + 64) >> 7) \quad (8\text{-}1290)$$

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xC + x][yC + y] is derived as follows:

$$ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum) \quad (8\text{-}1291)$$

Table x-xx – Specification of yM1, yM2, yP1, yP2 and yP3 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Delete the following table:

Condition	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 1	−1	−1	1	2	3
yL = = clipTopPos	0	0	1	2	3
yL = = clipBottomPos − 1	−2	−1	0	0	0
yL = = clipBottomPos − 2	−2	−1	1	1	1
yL = = clipBottomPos − 3	−2	−1	1	2	2
Otherwise	−2	−1	1	2	3

Add the following table instead of the above table:

Condition	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 1	-1	-1	1	2	2
yL = = clipTopPos	0	0	1	1	1
yL = = clipBottomPos − 1	0	0	0	0	0
yL = = clipBottomPos − 2	0	0	1	1	1
yL = = clipBottomPos − 3	-1	-1	1	2	2
Otherwise	−2	-1	1	2	3

Table y-yy – Specification of xM1, xM2, xP1, and xP2 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM2	xM1	xP1	xP2
xL = = clipLeftPos + 1	–1	–1	1	2
xL = = clipLeftPos	0	0	1	2
xL = = clipRightPos − 1	–2	–1	0	0
xL = = clipRightPos – 2	–2	–1	1	1
Otherwise	–2	–1	1	2

5.2. Embodiment #2

The working draft specified in JVET-P0080 may be changed as below.

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

– a reconstructed luma picture sample array $recPicture_L$ prior to the luma adaptive loop filtering process,

– a filtered reconstructed chroma picture sample array $alfPicture_C$,

– a chroma location (xC, yC) specifying the top left sample of the current block of chroma samples relative to the top left sample of the current picture,
– a width ccAlfWidth of block of chroma samples
– a height ccAlfHeight of block of chroma samples
– cross component filter coefficients CcAlfCoeff[j], with j = 0..13

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xC + x][yC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xC + x][yC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

– The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xC + x, yC + y) is set equal to ((xC + x) * SubWidthC, (yC + y) * SubHeightC)
– The luma locations (h_{xL+i}, v_{yL+j}) with i = −2..2, j = −2..3 inside the array $recPicture_L$ are derived as follows:
– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)

– Otherwise, the following applies:

h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and

yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)

– Otherwise, the following applies:

v_{y+j} = Clip3(0, pic_height_in_luma_samples − 1, yL + j) (8-1234)

– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.

– The vertical sample position offsets yM2, yM1, yP1, yP2 and yP3 are specified in Table 4-1. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-2. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

– according to the vertical luma sample position yL, clipLeftPos and clipRightPos.

– The horizontal sample position offsets xM1, xM2, xP1 and xP2 are specified in Table y-yyyy according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.

– The variable curr is derived as follows:

curr = $alfPicture_C$[xC + x, yC + y] (8-1286)

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..13:

f[j] = CcAlfCoeff[j] (8-1287)

– The variable sum is derived as follows:

sum = f[0] * recPicture$_L$[h_x, v_{y+yM2}] +
f[1] * recPicture$_L$[h_{x+xM1}, v_{y+yM1}] +
f[2] * recPicture$_L$[h_x, v_{y+yM1}] +
f[3] * recPicture$_L$[h_{x+xP1}, v_{y+yM1}] +
f[4] * recPicture$_L$[h_{x+xM2}, v_y] +
f[5] * recPicture$_L$[h_{x+xM1}, v_y] +
f[6] * recPicture$_L$[h_x, v_y] +
f[7] * recPicture$_L$[h_{x+xP1}, v_y] +
f[4] * recPicture$_L$[h_{x+xP2}, v_y] + (8-1289)
f[4] * recPicture$_L$[h_{x+xM2}, v_{y+yP1}] +
f[8] * recPicture$_L$[h_{x+xM1}, v_{y+yP1}] +
f[9] * recPicture$_L$[h_x, v_{y+yP1}] +
f[10] * recPicture$_L$[h_{x+xP1}, v_{y+yP1}] +
f[4] * recPicture$_L$[h_{x+xP2}, v_{y+yP1}] +
f[11] * recPicture$_L$[h_{x+xM1}, v_{y+yP2}] +
f[12] * recPicture$_L$[h_x, v_{y+yP2}] +
f[13] * recPicture$_L$[h_{x+xP1}, v_{y+yP2}] +
f[0] * recPicture$_L$[h_x, v_{y+yP3}] +

sum = curr + (sum + 64) >> 7) (8-1290)

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xC + x][yC + y] is derived as follows:

ccAlfPicture[xC + x][yC + y] = Clip3(0, (1 << BitDepth$_C$) − 1, sum) (8-1291)

Table x-xx – Specification of yM1, yM2, yP1, yP2 and yP3 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Delete the following table:

Condition	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 1	−1	−1	1	2	3
yL = = clipTopPos	0	0	1	2	3
yL = = clipBottomPos − 1	−2	−1	0	0	0
yL = = clipBottomPos − 2	−2	−1	1	1	1
yL = = clipBottomPos − 3	−2	−1	1	2	2
Otherwise	−2	−1	1	2	3

Add the following table:

Condition	yM2	yM1	yP1	yP2	yP3
yL = = clipTopPos + 2	-2	-1	1	2	2
yL = = clipTopPos + 1	-1	-1	1	1	1
yL = = clipTopPos	0	0	0	0	0
yL = = clipBottomPos − 1	0	0	0	0	0
yL = = clipBottomPos − 2	0	-1	1	1	1
yL = = clipBottomPos − 3	-2	-1	1	2	2
Otherwise	−2	-1	1	2	3

Table y-yy – Specification of xM1, xM2, xP1, and xP2 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM2	xM1	xP1	xP2
xL = = clipLeftPos + 1	–1	–1	1	2
xL = = clipLeftPos	0	0	1	2
xL = = clipRightPos − 1	–2	–1	0	0
xL = = clipRightPos – 2	–2	–1	1	1
Otherwise	–2	–1	1	2

5.3. Embodiment #3

The working draft specified in JVET-P1008 may be changed as below.

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

– a reconstructed luma picture sample array $recPicture_L$ prior to the luma adaptive loop filtering process,

– a filtered reconstructed chroma picture sample array $alfPicture_C$,

– a chroma location (xCtbC, yCtbC) specifying the top-left sample of the current chroma coding tree block relative to the top left sample of the current picture,
– a width ccAlfWidth of block of chroma samples
– a height ccAlfHeight of block of chroma samples
– cross component filter coefficients CcAlfCoeff[j], with j = 0..7

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xCtbC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yCtbC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY(8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xCtbC + x][yCtbC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xCtbC + x][yCtbC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

– The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xCtbC + x, yCtbC + y) is set equal to ((xCtbC + x) * SubWidthC, (yCtbC + y) * SubHeightC)
– The luma locations (h_{xL+i}, v_{yL+j}) with i = −1..1, j = −1..2 inside the array $recPicture_L$ are derived as follows:
– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)

– Otherwise, the following applies:

h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)

– Otherwise, the following applies:

v_{y+j} = Clip3(0, pic_height_in_luma_samples − 1, yL + j) (8-1234)

– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.

– The vertical sample position offsets yM1, yP1 and yP2 are specified in Table 4-1. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	**yP1**
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-2. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

– according to the vertical luma sample position yL, clipLeftPos and clipRightPos.

– The horizontal sample position offsets xM1 and xP1 are specified in Table y-yyyy according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.

– The variable curr is derived as follows:

curr = alfPicture$_C$[xCtbC + x, yCtbC + y]

(8-1286)

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..7:

f[j] = CcAlfCoeff[j] (8-1287)

– The variable sum is derived as follows:

sum = f[0] * recPicture$_L$[h_x, v_{y+yM1}] +
f[1] * recPicture$_L$[h_{x+xM1}, v_y] +
f[2] * recPicture$_L$[h_x, v_y] +
f[3] * recPicture$_L$[h_{x+xP1}, v_y] + (8-1289)
f[4] * recPicture$_L$[h_{x+xM1}, v_{y+yP1}] +
f[5] * recPicture$_L$[h_x, v_{y+yP1}] +
f[6] * recPicture$_L$[h_{x+xP1}, v_{y+yP1}] +
f[7] * recPicture$_L$[h_x, v_{y+yP2}]

sum = Clip3(−(1 << (BitDepth$_C$ − 1)), (1 << (BitDepth$_C$ − 1)) − 1, sum) (8-1290)

sum = curr + (sum + 64) >> (7 + (BitDepth$_Y$ − BitDepth$_C$)) (8-1290)

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xCtbC + x][yCtbC + y] is derived as follows:

ccAlfPicture[xCtbC + x][yCtbC + y] = Clip3(0, (1 << BitDepth$_C$) − 1, sum)(8-1291)

Table x-xx – Specification of yM1, yP1 and yP2 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Delete tbe following table:

Condition	yM1	yP1	yP2
yL = = clipTopPos + 1	−1	1	1
yL = = clipTopPos	0	0	1
yL = = clipBottomPos − 1	0	0	1
yL = = clipBottomPos − 2	−1	1	1
Otherwise	−1	1	2

Add the following table

Condition	yM1	yP1	yP2
yL = = clipTopPos	0	1	1
yL = = clipBottomPos − 1	0	0	0
yL = = clipBottomPos − 2	0	1	1
Otherwise	−1	1	2

Table y-yy – Specification of xM1 and xP1 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM1	xP1
xL = = clipLeftPos	0	0
xL = = clipRightPos − 1	0	0
xL = = clipRightPos – 2	–1	1
Otherwise	–1	1

5.4. Embodiment #4

The working draft specified in JVET-P1008 may be changed as below.

x.x.x.x Cross component filtering process for block of chroma samples

Inputs of this process are:

- a reconstructed luma picture sample array $recPicture_L$ prior to the luma adaptive loop filtering process,
- a filtered reconstructed chroma picture sample array $alfPicture_C$,
- a chroma location (xCtbC, yCtbC) specifying the top-left sample of the current chroma coding tree block relative to the top left sample of the current picture,
- a width ccAlfWidth of block of chroma samples
- a height ccAlfHeight of block of chroma samples
- cross component filter coefficients CcAlfCoeff[j], with j = 0..7

Output of this process is the modified filtered reconstructed chroma picture sample array ccAlfPicture.

The coding tree block luma location (xCtb, yCtb) is derived as follows:

xCtb = (((xCtbC * SubWidthC) >> CtbLog2SizeY) << CtbLog2SizeY (8-1229)

yCtb = (((yCtbC * SubHeightC) >> CtbLog2SizeY) << CtbLog2SizeY(8-1229)

For the derivation of the filtered reconstructed chroma samples ccAlfPicture[xCtbC + x][yCtbC + y], each reconstructed chroma sample inside the current chroma block of samples $alfPicture_C$[xCtbC + x][yCtbC + y] with x = 0..ccAlfWidth − 1, y = 0..ccAlfHeight – 1, is filtered as follows:

- The luma location (xL, yL) corresponding to the current chroma sample at chroma location (xCtbC + x, yCtbC + y) is set equal to ((xCtbC + x) * SubWidthC, (yCtbC + y) * SubHeightC)
- The luma locations (h_{xL+i}, v_{yL+j}) with i = −1..1, j = −1..2 inside the array $recPicture_L$ are derived as follows:

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and xL − PpsVirtualBoundariesPosX[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{xL+i} = Clip3(PpsVirtualBoundariesPosX[n], pic_width_in_luma_samples − 1, xL + i) (8-1229)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosX[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosX[n]− xL is greater than 0 and less than 4 for any n = 0..pps_num_ver_virtual_boundaries − 1, the following applies:

h_{x+i} = Clip3(0, PpsVirtualBoundariesPosX[n] − 1, xL + i) (8-1230)

– Otherwise, the following applies:

h_{x+i} = Clip3(0, pic_width_in_luma_samples − 1, xL + i) (8-1231)

– If pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and yL − PpsVirtualBoundariesPosY[n] is greater than or equal to 0 and less than 3 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(PpsVirtualBoundariesPosY[n], pic_height_in_luma_samples − 1, yL + j) (8-1232)

– Otherwise, if pps_loop_filter_across_virtual_boundaries_disabled_flag is equal to 1, and PpsVirtualBoundariesPosY[n] % CtbSizeY is not equal to 0, and PpsVirtualBoundariesPosY[n] − yL is greater than 0 and less than 4 for any n = 0..pps_num_hor_virtual_boundaries − 1, the following applies:

v_{y+j} = Clip3(0, PpsVirtualBoundariesPosY[n] − 1, yL + j) (8-1233)

– Otherwise, the following applies:

v_{y+j} = Clip3(0, pic_height_in_luma_samples − 1, yL + j) (8-1234)

– The variables clipLeftPos, clipRightPos, clipTopPos and clipBottomPos are derived by invoking the ALF boundary position derivation process as specified in clause 8.8.5.5 with (xCtb, yCtb) and (xL – xCtb, yL – yCtb) as inputs.

– The vertical sample position offsets yM1, yP1 and yP2 are specified in Table 4-1. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	0
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

Table 4-2. Specification of yP1 according to the vertical luma sample position yL and applyAlfLineBufBoundary

Condition	yP1
(yL = = CtbSizeY − 4) && (applyAlfLineBufBoundary = = 1)	1
(yL = = CtbSizeY − 5) && (applyAlfLineBufBoundary = = 1)	0
Otherwise	1

– according to the vertical luma sample position yL, clipLeftPos and clipRightPos.
– The horizontal sample position offsets xM1 and xP1 are specified in Table y-yyyy according to the horizontal luma sample position xL, clipLeftPos and clipRightPos.
– The variable curr is derived as follows:

$$curr = alfPicture_C[xCtbC + x, yCtbC + y] \quad (8\text{-}1286)$$

– The array of cross component filter coefficients f[j] is derived as follows with j = 0..7:

$$f[j] = CcAlfCoeff[j] \quad (8\text{-}1287)$$

– The variable sum is derived as follows:

$$\begin{aligned} sum = {} & f[0] * recPicture_L[h_x, v_{y+yM1}] + \\ & f[1] * recPicture_L[h_{x+xM1}, v_y] + \\ & f[2] * recPicture_L[h_x, v_y] + \\ & f[3] * recPicture_L[h_{x+xP1}, v_y] + \quad (8\text{-}1289) \\ & f[4] * recPicture_L[h_{x+xM1}, v_{y+yP1}] + \\ & f[5] * recPicture_L[h_x, v_{y+yP1}] + \\ & f[6] * recPicture_L[h_{x+xP1}, v_{y+yP1}] + \\ & f[7] * recPicture_L[h_x, v_{y+yP2}] \end{aligned}$$

$$sum = Clip3(-(1 << (BitDepth_C - 1)), (1 << (BitDepth_C - 1)) - 1, sum) \quad (8\text{-}1290)$$

$$sum = curr + (sum + 64) >> (7 + (BitDepth_Y - BitDepth_C)) \quad (8\text{-}1290)$$

– The modified filtered reconstructed chroma picture sample array ccAlfPicture[xCtbC + x][yCtbC + y] is derived as follows:

$$ccAlfPicture[xCtbC + x][yCtbC + y] = Clip3(0, (1 << BitDepth_C) - 1, sum) \quad (8\text{-}1291)$$

Table x-xx – Specification of yM1, yP1 and yP2 according to the vertical luma sample position yL, clipTopPos and clipBottomPos

Delete the following table:

Condition	yM1	yP1	yP2
yL = = clipTopPos + 1	−1	1	1
yL = = clipTopPos	0	0	1
yL = = clipBottomPos − 1	0	0	1
yL = = clipBottomPos − 2	−1	1	1
Otherwise	−1	1	2

Add the following table:

Condition	yM1	yP1	yP2
yL = = clipTopPos + 1	-1	1	1
yL = = clipTopPos	0	0	0
yL = = clipBottomPos − 1	0	0	0
yL = = clipBottomPos − 2	-1	1	1
Otherwise	−1	1	2

Table y-yy – Specification of xM1 and xP1 according to the horizontal luma sample position xL, clipLeftPos and clipRightPos

Condition	xM1	xP1
xL = = clipLeftPos	0	0
xL = = clipRightPos − 1	0	0
xL = = clipRightPos – 2	–1	1
Otherwise	–1	1

[00169] FIG. 32 is a block diagram showing an example video processing system 1900 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1900. The system 1900 may include input 1902 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 1902 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.

[00170] The system 1900 may include a coding component 1904 that may implement the various coding or encoding methods described in the present document. The coding component 1904 may reduce the average bitrate of video from the input 1902 to the output of the coding component 1904 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 1904 may be either stored, or transmitted via a communication connected, as represented by the component 1906. The stored or communicated bitstream (or coded) representation of the video received at the input 1902 may be used by the component 1908 for generating pixel values or displayable video that is sent to a display interface 1910. 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.

[00171] 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.

[00172] FIG. 33 is a block diagram of a video processing apparatus 3600. The apparatus 3600 may be used to implement one or more of the methods described herein. The apparatus 3600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 3600 may include one or more processors 3602, one or more memories 3604 and video processing hardware 3606. The processor(s) 3602 may be configured to implement one or more methods described in the present document. The memory (memories) 3604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 3606 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00173] FIG. 35 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

[00174] As shown in FIG. 35, 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.

[00175] Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[00176] 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.

[00177] Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

[00178] 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.

[00179] 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.

[00180] FIG. 36 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG.35.

[00181] Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 36, 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.

[00182] 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.

[00183] 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.

[00184] 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. 36 separately for purposes of explanation.

[00185] 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.

[00186] 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.

[00187] 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.

[00188] 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.

[00189] 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.

[00190] 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.

[00191] In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

[00192] 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.

[00193] 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.

[00194] 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.

[00195] 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.

[00196] 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.

[00197] 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.

[00198] 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.

[00199] 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.

[00200] 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.

[00201] 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.

[00202] After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

[00203] 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.

[00204] FIG. 37 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG.35.

[00205] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 37, 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.

[00206] In the example of FIG. 37, 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. 36).

[00207] 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.

[00208] 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.

[00209] Motion compensation unit 302 may use interpolation filters as used by video encoder 20 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.

[00210] Motion compensation unit 302 may uses 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.

[00211] 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.

[00212] 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 predication and also produces decoded video for presentation on a display device.

[00213] Some embodiments may be described using the following clause-based format. The first set of clauses show example embodiments of techniques discussed in the previous section (e.g., item 1 of Example of Embodiments).

[00214] 1. A method of video processing (e.g., method 3400 of FIG. 34), comprising: determining (3402), for a conversion between a video unit of a video and a coded representation of the video, based on a criterion, to use a cross-component adaptive loop filter operation, wherein the cross-component adaptive loop filter uses a mirrored padding technique for unavailable luma samples; and performing (3404) the conversion based on the determining. The present document discloses various embodiments of a cross-component adaptive loop filter and its operation and mirrored padding techniques, and also their relationship with a virtual buffer boundary.

[00215] 2. The method of clause 1, wherein the mirrored padding technique is further used for deriving one or more corresponding luma samples of the unavailable luma sample.

[00216] 3. The method of clause 2, wherein the one or more corresponding luma samples are determined based on a distance of the one or more corresponding luma samples from a representative luma sample or a distance of the unavailable sample from the representative luma sample.

[00217] 4. The method of clause 3, wherein the representative luma sample corresponds to a position of a chroma sample for which the cross-component adaptive loop filtering technique is used.

[00218] 5. The method of clause 3, wherein a position of the representative luma sample depends on a color format of the video.

[00219] 6. The method of any of clauses 3-4, wherein the distance corresponds to a distance in a first direction between a pixel line along a second direction containing the one or more luma samples and a row containing the representative sample.

[00220] 7. The method of clause 6, wherein C represents a center line along the second direction where the representative luma sample is located, M represents a line along the second direction where the unavailable sample is located and N represents a line along the second direction in which the one ore more luma samples are located, where C, M and N are positive integers and M is not equal to N, then the mirrored padding technique is applied based on a size and shape of the cross-component adaptive loop filter.

[00221] 8. The method of clause 1, wherein the cross-component adaptive loop filter has a KxL filter shape, where K is an even number and L is positive integer, and wherein the mirrored padding technique comprises padding unavailable samples located at a line along a second direction M or N away from a virtual boundary of the cross-component adaptive loop filter are padded from the nearest sample line near the virtual boundary.

[00222] 9. The method of clause 3, wherein, in case that a virtual boundary of the cross-component adaptive loop filter is below in the second direction the representative luma sample, then padding the unavailable samples using a nearest line in the second direction above the virtual boundary.

[00223] 10. The method of clause 3, wherein, in case that an unavailable sample is located in row M, wherein M is an integer less than C, wherein C is an integer indicative of a center line along the second direction of the representative luma sample, then a sample located in line N in the second direction is determined to be the corresponding samples when d(C, M) = d(N, C) – offset, wherein offset is an integer value, or d(C, M) < d(N, C), where d( ) is a distance function.

[00224] The following clauses show example embodiments of techniques discussed in the previous section (e.g., item 2).

[00225] 11. The method of any of clauses 1-10, wherein the first direction is a vertical direction and the second direction is a horizontal direction.

[00226] 12. The method of any of clauses 1-10, wherein the first direction is a horizonal direction and the second direction is a vertical direction.

[00227] 13. The method of any of clauses 11-12, wherein an orientation of the first direction and the second direction depends on an orientation of a boundary of the virtual buffer.

[00228] The following clauses show example embodiments of techniques discussed in the previous section (e.g., item 3).

[00229] 14. The method of any of clauses 1-13, wherein the video unit comprises a video picture, a video subpicture, a video slice, a video tile or a 360-degree boundary of video.

[00230] 15. The method of any of clauses 1-14, wherein the performing the conversion comprising encoding the video to generate the coded representation.

[00231] 16. The method of any of clauses 1-14, wherein the performing the conversion comprises parsing and decoding the coded representation to generate the video.

[00232] In the above-disclosed clauses, the orientation may be horizontal or vertical and correspondingly the first direction and the second direction may be vertical or horizontal directions referred to by pixel columns and pixel rows.

[00233] 17. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 16.

[00234] 18. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 16.

[00235] 19. 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 claims 1 to 16.

[00236] 20. A method, apparatus or system described in the present document.

[00237] The second set of clauses show example embodiments of techniques discussed in the previous sections (e.g., items 1-3 of Examples of Embodiments).

[00238] 1. A method of video processing (e.g., method 3810 of FIG. 38A), comprising: determining 3812 for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit; and performing 3814 the conversion based on the determining.

[00239] 2. The method of clause 1, wherein the loop filtering tool includes a cross-component adaptive loop filtering (CC-ALF) tool.

[00240] 3. The method of clause 1, wherein the loop filtering tool includes an adaptive loop filtering (ALF) tool.

[00241] 4. The method of clause 1, wherein the mirrored padding process includes padding a second sample that is a corresponding sample of a first sample in a filter support region in the loop filtering tool even when the second sample is available, and wherein the first sample is unavailable and to be padded.

[00242] 5. The method of clause 1, wherein the mirrored padding process is further used for deriving the unavailable luma sample and one or more corresponding luma samples to the unavailable luma sample.

[00243] 6. The method of clause 5, wherein a corresponding luma sample is determined based on a distance of the corresponding luma sample from a representative luma sample and/or a distance of the unavailable sample from the representative luma sample.

[00244] 7. The method of clause 6, wherein the representative luma sample is defined as a collocated luma sample of a chroma sample to be filtered.

[00245] 8. The method of clause 7, wherein a position of the collocated luma sample of the chroma sample depends on a color format of the video.

[00246] 9. The method of clause 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (2x, 2y) in the color format being 4:2:0.

[00247] 10. The method of clause 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (2x, y) in the color format being 4:2:2.

[00248] 11. The method of clause 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (x, y) in the color format being 4:4:4.

[00249] 12. The method of clause 6, wherein the distance refers to a distance along a first direction between a first row containing the corresponding luma sample and a second row containing the representative luma sample.

[00250] 13. The method of clause 6, wherein the distance is calculated as a difference along a first direction between a pixel line along a second direction containing the corresponding luma sample and a row containing the representative luma sample.

[00251] 14. The method of clause 13, wherein C represents a center line along the second direction where the representative luma sample is located, M represents a line along the second direction where the unavailable sample is located and N represents a line along the second direction in which the corresponding luma sample is located, where C, M and N are positive integers and M is not equal to N.

[00252] 15. The method of clause 5, wherein the one or more corresponding luma samples to be padded in the mirrored padding process are based on a number of rows of samples utilized by a shape of a filter used by the CC-ALF tool.

[00253] 16. The method of clause 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(C, M) = d (N, C), whereby d(x, y) denotes a distance between row x and row y, and M, C, N are positive integers.

[00254] 17. The method of clause 6, wherein the mirrored padding process corresponds to one used during an application of an adaptive loop filtering (ALF) tool in a case that 1) a center luma sample is selected as the representative luma sample and 2) the CC-ALF tool has a K x L filter shape, whereby K is odd number and L is a positive integer.

[00255] 18. The method of clause 6, wherein, in a case that the CC-ALF tool has a K x L filter shape and the unavailable luma sample is located at row M above or row N below from a virtual boundary, the mirrored padding process includes padding the one or more corresponding luma samples located at row N or M away from the virtual boundary from a nearest sample row above the row N or below the row M, whereby K is odd number and, L, M, N are positive integers.

[00256] 19. The method of clause 18, wherein the unavailable luma sample is padded using a nearest row below the virtual boundary in a case that the virtual boundary is above the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row above a row containing the one or more corresponding luma samples in a case that the virtual boundary is above the representative luma sample.

[00257] 20. The method of clause 18, wherein the unavailable luma sample is padded using a nearest row above the virtual boundary in a case that the virtual boundary is below the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row below a row containing the one or more corresponding luma samples in a case that the virtual boundary is below the representative luma sample.

[00258] 21. The method of clause 19 or 20, wherein K=2, yL=0, yP1=1, and the virtual boundary is equal to CtbSizeY-4, whereby yL and yP1 are y-coordinates of two sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00259] 22. The method of clause 21, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yL is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

[00260] 23. The method of clause 19 or 20, wherein K=4, yM1=-1, yL=0, yP1=1, yP2=2, and the virtual boundary is equal to CtbSizeY-4, whereby yM1, yL, yP1, yP2 are y-coordinates of four sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00261] 24. The method of clause 23, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM1 is unavailable, a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

[00262] 25. The method of clause 23, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM1 is unavailable, a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL and samples (x, yP2) and (x, yP1) in rows at yP2 and yP1 are padded using a sample (x, yL) in the row at yL.

[00263] 26. The method of clause 19 or 20, wherein K=6, yM2=-2, yM1=-1, yL=0, yP1=1, yP2=2, yP3=3 and the virtual boundary is equal to CtbSizeY-4, whereby yM2, yM1, yL, yP1, yP2, yP3 are y-coordinates of six sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00264] 27. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 2 and a row above yM2 is unavailable, a sample (x, yP3) in a row at yP3 is padded using a sample (x, yP2) in a row at yP2.

[00265] 28. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM2 and a row at yM2 are unavailable, a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1 and samples (x, yP3) and (x, yP2) in rows at yP3 and yP2 are padded using a sample (x, yP1) in a row at yP1.

[00266] 29. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM2, a row at yM2 and a row at yM1 are unavailable, samples (x, yM2) and (x, yM1) in rows at yM2 and yM1 are padded using a sample (x, yL) at a row at yL and samples (x, yP3), (x, yP2), and (x, yP1) in rows yP3, yP2, and yP1 are padded using a sample (x, yL) in the row at yL.

[00267] 30. The method of clause 21, wherein in a case that yL is equal to CtbSizeY – 5 and a row at yP1 is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

[00268] 31. The method of clause 23, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP2 is unavailable, a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

[00269] 32. The method of clause 23, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP2 and yP1 are unavailable, a sample (x, yP1) in a row at yP1 and a sample (x, yP2) in

a row at yP2 are padded using a sample (x, yL) in a row at yL and a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL.

[00270] 33. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 7 and a row at yP3 is unavailable, a sample (x, yP3) in the row at yP3 is padded using a sample (x, yP2) in a row at yP2.

[00271] 34. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP3 and a row at yP2 are unavailable, a sample (x, yP3) in the row at yP3 and a sample (x, yP2) in the row at yP2 are padded using a sample (x, yP1) in a row at yP1 and a sample (x, yM2) is padded using a sample (x, yM1) in a row at yM1.

[00272] 35. The method of clause 26, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP3, yP2, and yP1 are unavailable, a sample (x, yP3) in a row at yP3, a sample (x, yP2) in a row at yP2, and a sample (x, yP1) in a row at yP1 are padded using a sample (x, yL) in a row at yL and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in the row at yL.

[00273] 36. The method of clause 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(C, M) = d (N, C) - offset or d(C, M) < d(N, C), whereby d(x, y) denotes a distance between row x and row y, offset is an integer, and M, C, N are positive integers.

[00274] 37. The method of clause 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(M, C) = d(C, N) – offset or d(C, M) < d(N, C), whereby d(x, y) denotes a distance between row x and row y, offset is an integer, and M, C, N are positive integers.

[00275] 38. The method of clause 36 or 37, wherein the offset is equal to 1.

[00276] 39. The method of clause 36, wherein, in a case that the unavailable luma sample is located at row M above or row N below from a virtual boundary, the mirrored padding process includes padding the one or more corresponding luma samples located at row N below the virtual boundary or row M above the virtual boundary from a nearest sample row above the row N or below the row M, whereby M and N are positive integers.

[00277] 40. The method of clause 39, wherein the unavailable luma sample is padded using a nearest row above the virtual boundary in a case that the virtual boundary is below the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row below a row containing the one or more corresponding luma samples.

[00278] 41. The method of clause 39, wherein the unavailable luma sample is padded using a nearest row below the virtual boundary in a case that the virtual boundary is above the representative luma sample and wherein the one or more corresponding samples are padded using a nearest row above a row containing the one or more corresponding luma samples.

[00279] 42. The method of clause 40 or 41, wherein K=2, yL=0, yP1=1, and the virtual boundary is equal to CtbSizeY-4, whereby yL and yP1 are y-coordinates of two sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00280] 43. The method of clause 42, wherein, in a case that yL is equal to CtbSizeY – 5 and a row at yP1 is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

[00281] 44. The method of clause 40 or 41, wherein K=4, yM1=-1, yL=0, yP1=1, yP2=2, and the virtual boundary is equal to CtbSizeY-4, whereby yM1, yL, yP1, yP2 are y-coordinates of four sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00282] 45. The method of clause 44, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM1 and a row at yM1 are unavailable, a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL and a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

[00283] 46. The method of clause 44, wherein in a case that that yL is equal to CtbSizeY – 6 and a row at yP2 is unavailable, a sample (x, yP2) in the row at yP2 is padded using a sample (x, yP1) in a row at yP1.

[00284] 47. The method of clause 44, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP2 and yP1 are unavailable, a sample (x, yP2) at row yP2 are padded using a sample (x, yL) in a row at yL and a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in the row at yL.

[00285] 48. The method of clause 40 or 41, wherein K=6, yM2=-2, yM1=-1, yL=0, yP1=1, yP2=2, yP3=3 and the virtual boundary is equal to CtbSizeY-4, whereby yM2, yM1, yL, yP1, yP2, yP3 are y-coordinates of six sample rows and CtbSizeY represents a size of coding tree unit (CTU).

[00286] 49. The method of clause 48, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM2 and a row at yM2 are unavailable, a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1 and a sample (x, yP3) in a row at yP3 is padded using a sample (x, yP2) in a row at yP2.

[00287] 50. The method of clause 48, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM2, a row at yM2 and a row at yM1 are unavailable, samples (x, yM2) and (x, yM1)

in rows at yM2 and yM1 are padded using a sample (x, yL) in a row at yL and samples (x, yP3) and (x, yP2) in rows yP3 and yP2 are padded using a sample (x, yP1) in a row at yP1.

[00288] 51. The method of clause 48, wherein, in a case that yL is equal to CtbSizeY – 7 and a row at yP3 is unavailable, a sample (x, yP3) in the row at yP3 is padded using a sample (x, yP2) in a row at yP2 and a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1.

[00289] 52. The method of clause 48, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP3 and a row at yP2 are unavailable, a sample (x, yP3) in the row at yP3 and a sample (x, yP2) in the row at yP2 are padded using a sample (x, yP1) in a row at yP1 and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in a row at yL.

[00290] 53. The method of clause 48, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP3, yP2, and yP1 are unavailable, a sample (x, yP3) in a row at yP3, a sample (x, yP2) in a row at yP2, and a sample (x, yP1) in a row at yP1 are padded using a sample (x, yL) in a row at yL and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in the row at yL.

[00291] 54. The method of clause 1, wherein a result of the determining is included in the bitstream representation at a sequence level, a picture level, a slice level, or a tile group level.

[00292] 55. The method of any of previous clauses, wherein the first direction is a vertical direction and the second direction is a horizontal direction.

[00293] 56. The method of any of previous clauses, wherein the first direction is a horizonal direction and the second direction is a vertical direction.

[00294] 57. The method of clause 55 or 56, wherein an orientation of the first direction and the second direction depends on an orientation of a boundary of a virtual buffer.

[00295] 58. A method of video processing (e.g., method 3820 of FIG. 38B), comprising: determining 3822, for a conversion between a video unit of a video and a bitstream representation of the video, whether to apply a repetitive padding process and/or a mirrored padding process for padding a sample located at a virtual boundary based on coded information of the video unit; and performing 3824 the conversion based on the determining.

[00296] 59. The method of clause 58, wherein the coded information includes a size of the video unit that is a coding tree unit (CTU) or coding tree block (CTB).

[00297] 60. The method of clause 59, wherein the mirrored padding process is applied in a case that the CTU or CTB size is greater than or equal to T, whereby T is a positive integer.

[00298] 61. The method of clause 59, wherein the repetitive padding process is applied in a case that the CTU or CTB size is smaller than or equal to T, whereby T is a positive integer.

[00299] 62. The method of any of previous clauses, wherein the video unit comprises a picture, a subpicture, a slice, a tile or a 360-degree boundary of the video.

[00300] 63. The method of any of previous clauses, wherein in the CC-ALF tool, sample values of the video unit of a video component are predicted from sample values of the video unit of another video component.

[00301] 64. The method of any of clauses 1 to 63, wherein the conversion includes encoding the video into the bitstream representation.

[00302] 65. The method of any of clauses 1 to 63, wherein the conversion includes decoding the video from the bitstream representation.

[00303] 66. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 65.

[00304] 67. A computer readable medium storing program code that, when executed, causes a processor to implement a method recited in any one or more of clauses 1 to 65.

[00305] 68. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

[00306] In the present document, the term "video processing" may refer to video encoding, video decoding, video compression or video decompression. For example, video compression algorithms may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa. The bitstream representation of a current video block may, for example, correspond to bits that are either co-located or spread in different places within the bitstream, as is defined by the syntax. For example, a macroblock may be encoded in terms of transformed and coded error residual values and also using bits in headers and other fields in the bitstream.

[00307] 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.

[00308] 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.

[00309] 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).

[00310] 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.

[00311] 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.

[00312] 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.

[00313] 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.

CLAIMS

1. A method of video processing, comprising:

determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit; and

performing the conversion based on the determining.

2. The method of claim 1, wherein the loop filtering tool includes a cross-component adaptive loop filtering (CC-ALF) tool.

3. The method of claim 1, wherein the loop filtering tool includes an adaptive loop filtering (ALF) tool.

4. The method of claim 1, wherein the mirrored padding process includes padding a second sample that is a corresponding sample of a first sample in a filter support region in the loop filtering tool even when the second sample is available, and wherein the first sample is unavailable and to be padded.

5. The method of claim 1, wherein the mirrored padding process is further used for deriving the unavailable luma sample and one or more corresponding luma samples to the unavailable luma sample.

6. The method of claim 5, wherein a corresponding luma sample is determined based on a distance of the corresponding luma sample from a representative luma sample and/or a distance of the unavailable sample from the representative luma sample.

7. The method of claim 6, wherein the representative luma sample is defined as a collocated luma sample of a chroma sample to be filtered.

8. The method of claim 7, wherein a position of the collocated luma sample of the chroma sample depends on a color format of the video.

9. The method of claim 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (2x, 2y) in the color format being 4:2:0.

10. The method of claim 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (2x, y) in the color format being 4:2:2.

11. The method of claim 8, wherein the collocated luma sample of the chroma sample located at (x, y) is defined as one located at (x, y) in the color format being 4:4:4.

12. The method of claim 6, wherein the distance refers to a distance along a first direction between a first row containing the corresponding luma sample and a second row containing the representative luma sample.

13. The method of claim 6, wherein the distance is calculated as a difference along a first direction between a pixel line along a second direction containing the corresponding luma sample and a row containing the representative luma sample.

14. The method of claim 13, wherein C represents a center line along the second direction where the representative luma sample is located, M represents a line along the second direction where the unavailable sample is located and N represents a line along the second direction in which the corresponding luma sample is located, where C, M and N are positive integers and M is not equal to N.

15. The method of claim 5, wherein the one or more corresponding luma samples to be padded in the mirrored padding process are based on a number of rows of samples utilized by a shape of a filter used by the CC-ALF tool.

16. The method of claim 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(C, M) = d (N, C), whereby d(x, y) denotes a distance between row x and row y, and M, C, N are positive integers.

17. The method of claim 6, wherein the mirrored padding process corresponds to one used during an application of an adaptive loop filtering (ALF) tool in a case that 1) a center luma sample is selected as the representative luma sample and 2) the CC-ALF tool has a K x L filter shape, whereby K is odd number and L is a positive integer.

18. The method of claim 6, wherein, in a case that the CC-ALF tool has a K x L filter shape and the unavailable luma sample is located at row M above or row N below from a virtual boundary, the mirrored padding process includes padding the one or more corresponding luma samples located at row N or M away from the virtual boundary from a nearest sample row above the row N or below the row M, whereby K is odd number and, L, M, N are positive integers.

19. The method of claim 18, wherein the unavailable luma sample is padded using a nearest row below the virtual boundary in a case that the virtual boundary is above the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row above a row containing the one or more corresponding luma samples in a case that the virtual boundary is above the representative luma sample.

20. The method of claim 18, wherein the unavailable luma sample is padded using a nearest row above the virtual boundary in a case that the virtual boundary is below the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row below a row containing the one or more corresponding luma samples in a case that the virtual boundary is below the representative luma sample.

21. The method of claim 19 or 20, wherein K=2, yL=0, yP1=1, and the virtual boundary is equal to CtbSizeY-4, whereby yL and yP1 are y-coordinates of two sample rows and CtbSizeY represents a size of coding tree unit (CTU).

22. The method of claim 21, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yL is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

23. The method of claim 19 or 20, wherein K=4, yM1=-1, yL=0, yP1=1, yP2=2, and the virtual boundary is equal to CtbSizeY-4, whereby yM1, yL, yP1, yP2 are y-coordinates of four sample rows and CtbSizeY represents a size of coding tree unit (CTU).

24. The method of claim 23, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM1 is unavailable, a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

25. The method of claim 23, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM1 is unavailable, a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL and samples (x, yP2) and (x, yP1) in rows at yP2 and yP1 are padded using a sample (x, yL) in the row at yL.

26. The method of claim 19 or 20, wherein K=6, yM2=-2, yM1=-1, yL=0, yP1=1, yP2=2, yP3=3 and the virtual boundary is equal to CtbSizeY-4, whereby yM2, yM1, yL, yP1, yP2, yP3 are y-coordinates of six sample rows and CtbSizeY represents a size of coding tree unit (CTU).

27. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 2 and a row above yM2 is unavailable, a sample (x, yP3) in a row at yP3 is padded using a sample (x, yP2) in a row at yP2.

28. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM2 and a row at yM2 are unavailable, a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1 and samples (x, yP3) and (x, yP2) in rows at yP3 and yP2 are padded using a sample (x, yP1) in a row at yP1.

29. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM2, a row at yM2 and a row at yM1 are unavailable, samples (x, yM2) and (x, yM1) in rows at yM2 and yM1 are padded using a sample (x, yL) at a row at yL and samples (x, yP3), (x, yP2), and (x, yP1) in rows yP3, yP2, and yP1 are padded using a sample (x, yL) in the row at yL.

30. The method of claim 21, wherein in a case that yL is equal to CtbSizeY – 5 and a row at yP1 is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

31. The method of claim 23, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP2 is unavailable, a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

32. The method of claim 23, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP2 and yP1 are unavailable, a sample (x, yP1) in a row at yP1 and a sample (x, yP2) in a row at yP2 are padded using a sample (x, yL) in a row at yL and a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL.

33. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 7 and a row at yP3 is unavailable, a sample (x, yP3) in the row at yP3 is padded using a sample (x, yP2) in a row at yP2.

34. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP3 and a row at yP2 are unavailable, a sample (x, yP3) in the row at yP3 and a sample (x, yP2) in the row at yP2 are padded using a sample (x, yP1) in a row at yP1 and a sample (x, yM2) is padded using a sample (x, yM1) in a row at yM1.

35. The method of claim 26, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP3, yP2, and yP1 are unavailable, a sample (x, yP3) in a row at yP3, a sample (x, yP2) in a row at yP2, and a sample (x, yP1) in a row at yP1 are padded using a sample (x, yL) in a row at yL and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in the row at yL.

36. The method of claim 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(C, M) = d (N, C) - offset or d(C, M) < d(N, C), whereby d(x, y) denotes a distance between row x and row y, offset is an integer, and M, C, N are positive integers.

37. The method of claim 5, wherein the unavailable luma sample is located at row M and samples located at row N are determined as the one or more corresponding luma samples to be padded, and wherein d(M, C) = d(C, N) – offset or d(C, M) < d(N, C), whereby d(x, y) denotes a distance between row x and row y, offset is an integer, and M, C, N are positive integers.

38. The method of claim 36 or 37, wherein the offset is equal to 1.

39. The method of claim 36, wherein, in a case that the unavailable luma sample is located at row M above or row N below from a virtual boundary, the mirrored padding process includes padding the one or more corresponding luma samples located at row N below the virtual boundary or row M above the virtual boundary from a nearest sample row above the row N or below the row M, whereby M and N are positive integers.

40. The method of claim 39, wherein the unavailable luma sample is padded using a nearest row above the virtual boundary in a case that the virtual boundary is below the representative luma sample and wherein the one or more corresponding luma samples are padded using a nearest row below a row containing the one or more corresponding luma samples.

41. The method of claim 39, wherein the unavailable luma sample is padded using a nearest row below the virtual boundary in a case that the virtual boundary is above the representative luma sample and wherein the one or more corresponding samples are padded using a nearest row above a row containing the one or more corresponding luma samples.

42. The method of claim 40 or 41, wherein K=2, yL=0, yP1=1, and the virtual boundary is equal to CtbSizeY-4, whereby yL and yP1 are y-coordinates of two sample rows and CtbSizeY represents a size of coding tree unit (CTU).

43. The method of claim 42, wherein, in a case that yL is equal to CtbSizeY – 5 and a row at yP1 is unavailable, a sample (x, yP1) in a row at yP1 is padded using a sample (x, yL) in a row at yL.

44. The method of claim 40 or 41, wherein K=4, yM1=-1, yL=0, yP1=1, yP2=2, and the virtual boundary is equal to CtbSizeY-4, whereby yM1, yL, yP1, yP2 are y-coordinates of four sample rows and CtbSizeY represents a size of coding tree unit (CTU).

45. The method of claim 44, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM1 and a row at yM1 are unavailable, a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in a row at yL and a sample (x, yP2) in a row at yP2 is padded using a sample (x, yP1) in a row at yP1.

46. The method of claim 44, wherein in a case that that yL is equal to CtbSizeY – 6 and a row at yP2 is unavailable, a sample (x, yP2) in the row at yP2 is padded using a sample (x, yP1) in a row at yP1.

47. The method of claim 44, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP2 and yP1 are unavailable, a sample (x, yP2) at row yP2 are padded using a sample (x, yL) in a row at yL and a sample (x, yM1) in a row at yM1 is padded using a sample (x, yL) in the row at yL.

48. The method of claim 40 or 41, wherein K=6, yM2=-2, yM1=-1, yL=0, yP1=1, yP2=2, yP3=3 and the virtual boundary is equal to CtbSizeY-4, whereby yM2, yM1, yL, yP1, yP2, yP3 are y-coordinates of six sample rows and CtbSizeY represents a size of coding tree unit (CTU).

49. The method of claim 48, wherein, in a case that yL is equal to CtbSizeY – 3 and a row above yM2 and a row at yM2 are unavailable, a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1 and a sample (x, yP3) in a row at yP3 is padded using a sample (x, yP2) in a row at yP2.

50. The method of claim 48, wherein, in a case that yL is equal to CtbSizeY – 4 and a row above yM2, a row at yM2 and a row at yM1 are unavailable, samples (x, yM2) and (x, yM1) in rows at yM2 and yM1 are padded using a sample (x, yL) in a row at yL and samples (x, yP3) and (x, yP2) in rows yP3 and yP2 are padded using a sample (x, yP1) in a row at yP1.

51. The method of claim 48, wherein, in a case that yL is equal to CtbSizeY – 7 and a row at yP3 is unavailable, a sample (x, yP3) in the row at yP3 is padded using a sample (x, yP2) in a row at yP2 and a sample (x, yM2) in a row at yM2 is padded using a sample (x, yM1) in a row at yM1.

52. The method of claim 48, wherein, in a case that yL is equal to CtbSizeY – 6 and a row at yP3 and a row at yP2 are unavailable, a sample (x, yP3) in the row at yP3 and a sample (x, yP2) in the row at yP2 are padded using a sample (x, yP1) in a row at yP1 and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in a row at yL.

53. The method of claim 48, wherein, in a case that yL is equal to CtbSizeY – 5 and rows at yP3, yP2, and yP1 are unavailable, a sample (x, yP3) in a row at yP3, a sample (x, yP2) in a row at yP2, and a sample (x, yP1) in a row at yP1 are padded using a sample (x, yL) in a row at yL and a sample (x, yM2) in a row at yM2 and a sample (x, yM1) in a row at yM1 are padded using a sample (x, yL) in the row at yL.

54. The method of claim 1, wherein a result of the determining is included in the bitstream representation at a sequence level, a picture level, a slice level, or a tile group level.

55. The method of any of claims 1-54, wherein the first direction is a vertical direction and the second direction is a horizontal direction.

56. The method of any of claims 1-54, wherein the first direction is a horizonal direction and the second direction is a vertical direction.

57. The method of claim 55 or 56, wherein an orientation of the first direction and the second direction depends on an orientation of a boundary of a virtual buffer.

58. A method of video processing, comprising:

determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to apply a repetitive padding process and/or a mirrored padding process for padding a sample located at a virtual boundary based on coded information of the video unit; and

performing the conversion based on the determining.

59. The method of claim 58, wherein the coded information includes a size of the video unit that is a coding tree unit (CTU) or coding tree block (CTB).

60. The method of claim 59, wherein the mirrored padding process is applied in a case that the CTU or CTB size is greater than or equal to T, whereby T is a positive integer.

61. The method of claim 59, wherein the repetitive padding process is applied in a case that the CTU or CTB size is smaller than or equal to T, whereby T is a positive integer.

62. The method of any of previous claims, wherein the video unit comprises a picture, a subpicture, a slice, a tile or a 360-degree boundary of the video.

63. The method of any of previous claims, wherein in the CC-ALF tool, sample values of the video unit of a video component are predicted from sample values of the video unit of another video component.

64. The method of any of claims 1 to 63, wherein the conversion includes encoding the video into the bitstream representation.

65. The method of any of claims 1 to 63, wherein the conversion includes decoding the video from the bitstream representation.

66. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of claims 1 to 65.

67. A computer readable medium storing program code that, when executed, causes a processor to implement a method recited in any one or more of claims 1 to 65.

68. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

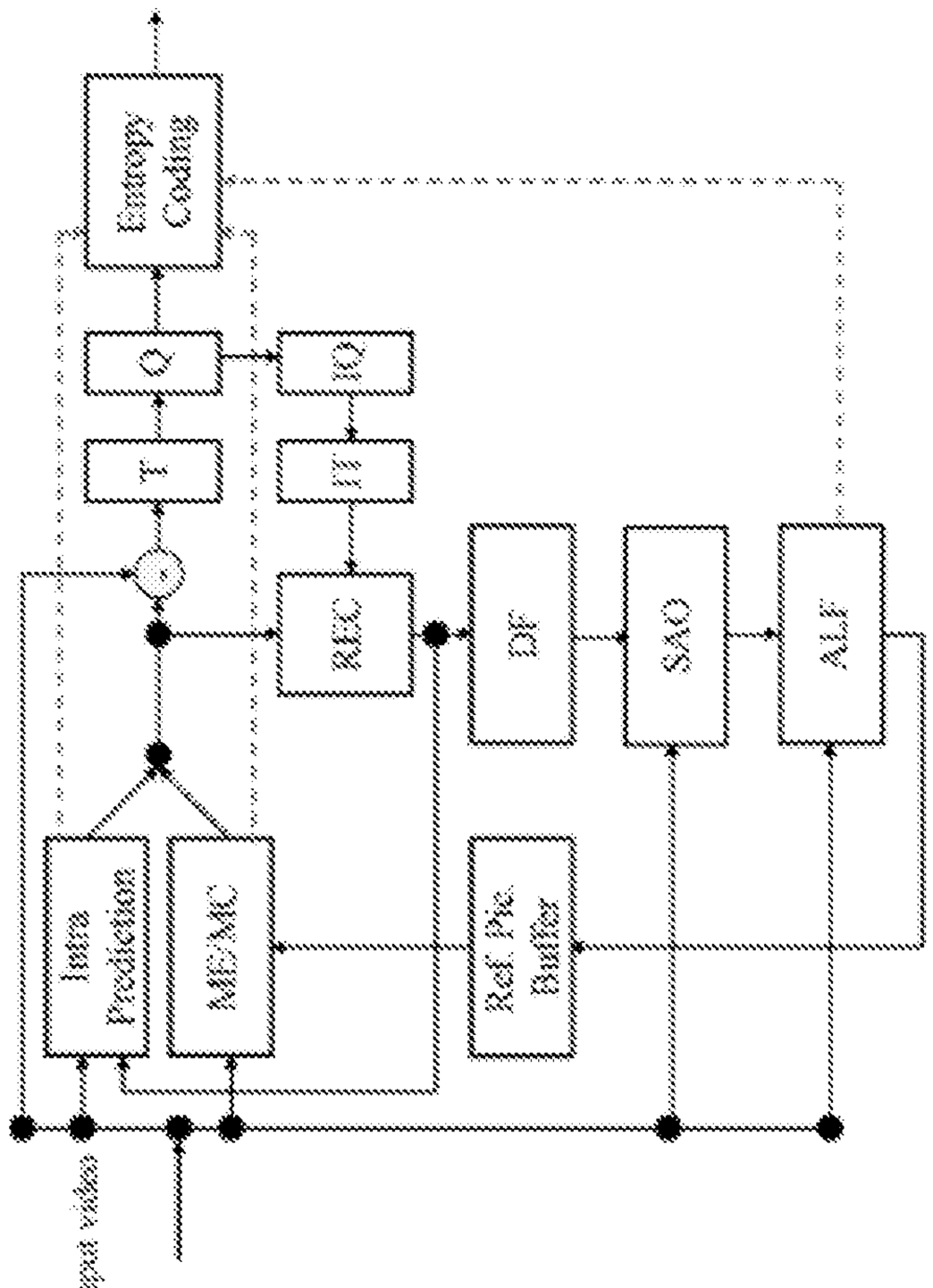

Entropy Coding
Q
IQ
T
IT
REC
DF
SAO
ALF
Intra Prediction
MC/MC
Ref. Pic. Buffer

FIG. 2

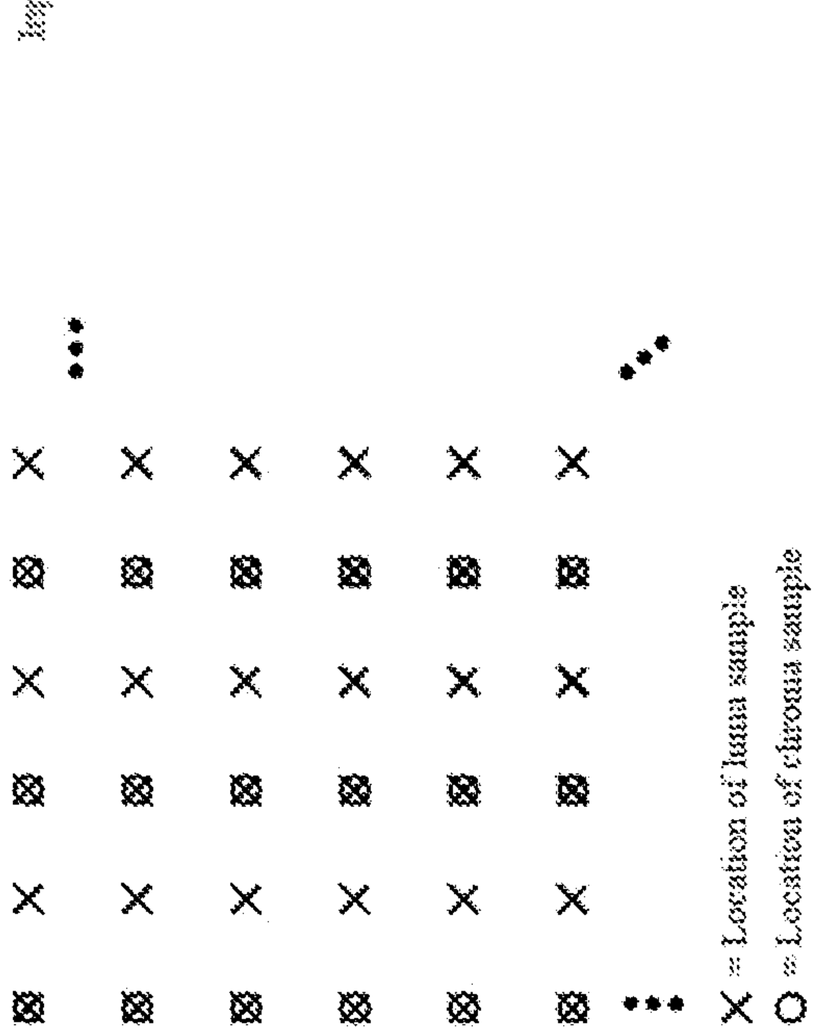

location of luma sample
location of chroma sample

FIG. 1

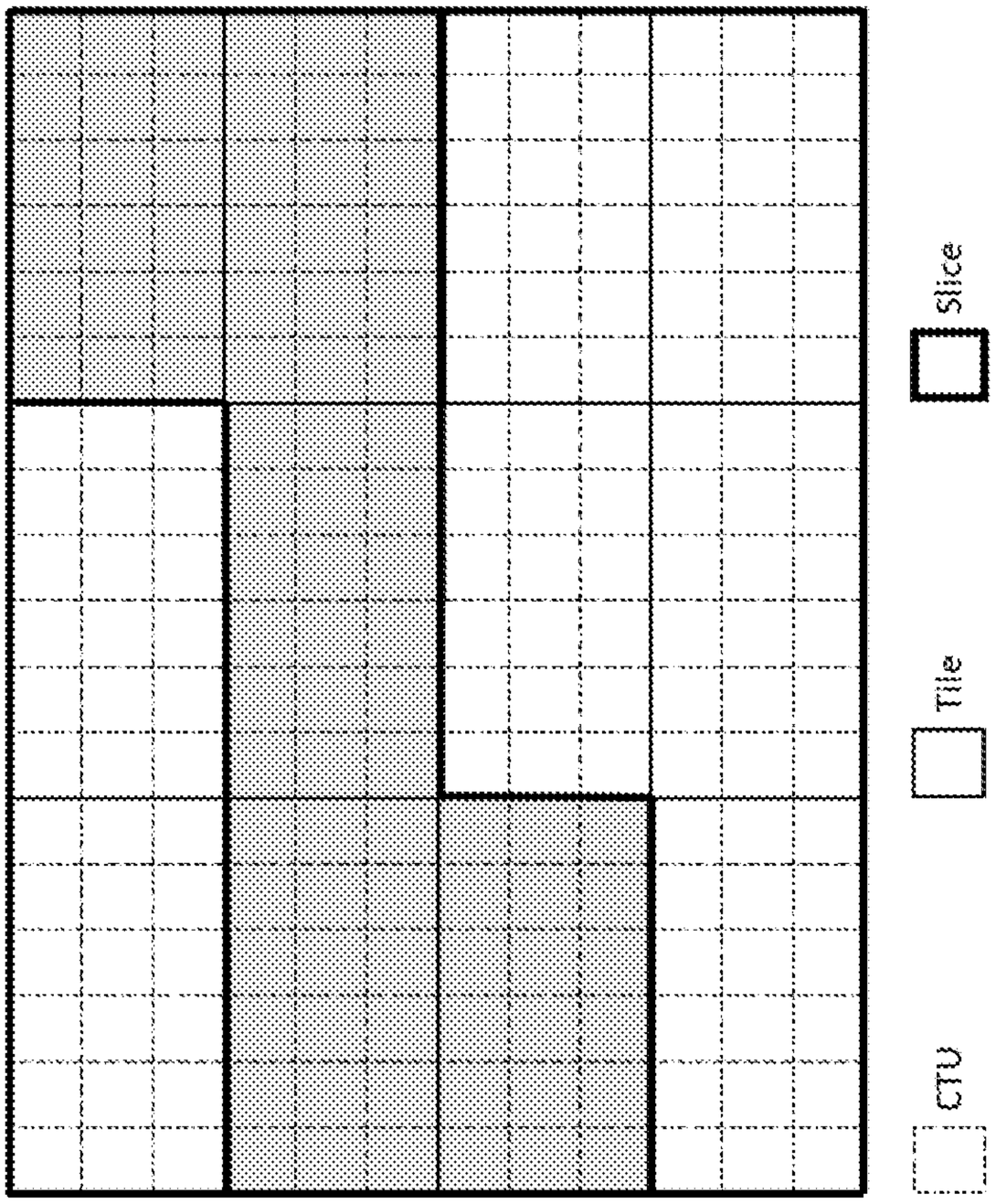


FIG. 3

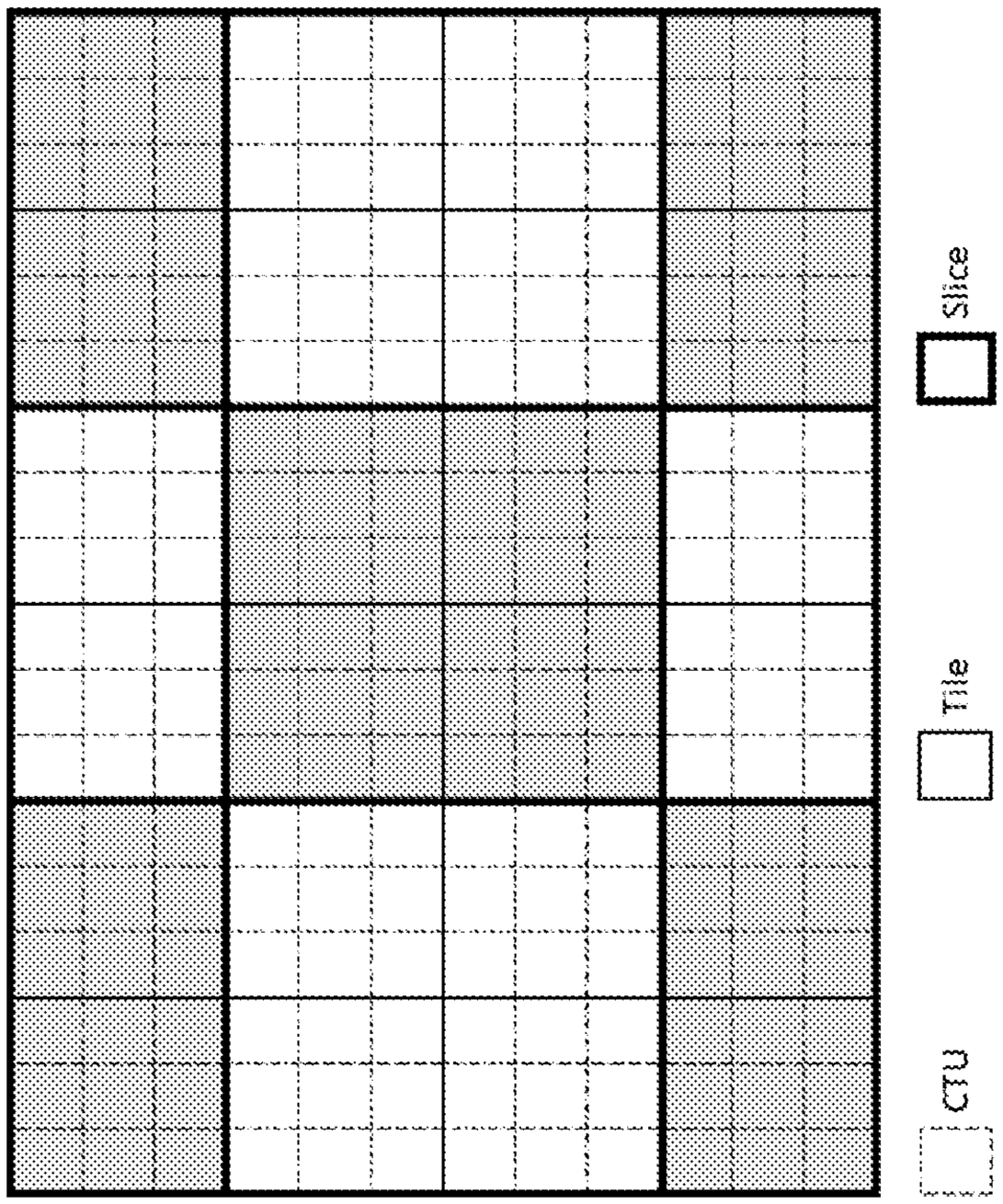


FIG. 4

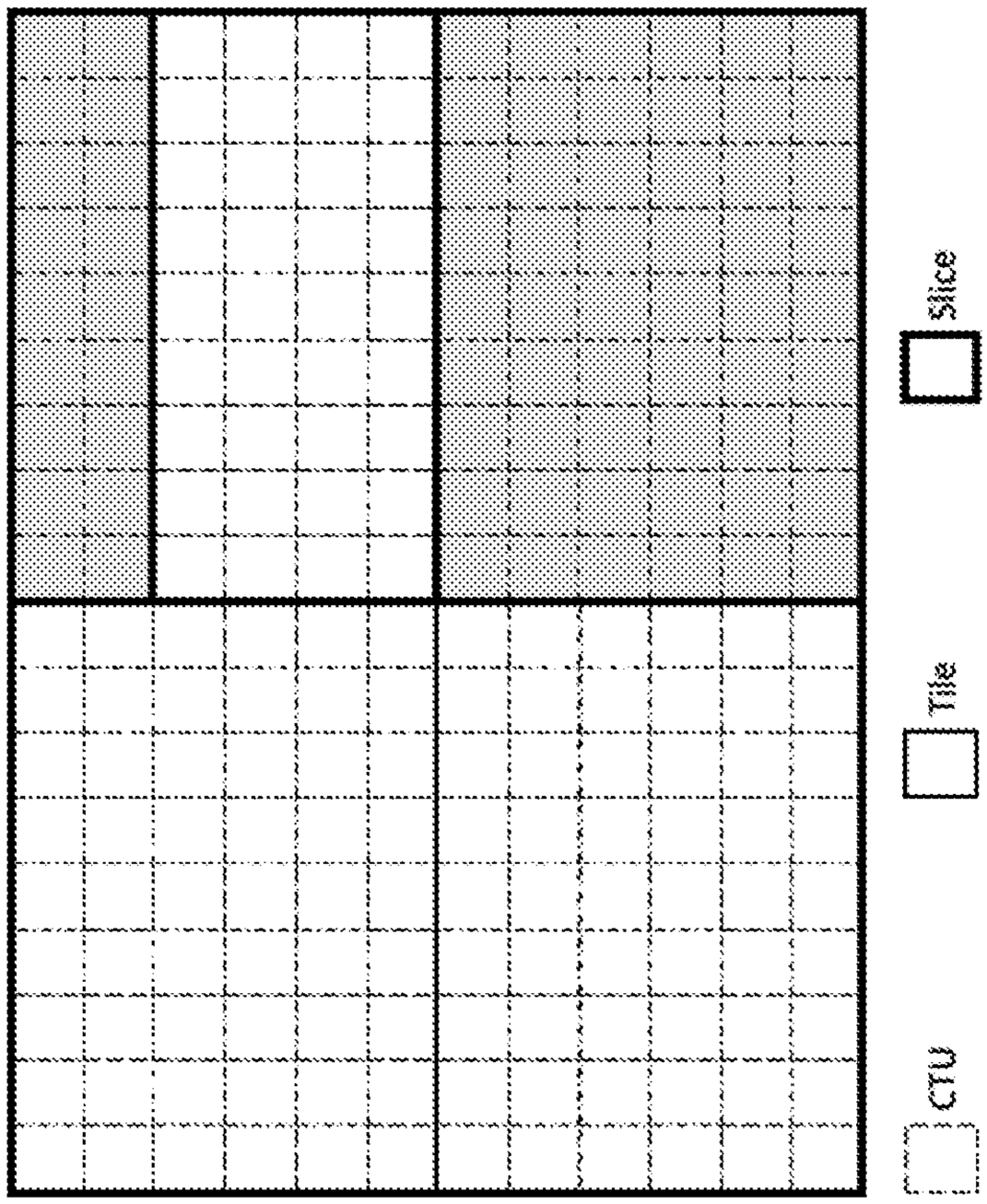


FIG. 5

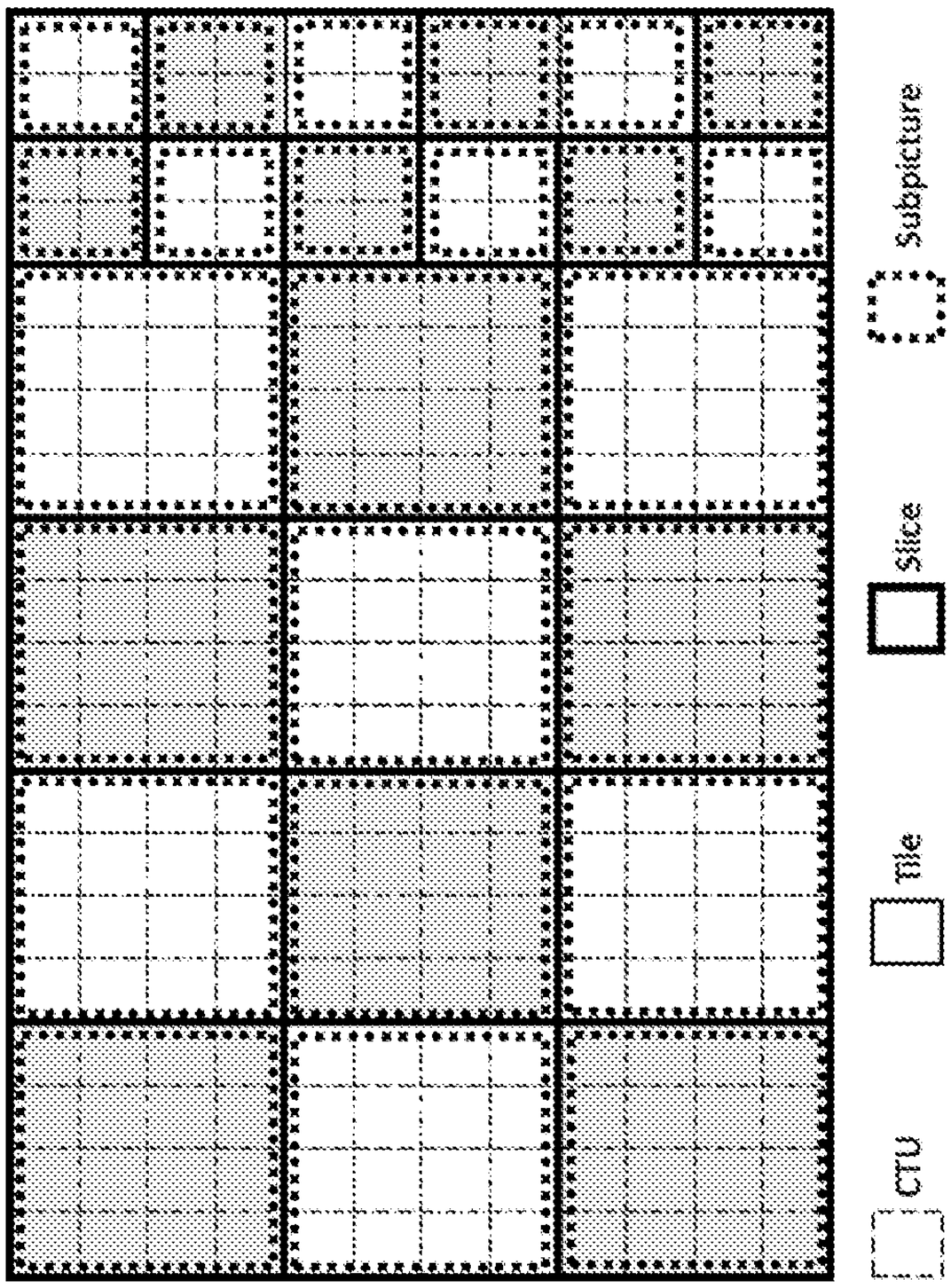


FIG. 6

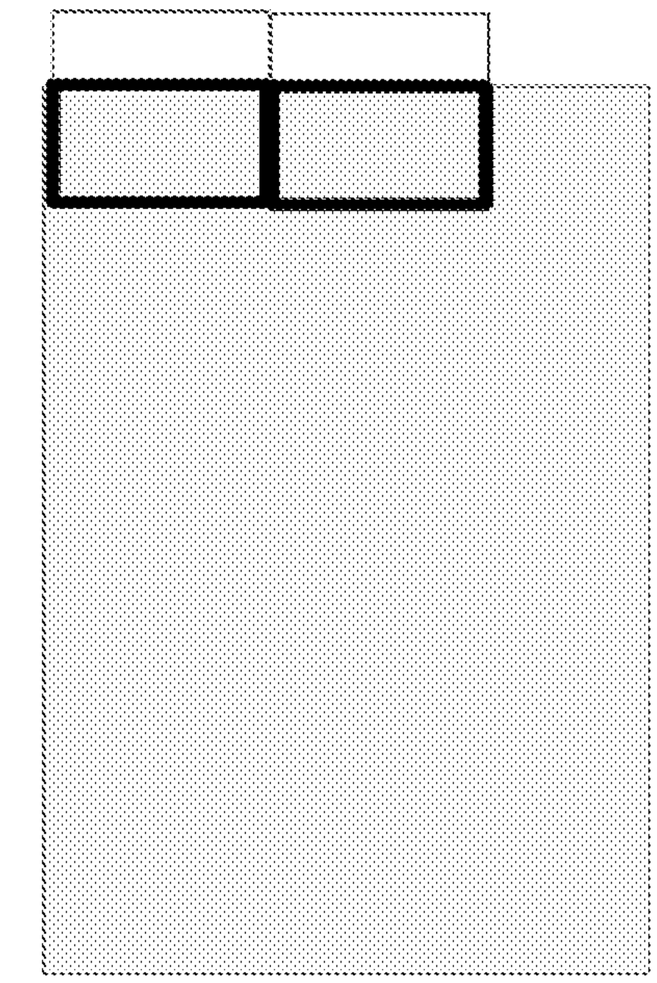


FIG. 7B

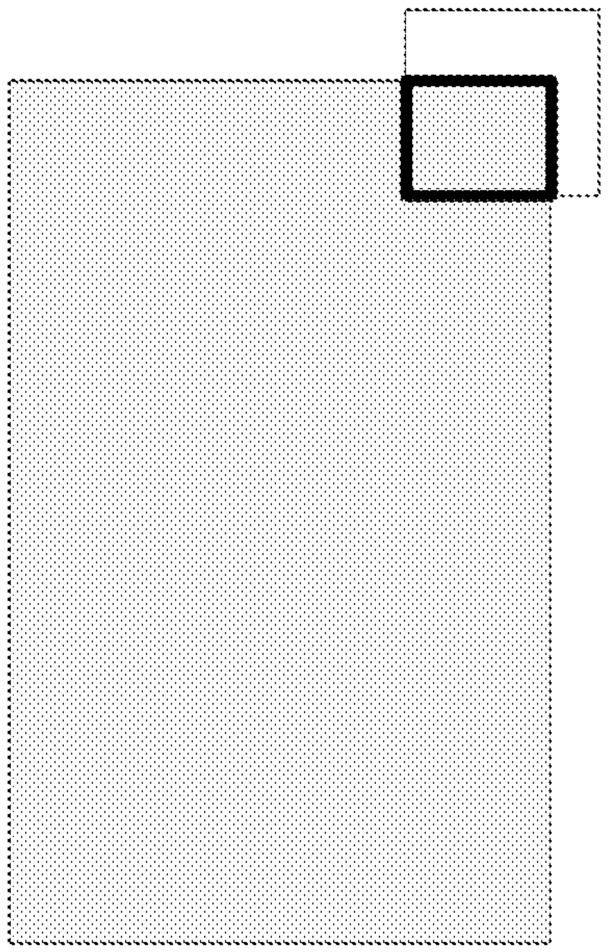


FIG. 7C

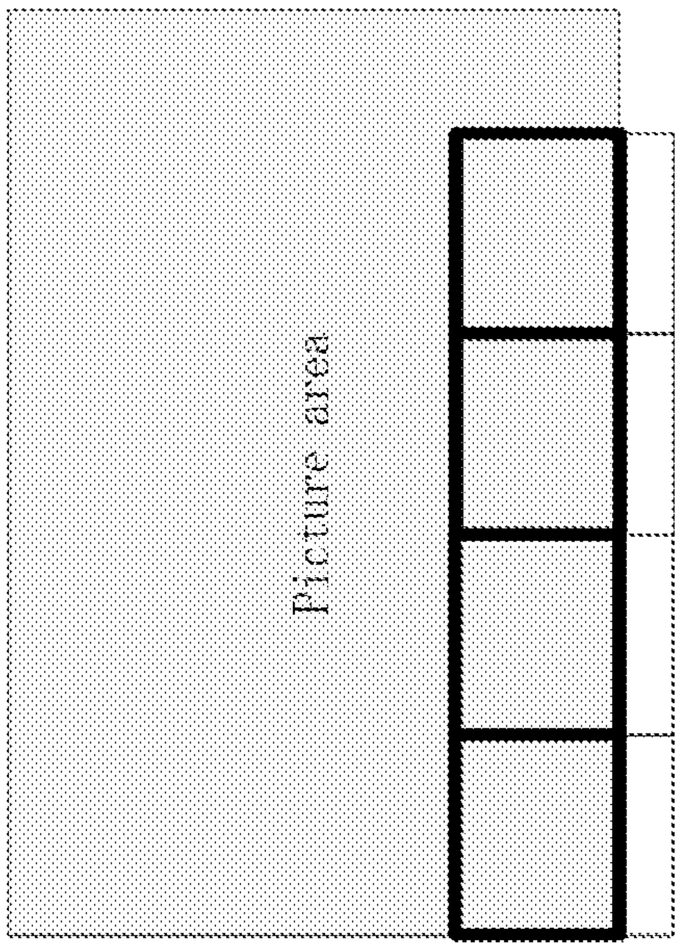


FIG. 7A

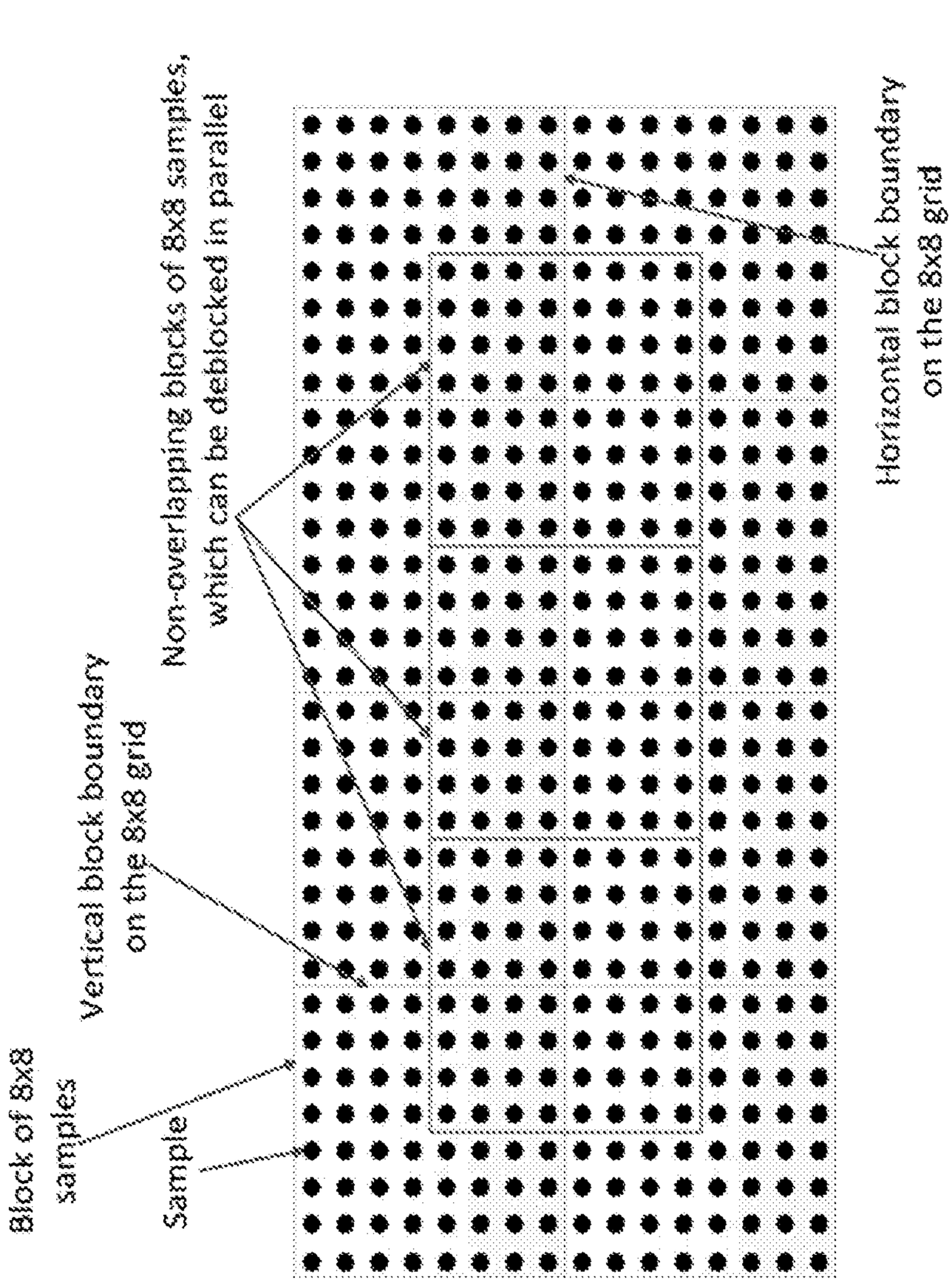

Non-overlapping blocks of 8x8 samples, which can be deblocked in parallel
Horizontal block boundary on the 8x8 grid
Vertical block boundary on the 8x8 grid
Block of 8x8 samples
Sample

FIG. 8

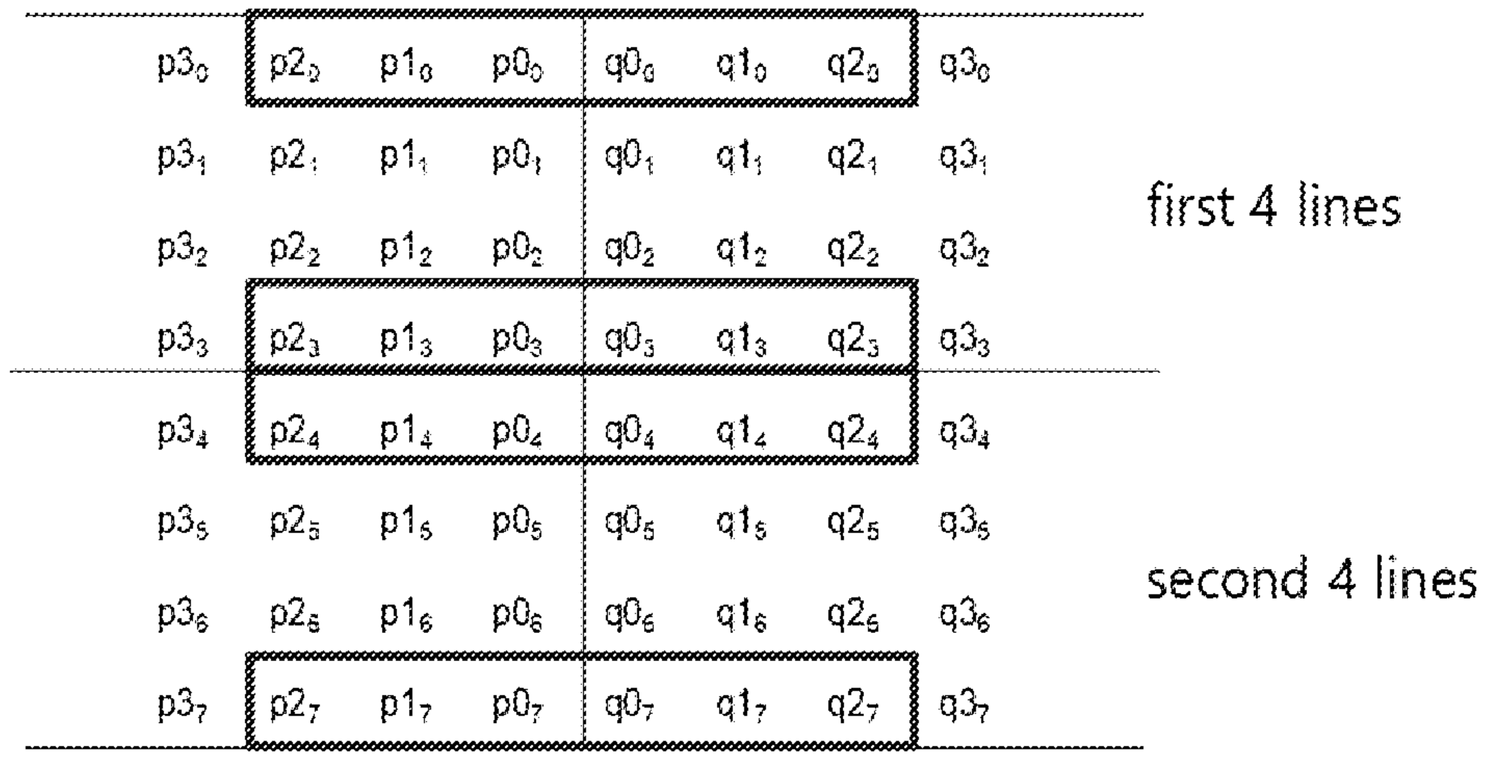

p3₀ p2₀ p1₀ p0₀ q0₀ q1₀ q2₀ q3₀
p3₁ p2₁ p1₁ p0₁ q0₁ q1₁ q2₁ q3₁
p3₂ p2₂ p1₂ p0₂ q0₂ q1₂ q2₂ q3₂
p3₃ p2₃ p1₃ p0₃ q0₃ q1₃ q2₃ q3₃
p3₄ p2₄ p1₄ p0₄ q0₄ q1₄ q2₄ q3₄
p3₅ p2₅ p1₅ p0₅ q0₅ q1₅ q2₅ q3₅
p3₆ p2₆ p1₆ p0₆ q0₆ q1₆ q2₆ q3₆
p3₇ p2₇ p1₇ p0₇ q0₇ q1₇ q2₇ q3₇
first 4 lines
second 4 lines

FIG. 9

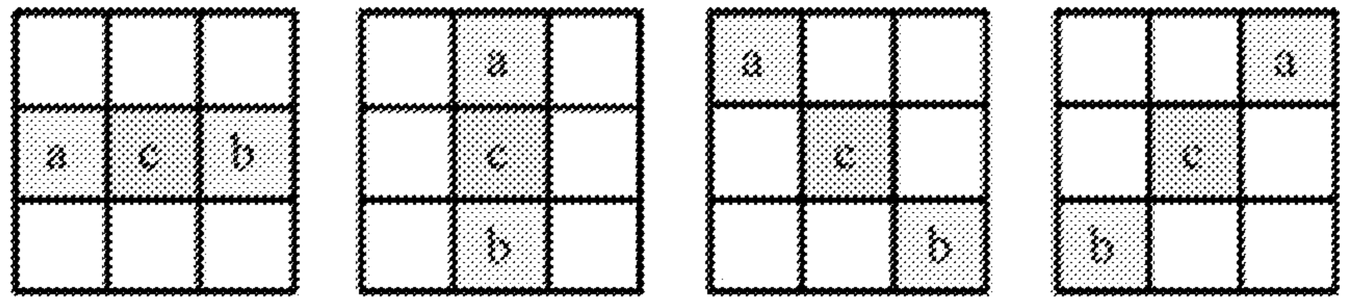

a
c
b

FIG. 10

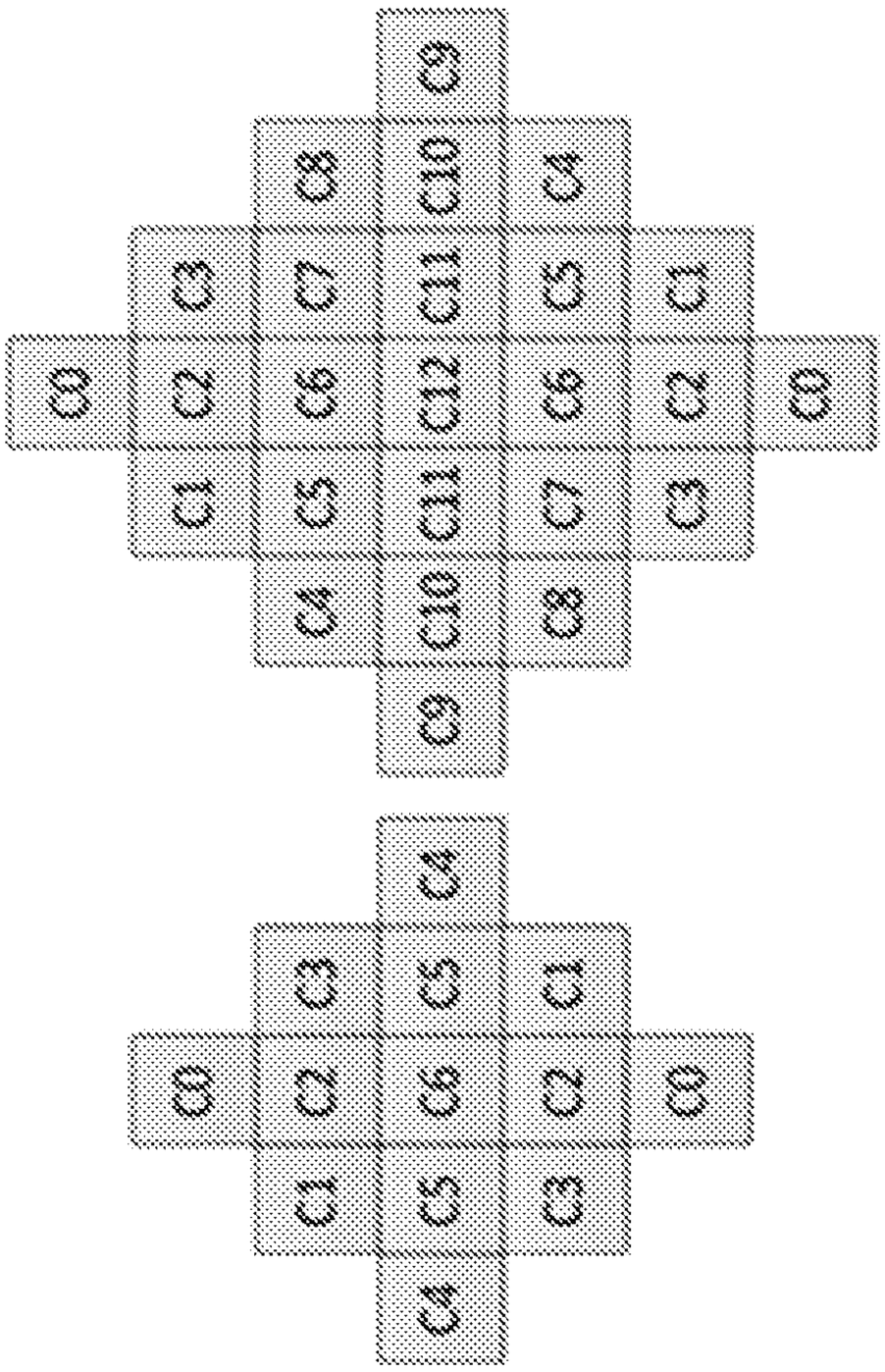

C9
C8 C10 C4
C3 C7 C11 C5 C1
C9 C2 C6 C12 C8 C2 C9
C1 C5 C11 C7 C3
C4 C10 C8
C9
C4
C3 C5 C1
C9 C2 C9 C2 C9
C3 C5 C3
C4

FIG. 11

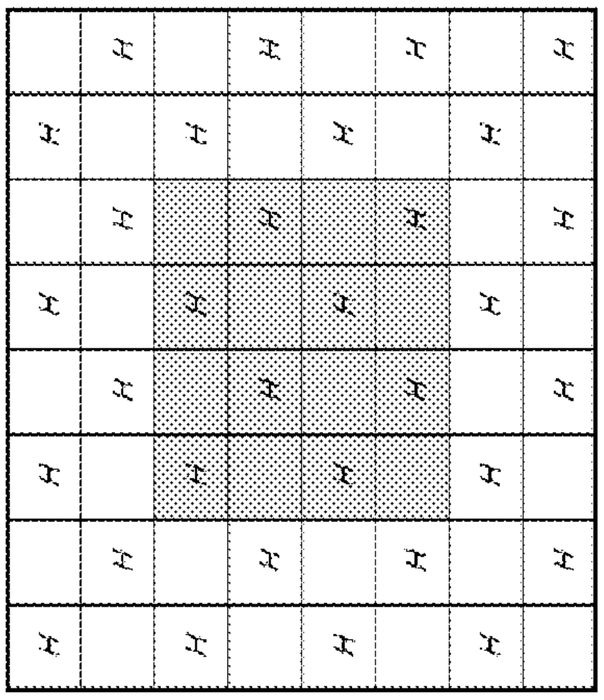


FIG. 12B

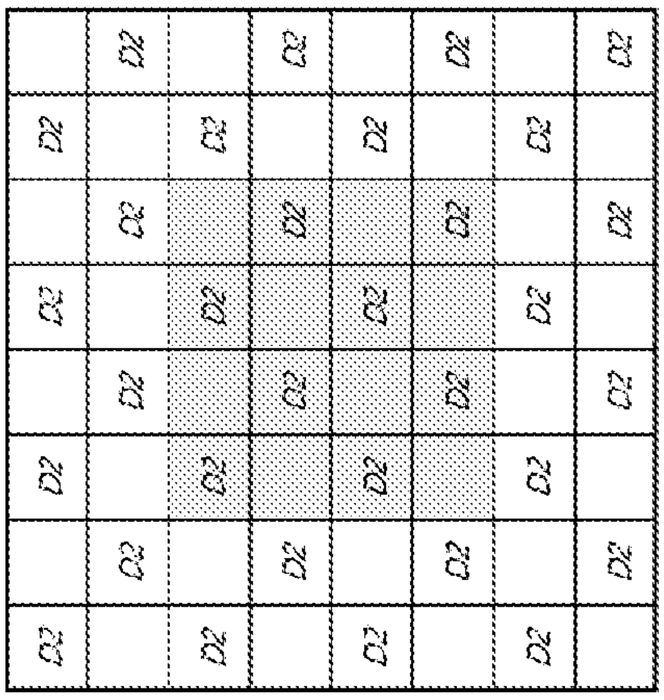


FIG. 12D

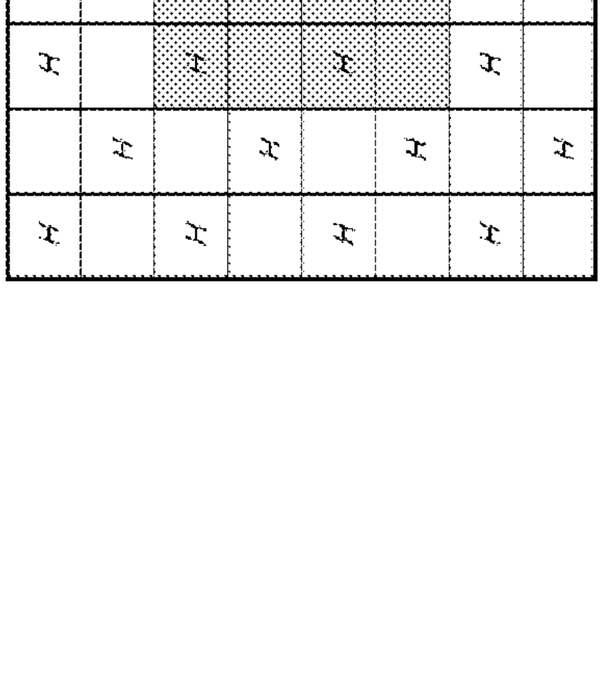


FIG. 12A

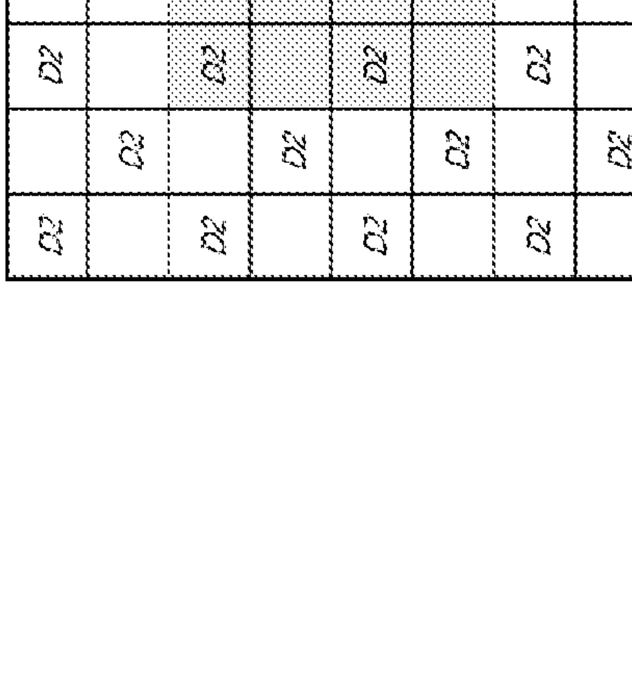


FIG. 12C

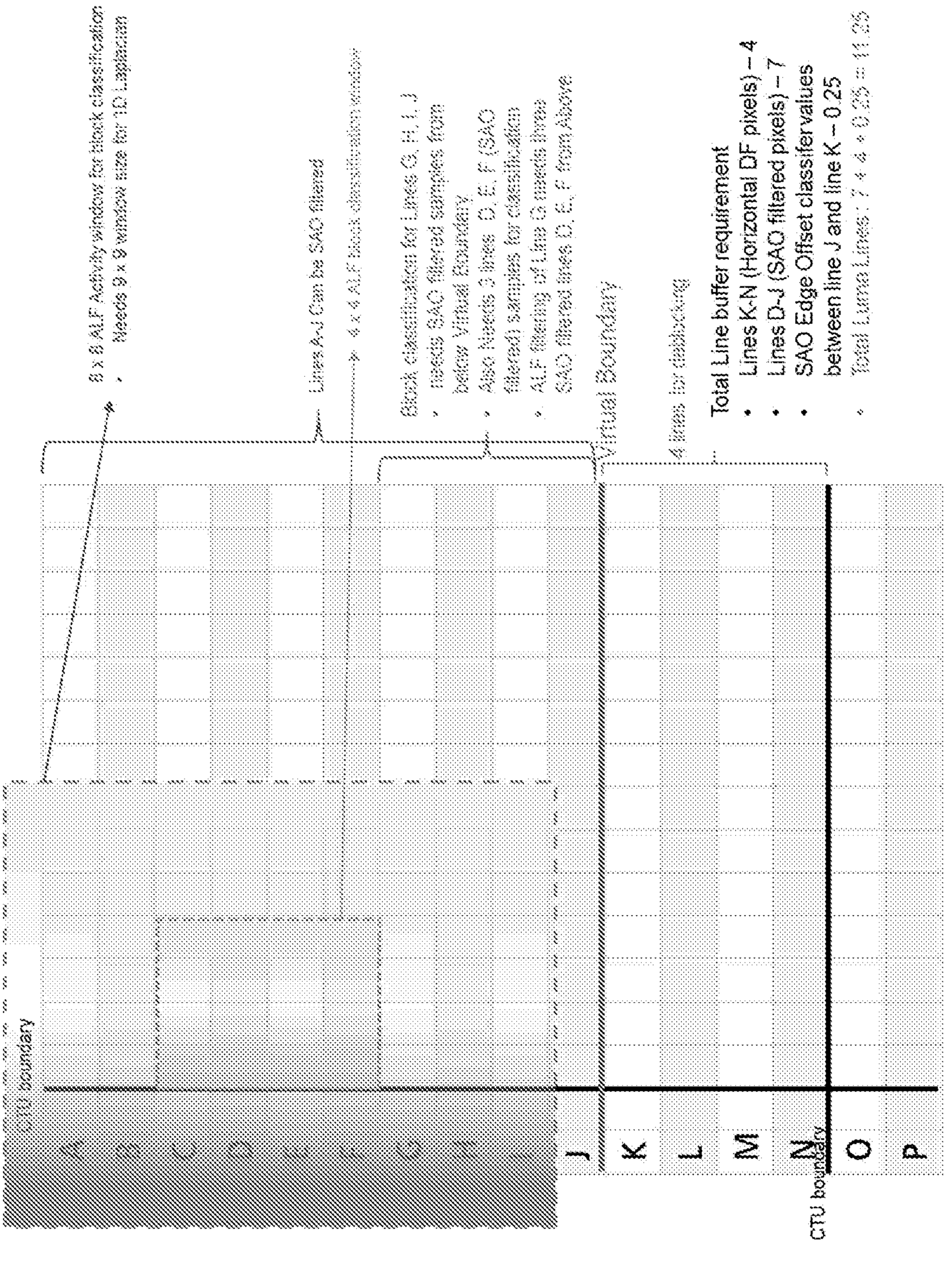

8 x 8 ALF Activity window for block classification
Needs 9 x 9 window size for 1D Laplacian
Lines A-J Can be SAO filtered
4 x 4 ALF block classification window
Block classification for Lines G, H, I, J
needs SAO filtered samples from below Virtual Boundary
Also Needs 3 lines D, E, F (SAO filtered) samples for classification
ALF filtering of Line G needs three SAO filtered lines D, E, F from Above
Virtual Boundary
4 lines for deblocking
Total Line buffer requirement
Lines K-N (Horizontal DF pixels) – 4
Lines D-J (SAO filtered pixels) – 7
SAO Edge Offset classifer values between line J and line K – 0.25
Total Luma Lines: 7 + 4 + 0.25 = 11.25
CTU boundary
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
CTU boundary

FIG. 13

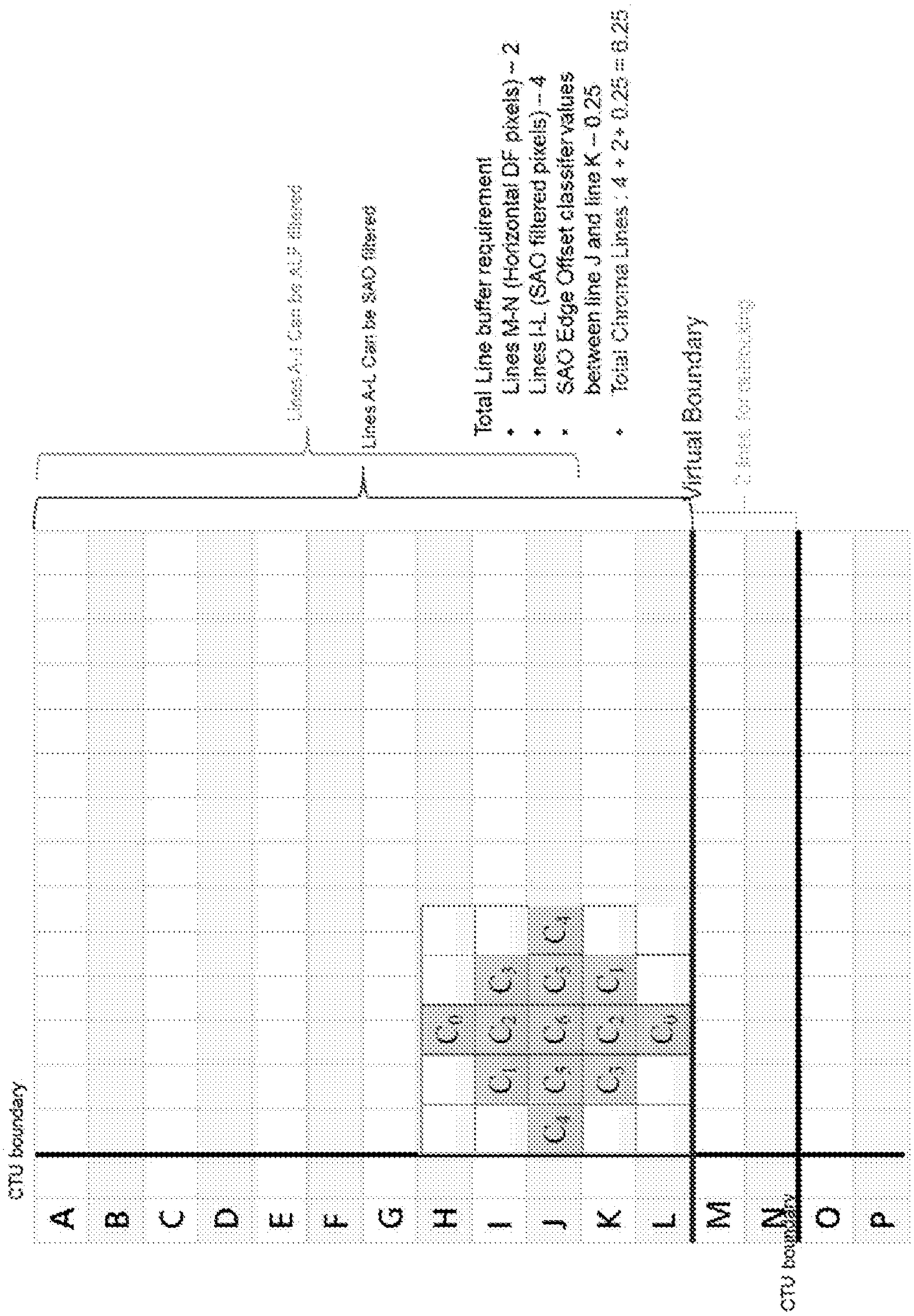

Lines A-L Can be ALF filtered
Lines A-L Can be SAO filtered
Total Line buffer requirement
Lines M-N (Horizontal DF pixels) – 4
Lines H-L (SAO filtered pixels) – 4
SAO Edge Offset classifier values between line J and line K – 0.25
Total Chroma Lines : 4 + 2 + 0.25 = 6.25
Virtual Boundary
CTU boundary
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

FIG. 14

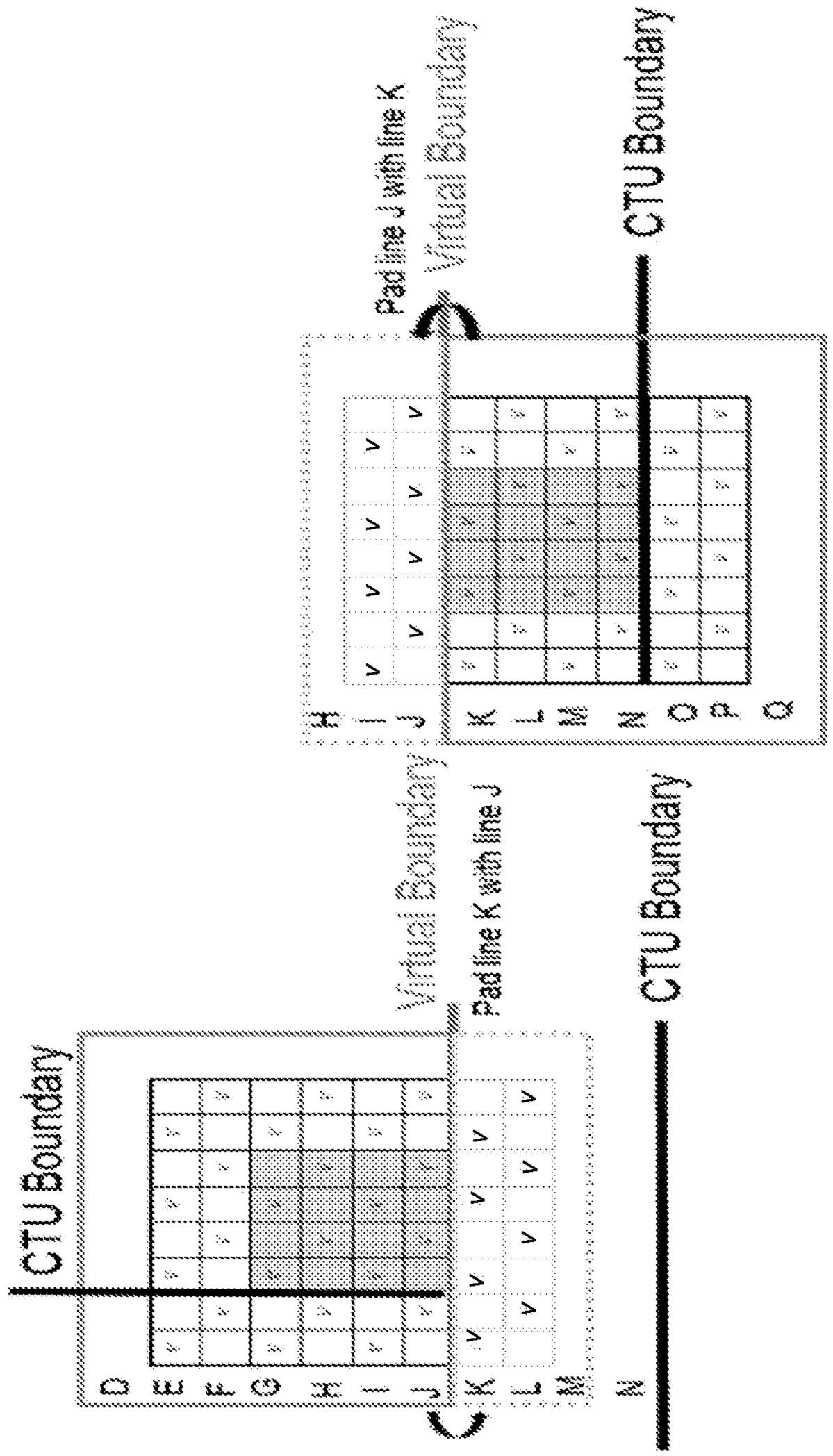

Pad line J with line K
Virtual Boundary
CTU Boundary
CTU Boundary
Virtual Boundary
Pad line K with line J
CTU Boundary

FIG. 15

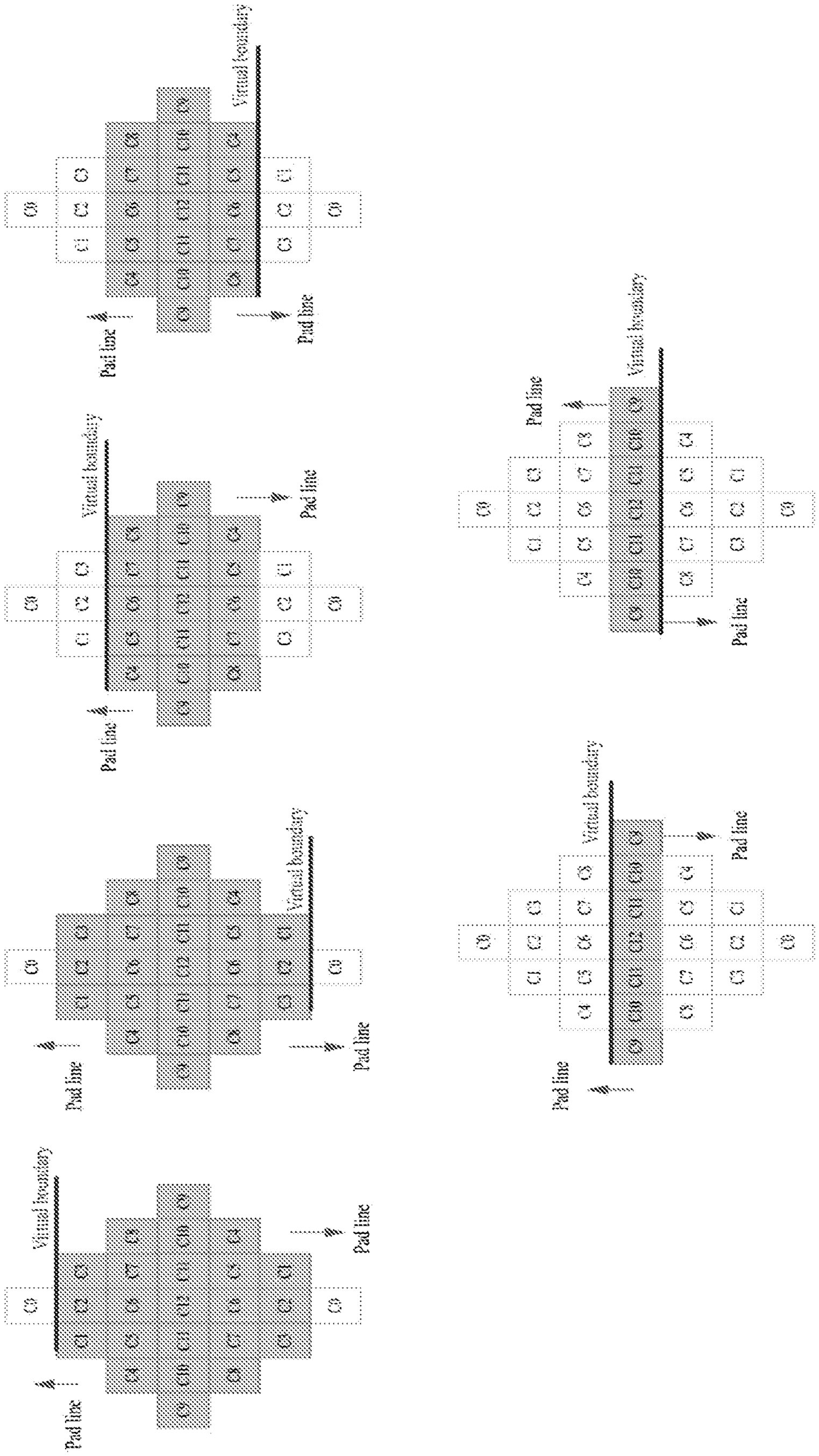

Virtual boundary
Pad line
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12

FIG. 16

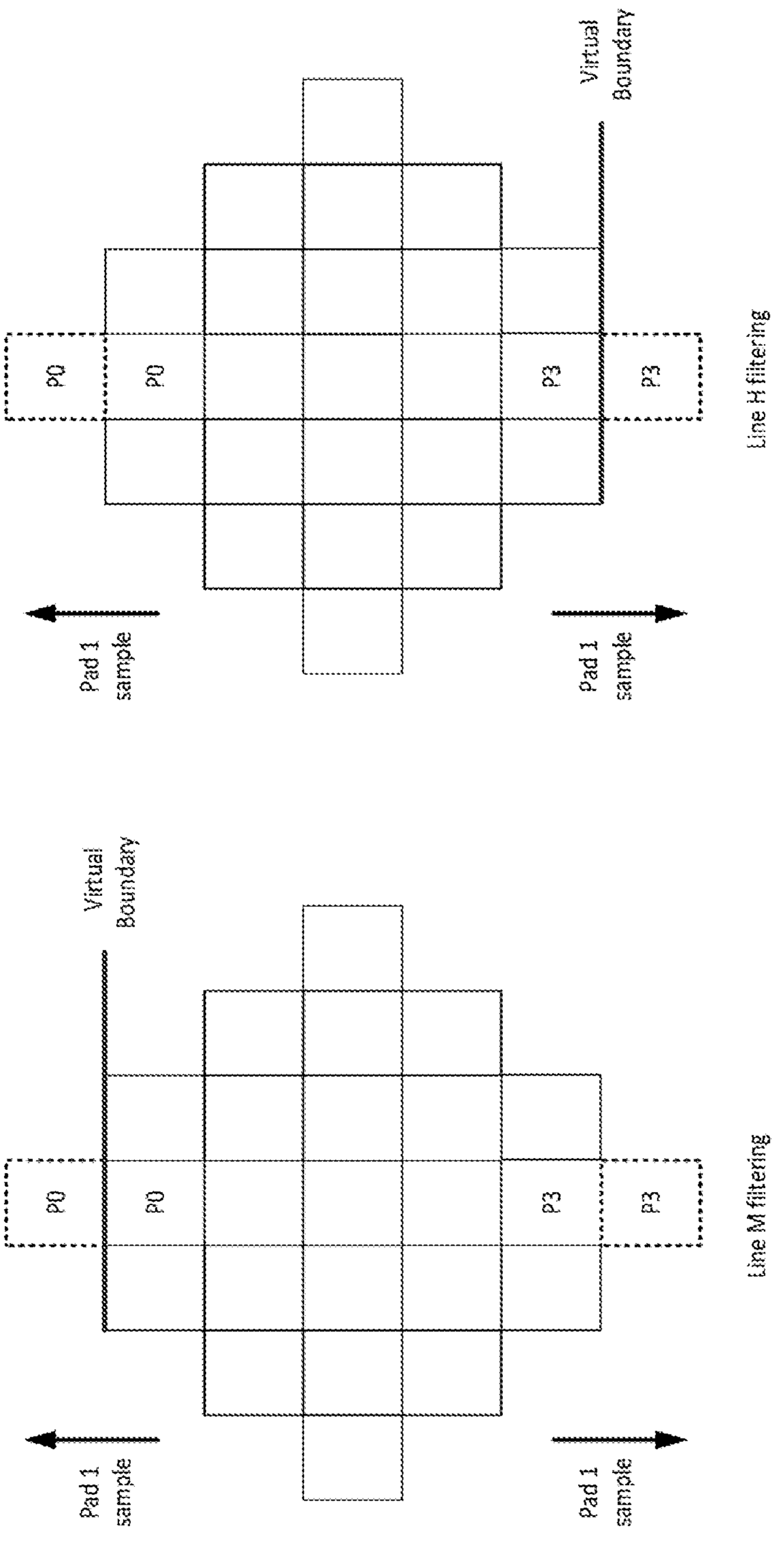

Virtual Boundary
P0
P0
P3
P3
Pad 1 sample
Pad 1 sample
Line H filtering
Virtual Boundary
P0
P0
P3
P3
Pad 1 sample
Pad 1 sample
Line M filtering

FIG. 17A

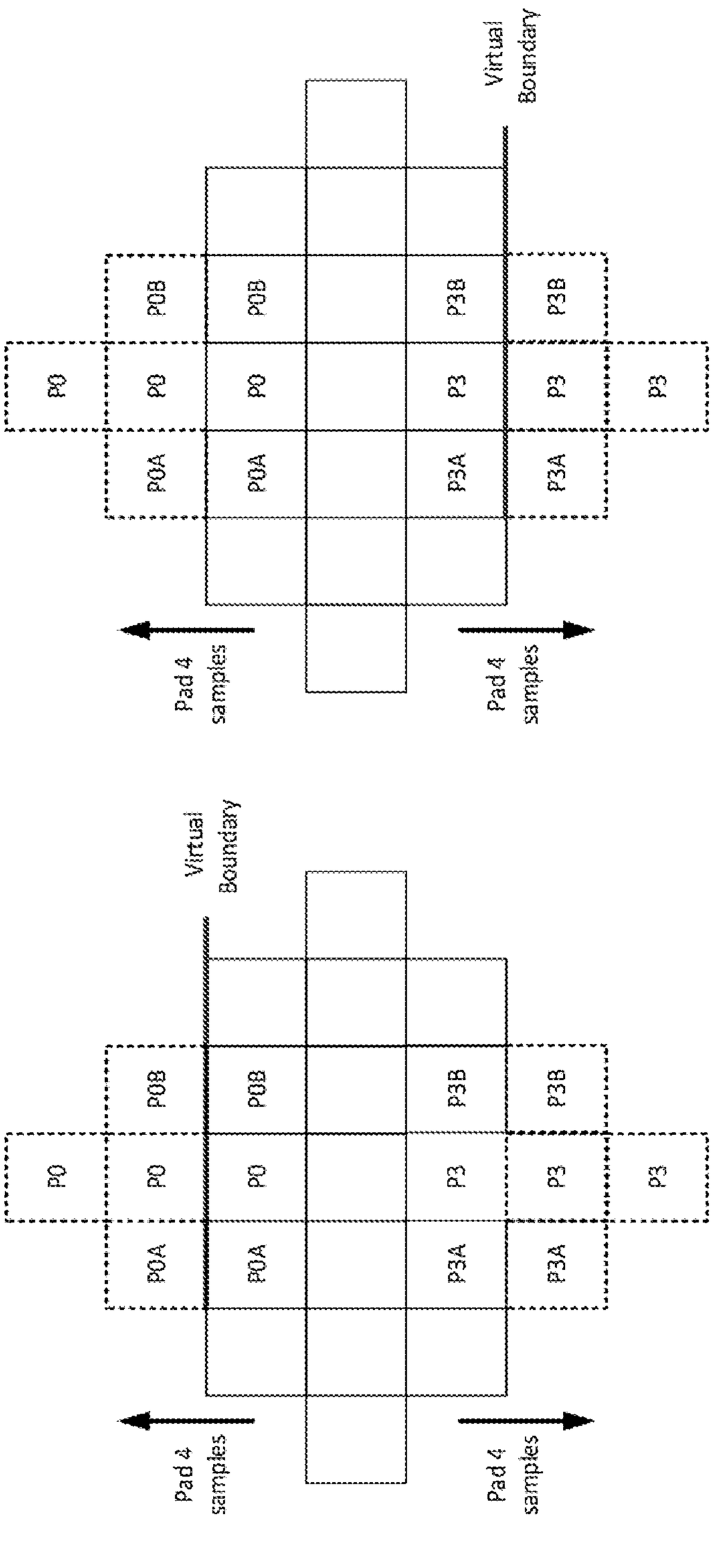

Virtual Boundary
P0
P0A
P0B
P3
P3A
P3B
Pad 4 samples
Pad 4 samples
Line L filtering

FIG. 17B

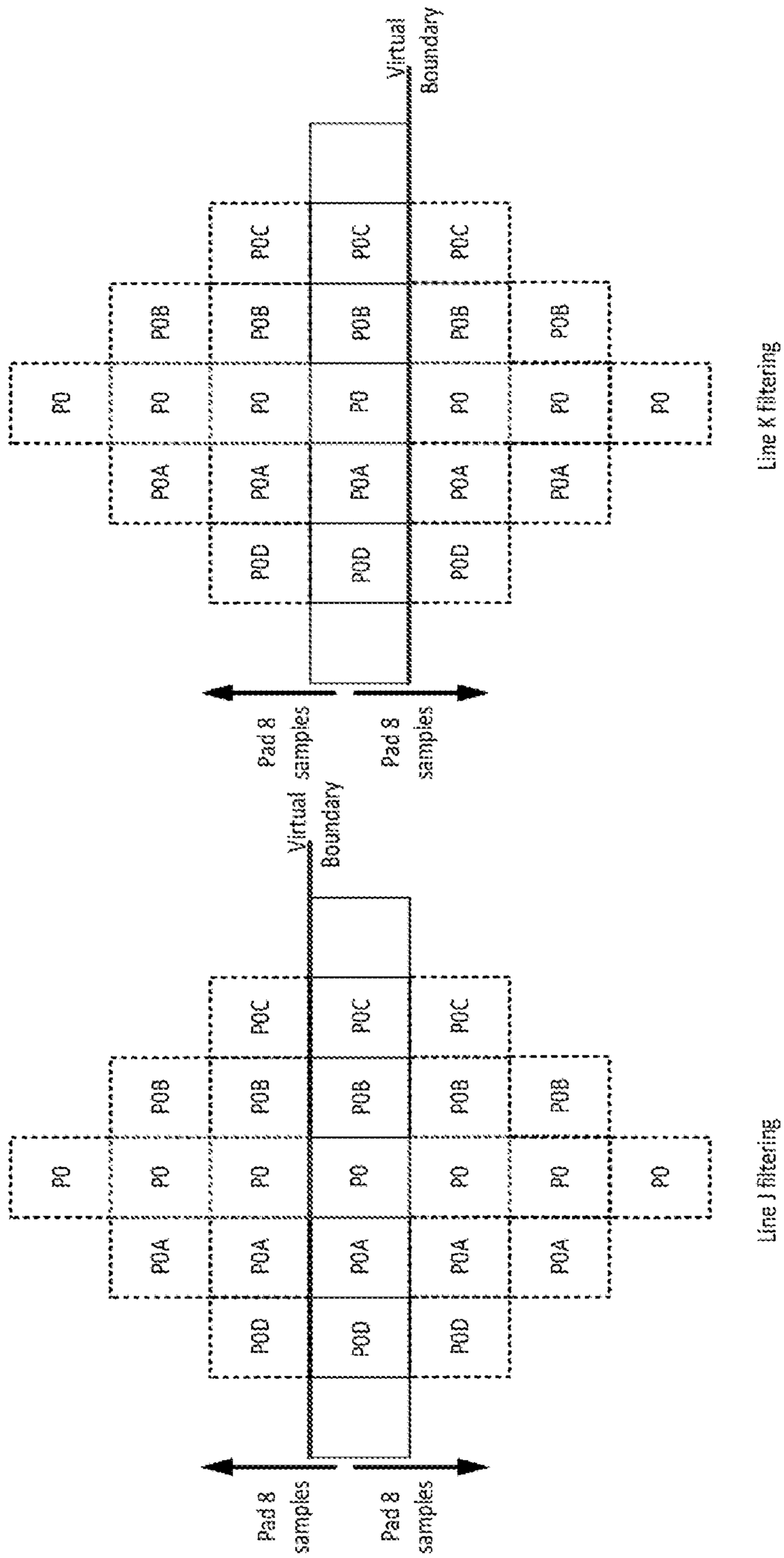

Virtual Boundary
P0
P0A
P0B
P0C
P0D
Pad 8 samples
Line K filtering
Line J filtering

FIG. 17C

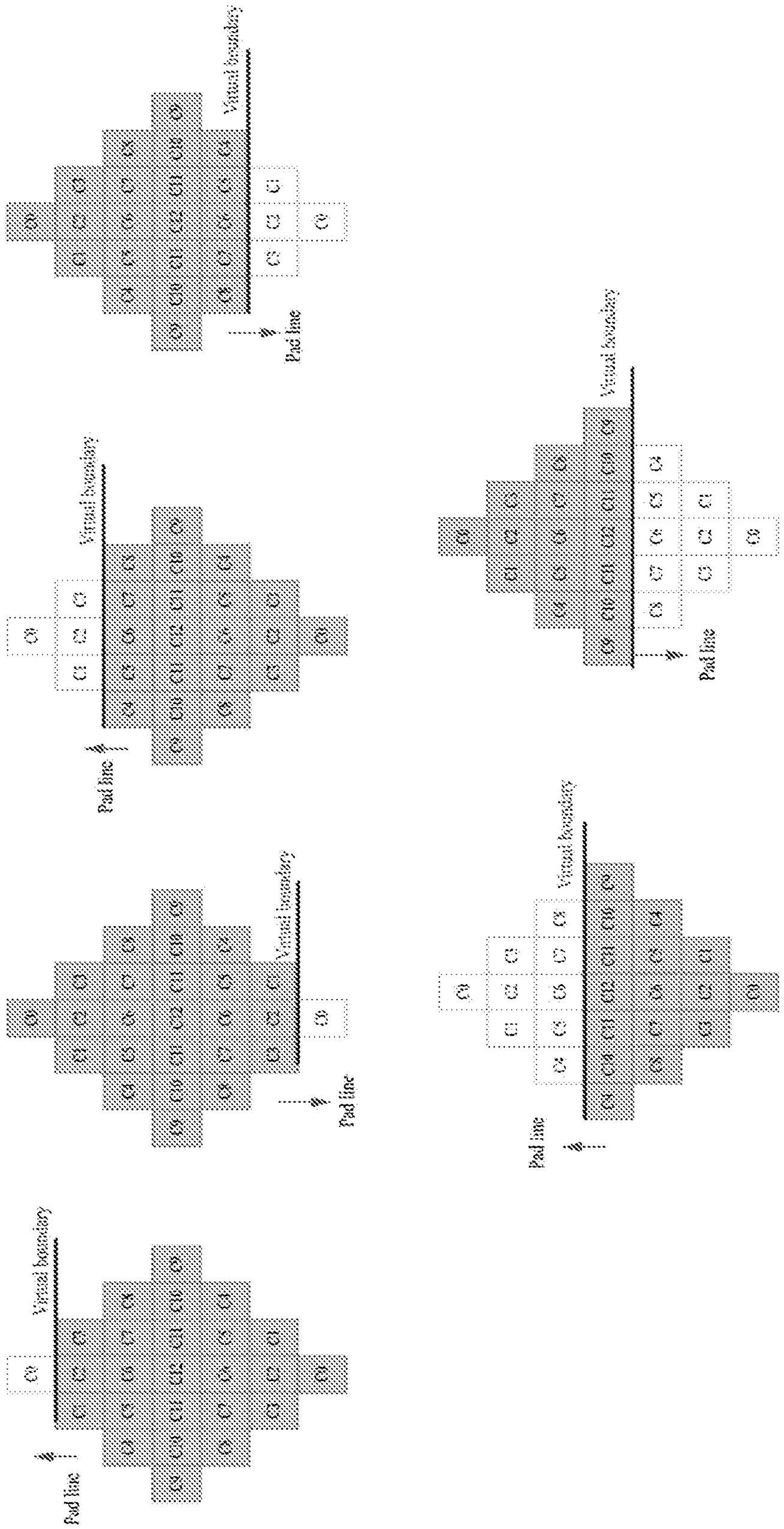


FIG. 18

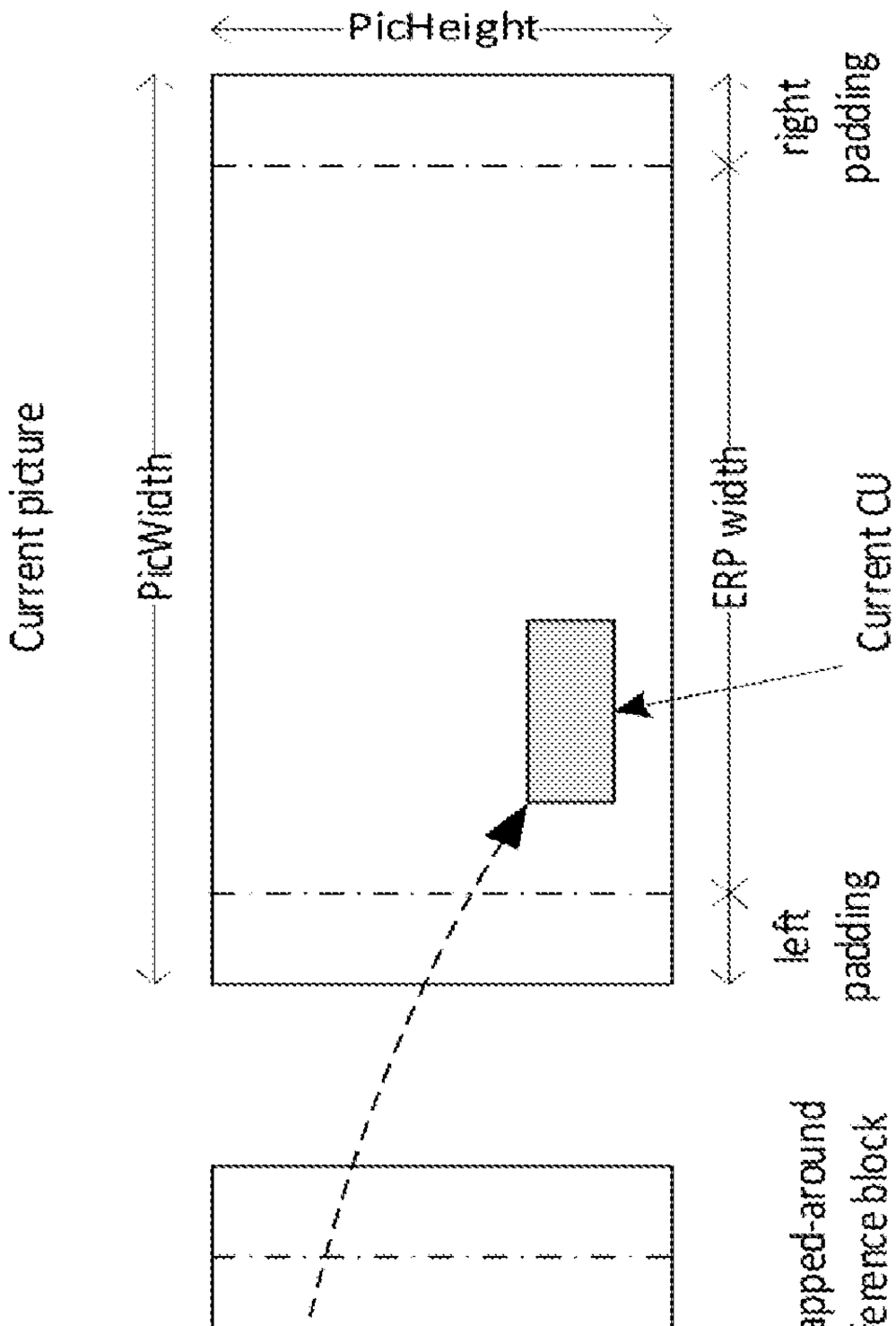

PicHeight
right padding
Current picture
PicWidth
ERP width
Current CU
left padding
Reference picture
ERP width
MV
Co-located CU
Wrapped-around reference block
Reference block (shaded part generated by horizontal wrap-around MC)

FIG. 19

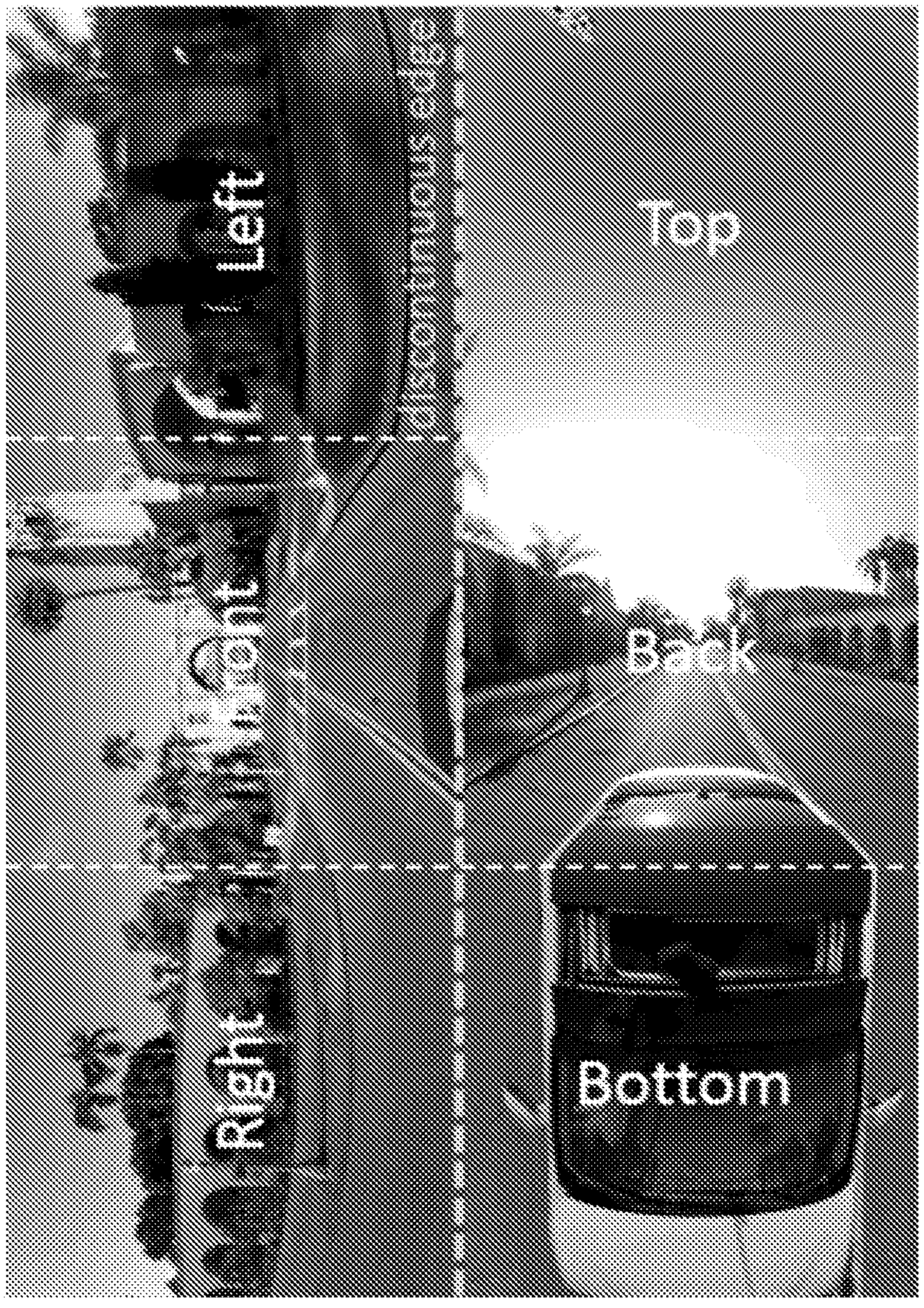


FIG. 20

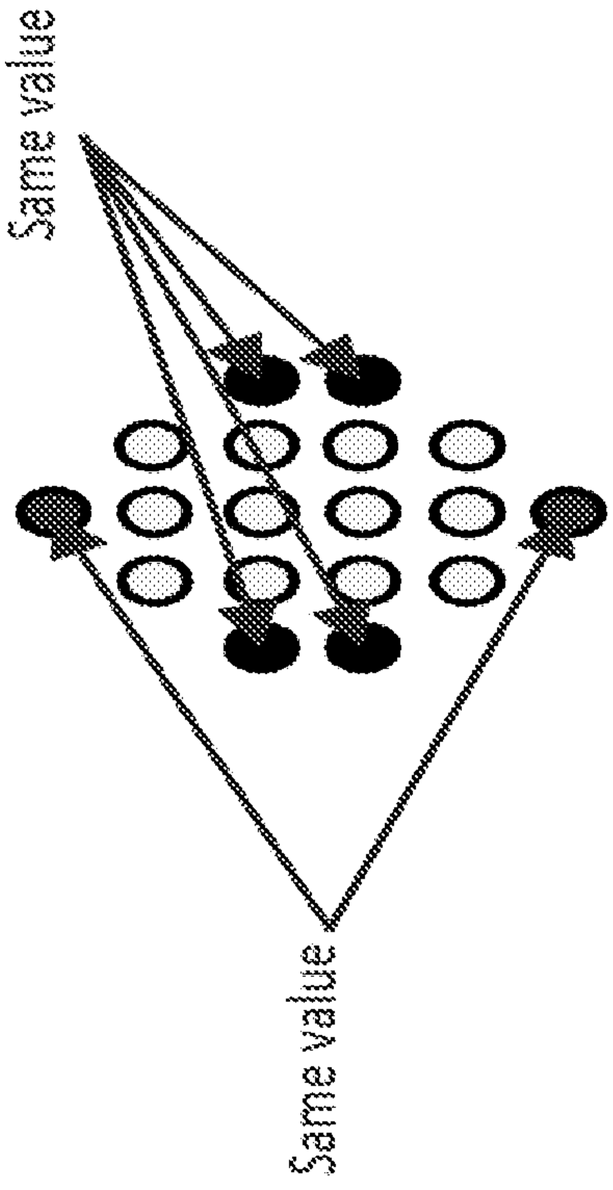


FIG. 21B

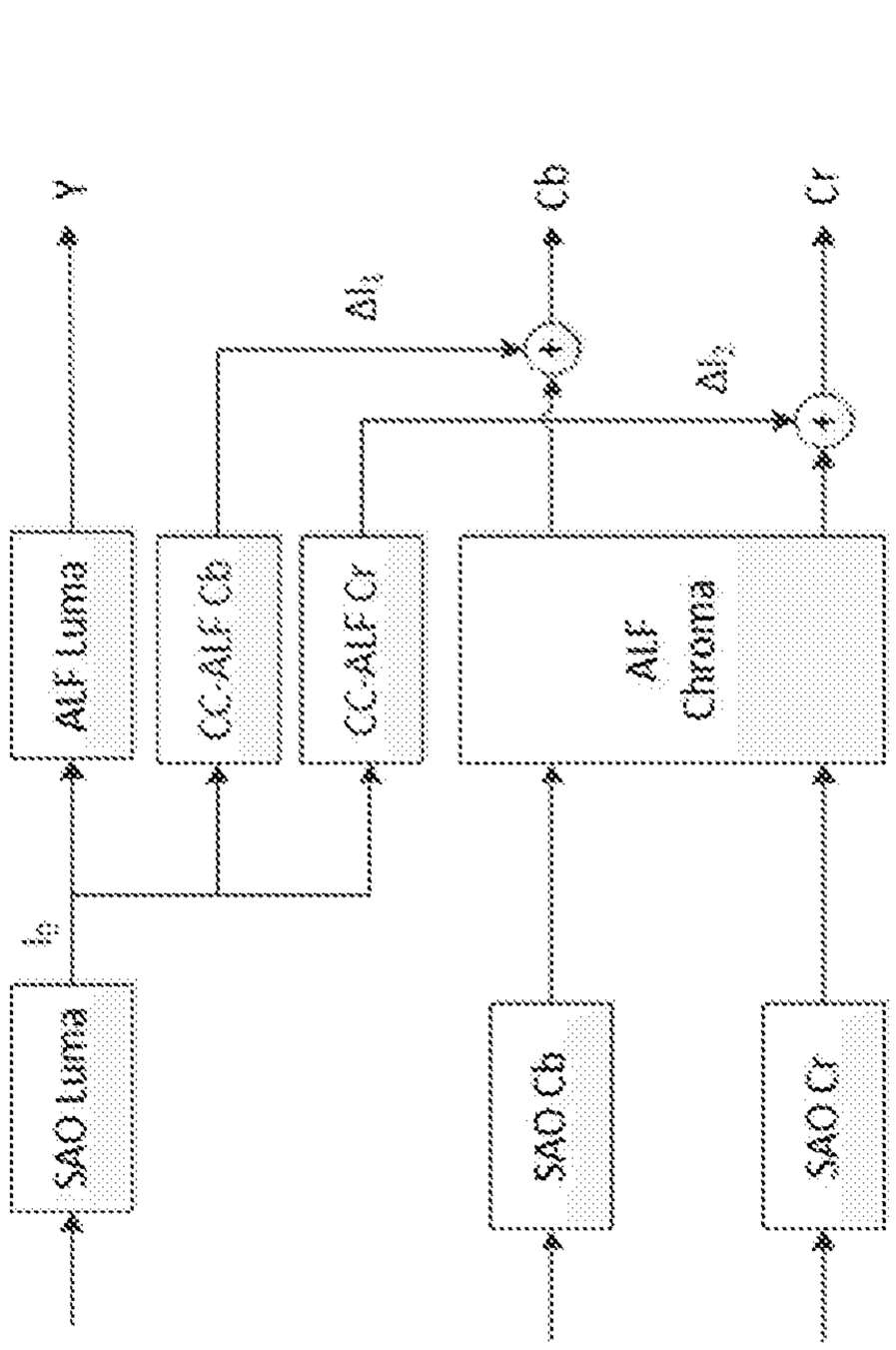


FIG. 21A

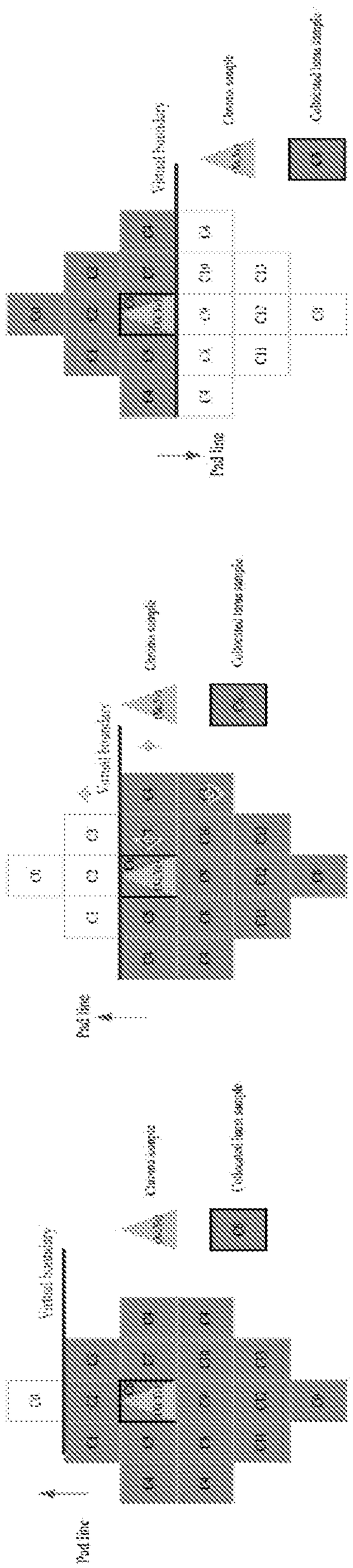

Virtual boundary
Change sample
Collected base sample
Pad line

FIG. 22

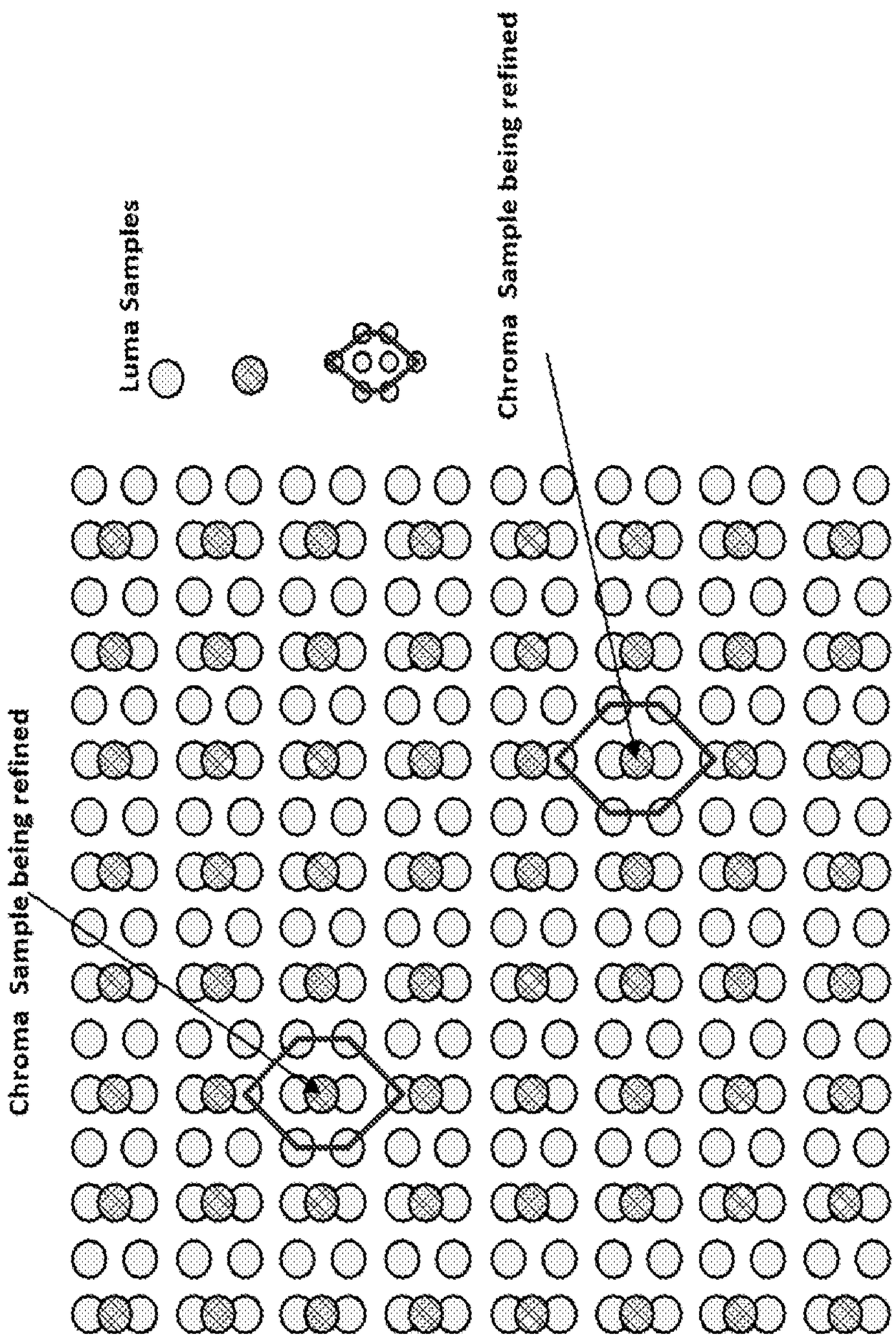

Luma Samples
Chroma Sample being refined
Chroma Sample being refined

FIG. 23

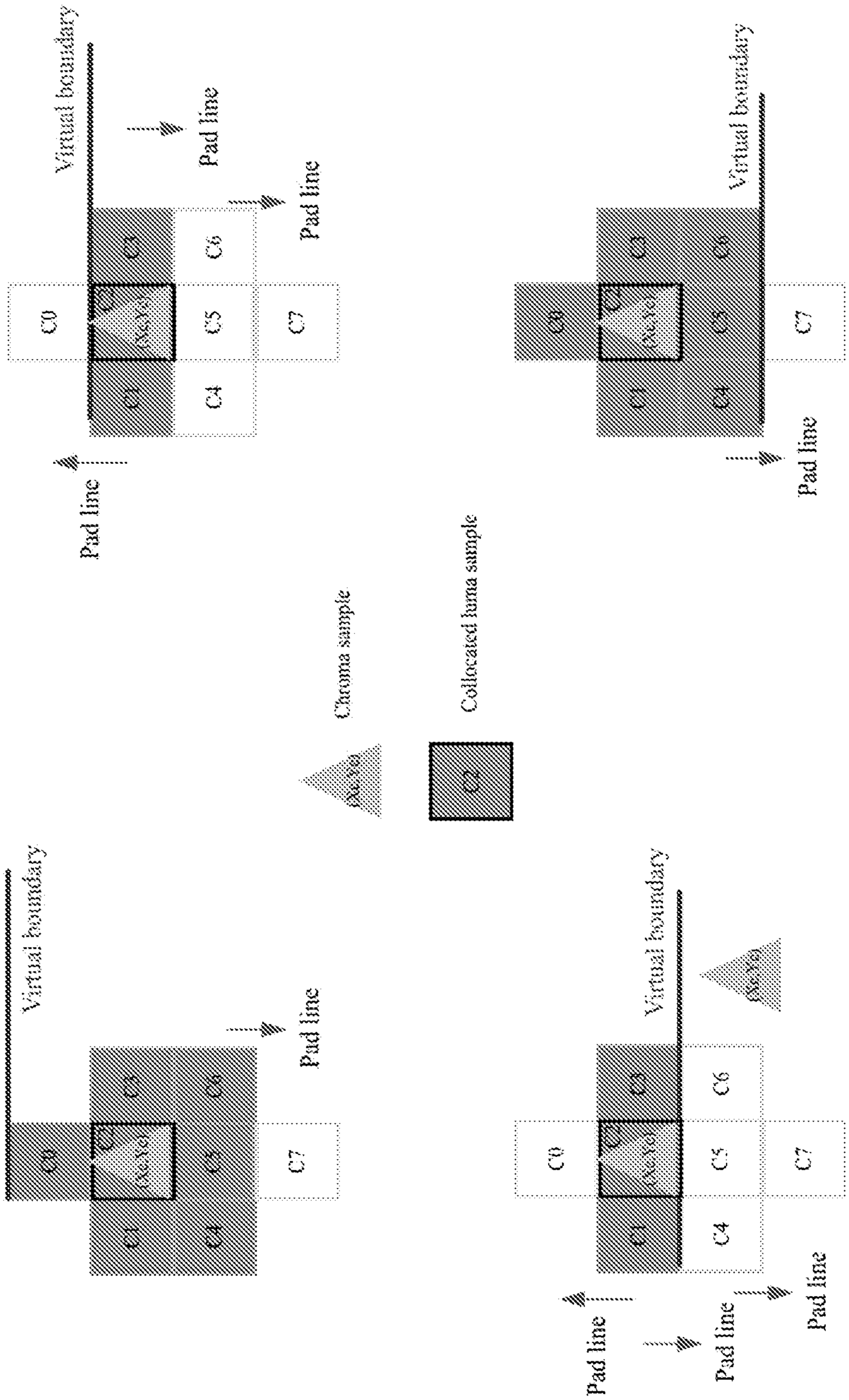

Virtual boundary
Pad line
Pad line
C0
C1
C2
C3
C4
C5
C6
C7
C8
Chroma sample
Collocated luma sample

FIG. 24

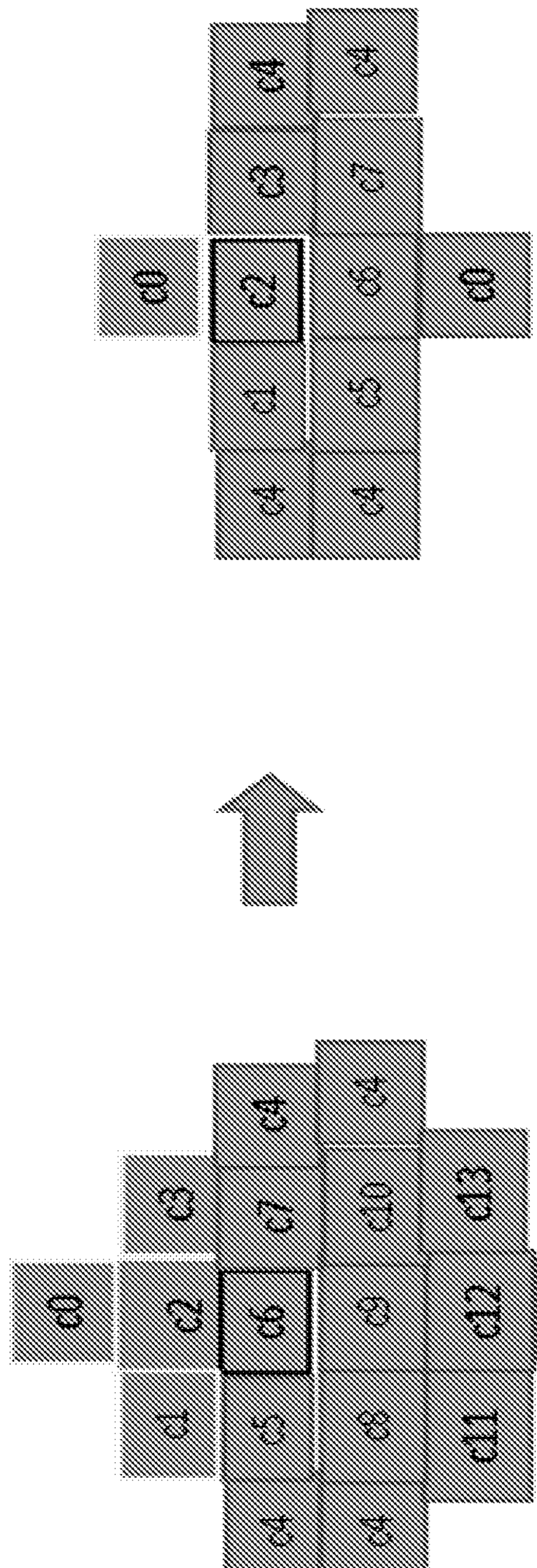

c4
c5
c3
c7
c0
c2
c6
c8
c1
c3
c4
c4
c7
c4
c10
c13
c0
c2
c6
c9
c12
c8
c1
c5
c8
c11
c4
c5
Center sample
c6

FIG. 25

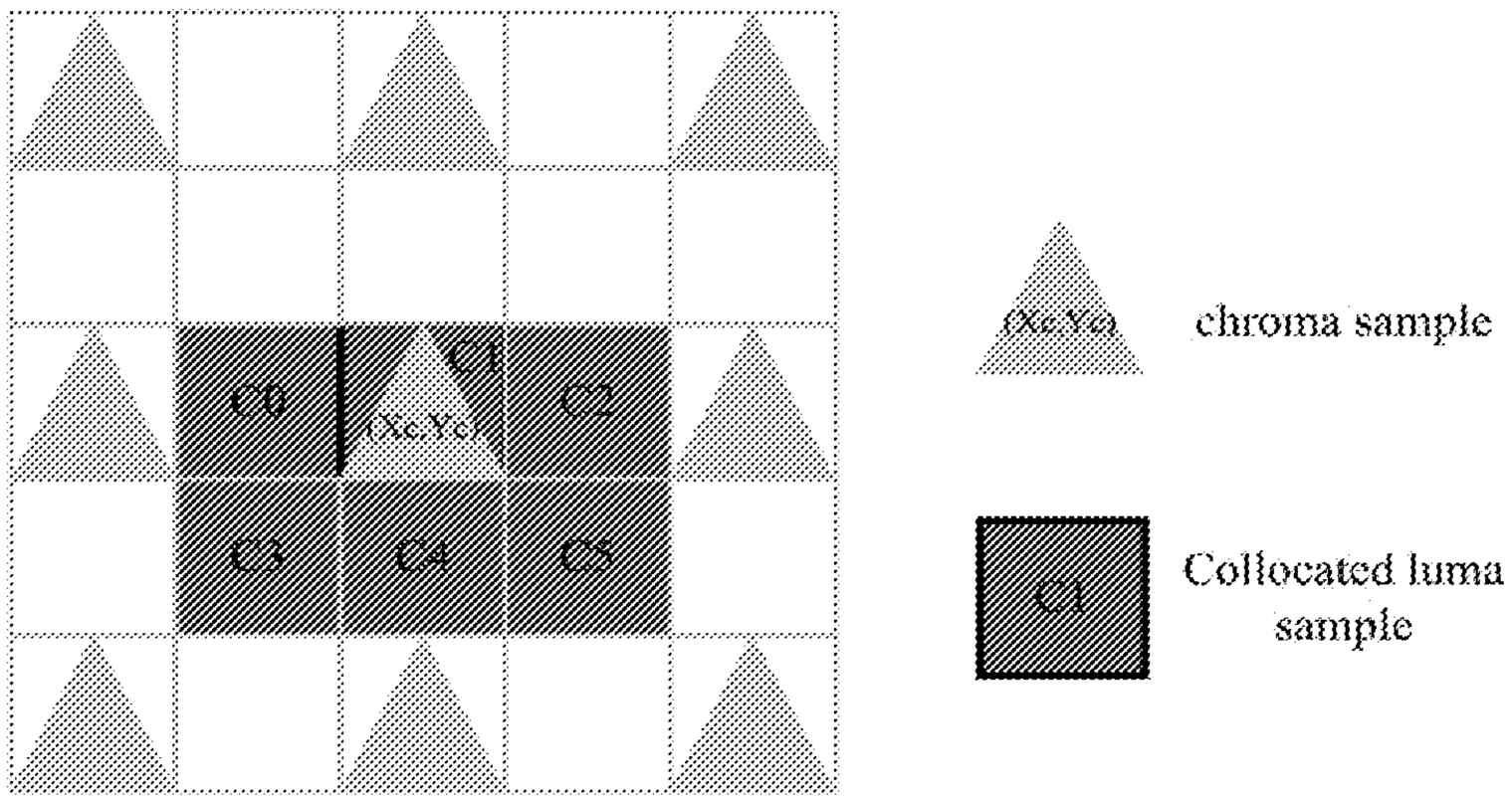

C0
C1
(Xc,Yc)
C2
C3
C4
C5
(Xc,Yc)
chroma sample
C1
Collocated luma sample

FIG. 26

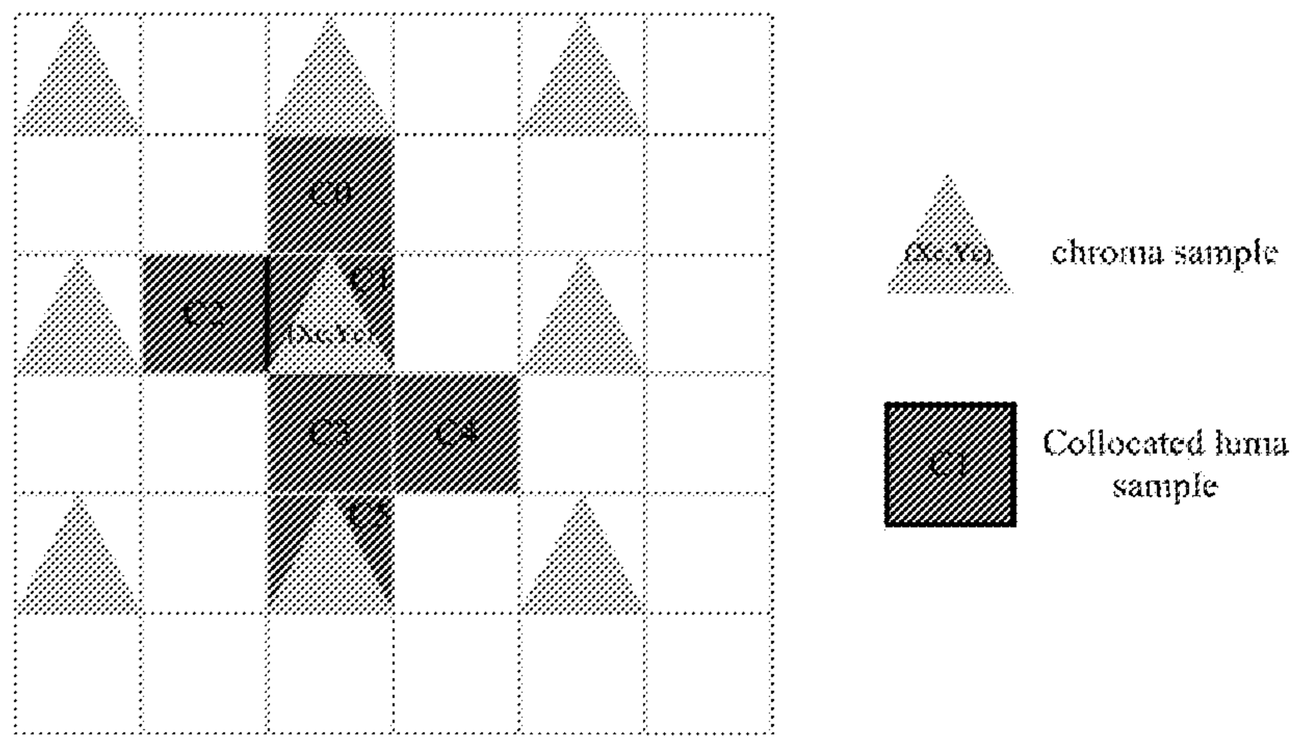

C0
C1
C2
(Xc,Yc)
C3
C4
C5
(Xc,Yc)
chroma sample
C1
Collocated luma sample

FIG. 27

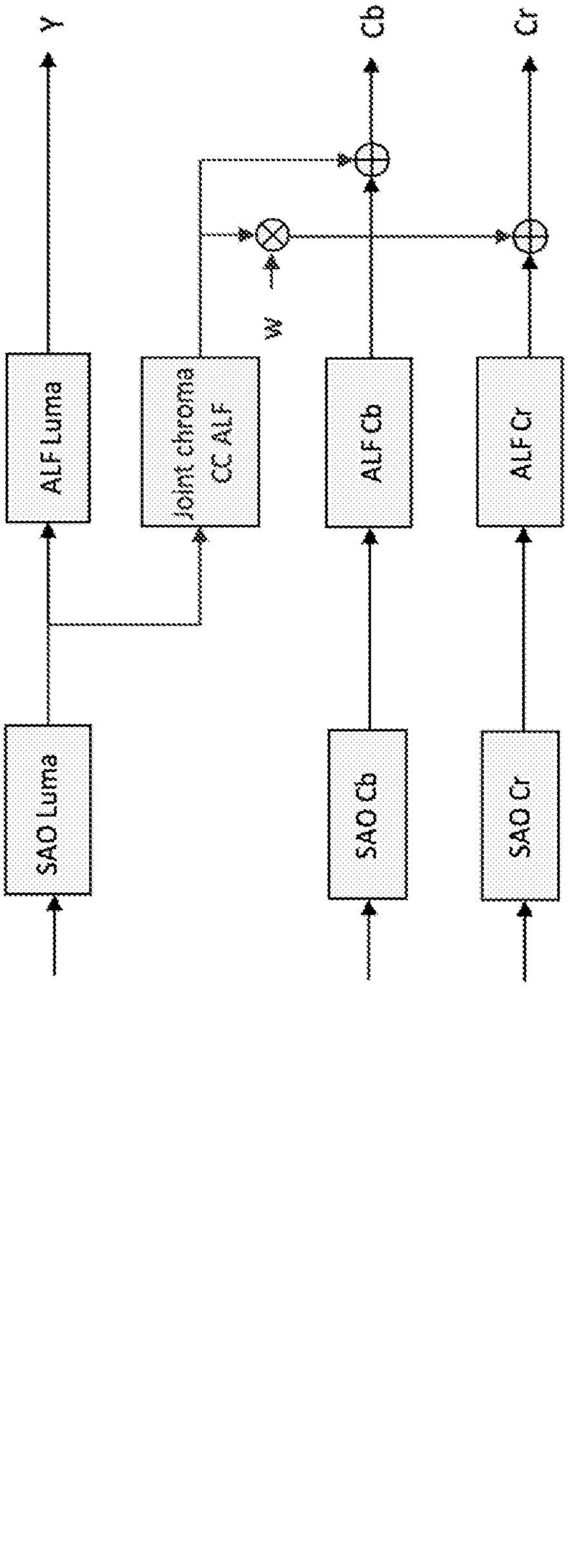

Y
Cb
Cr
ALF Luma
Joint chroma CC ALF
ALF Cb
ALF Cr
W
SAO Luma
SAO Cb
SAO Cr

FIG. 28

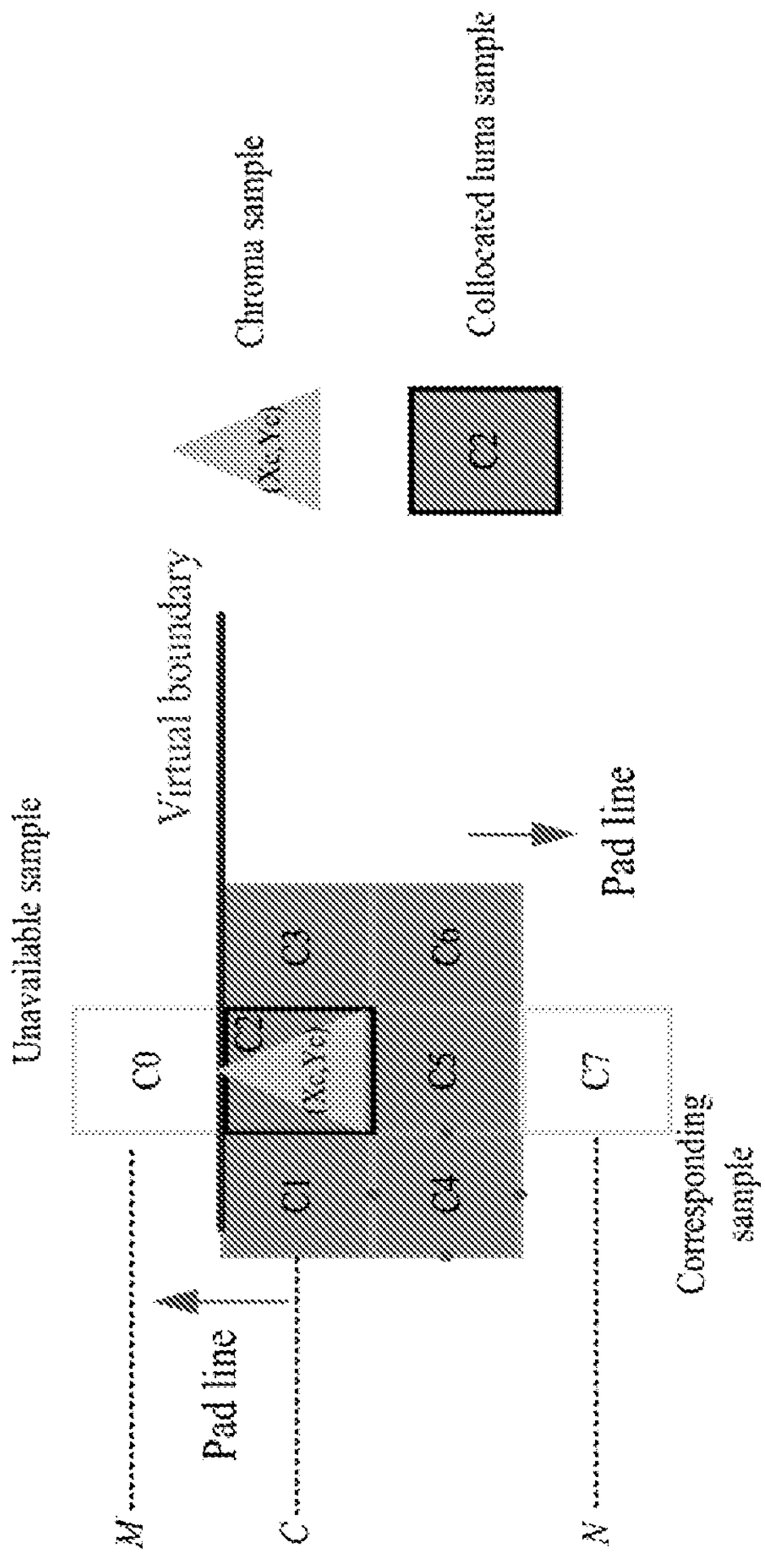

Unavailable sample
Virtual boundary
Chroma sample
Collocated luma sample
(x_c, y_c)
C
C0
C1
C2
C3
C4
C5
C6
C7
M
N
Pad line
Pad line
Corresponding sample

FIG. 29

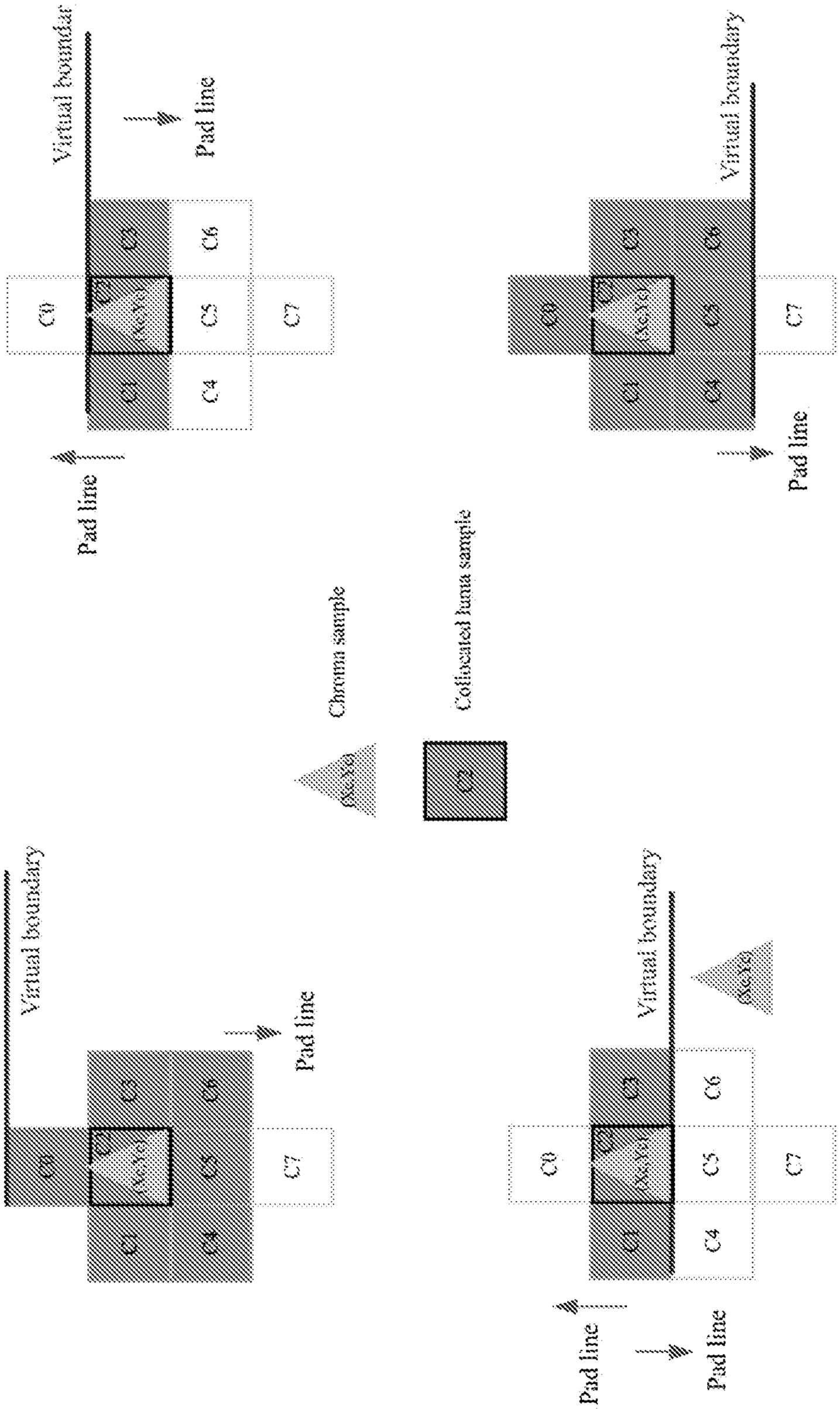

Virtual boundary
Pad line
C0
C1
C2
(x,y)
C3
C4
C5
C6
C7
Chroma sample
Collocated luma sample

FIG. 30

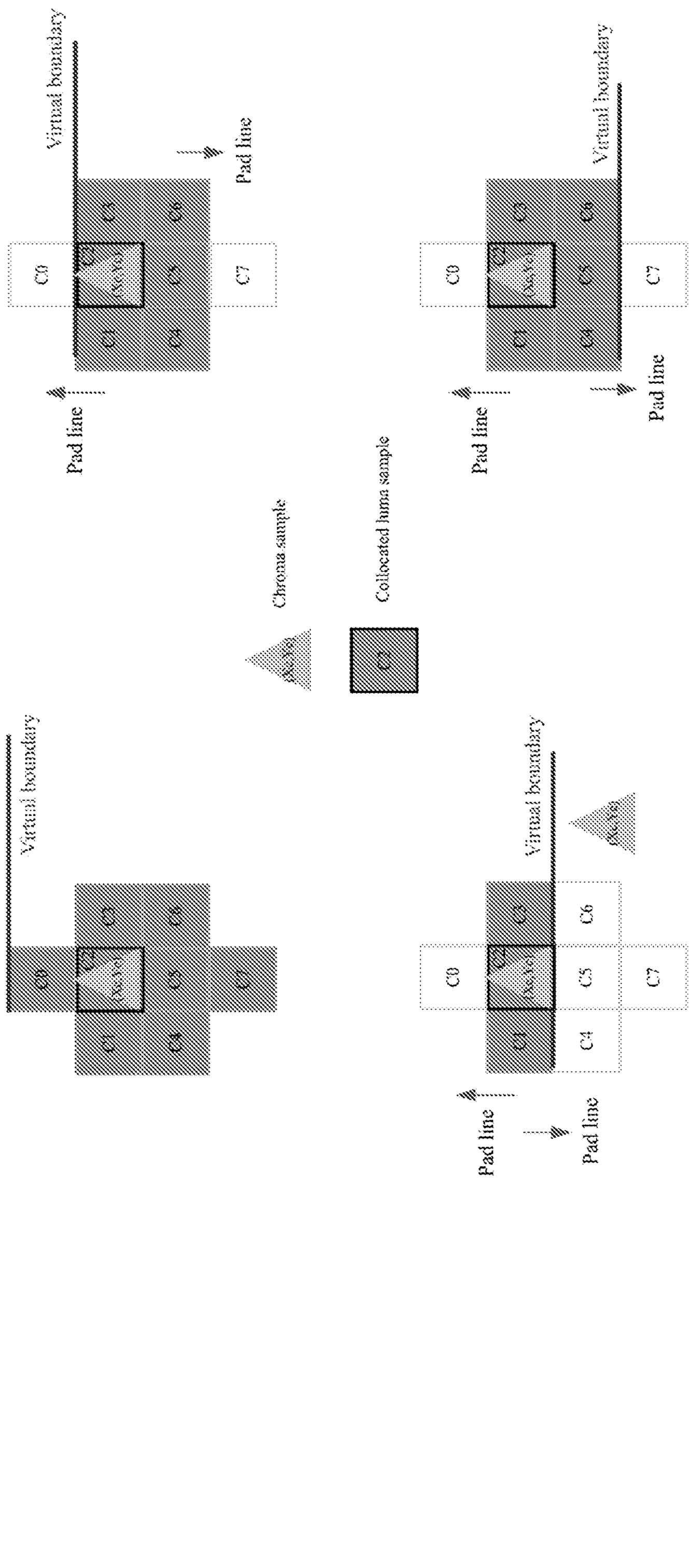

Virtual boundary
C0
C1
C2
(x,y)
C3
C4
C5
C6
C7
Pad line
Pad line
Chroma sample
Collocated luma sample
Virtual boundary
C0
C1
C2
(x,y)
C3
C4
C5
C6
C7
Pad line
Pad line

FIG. 31

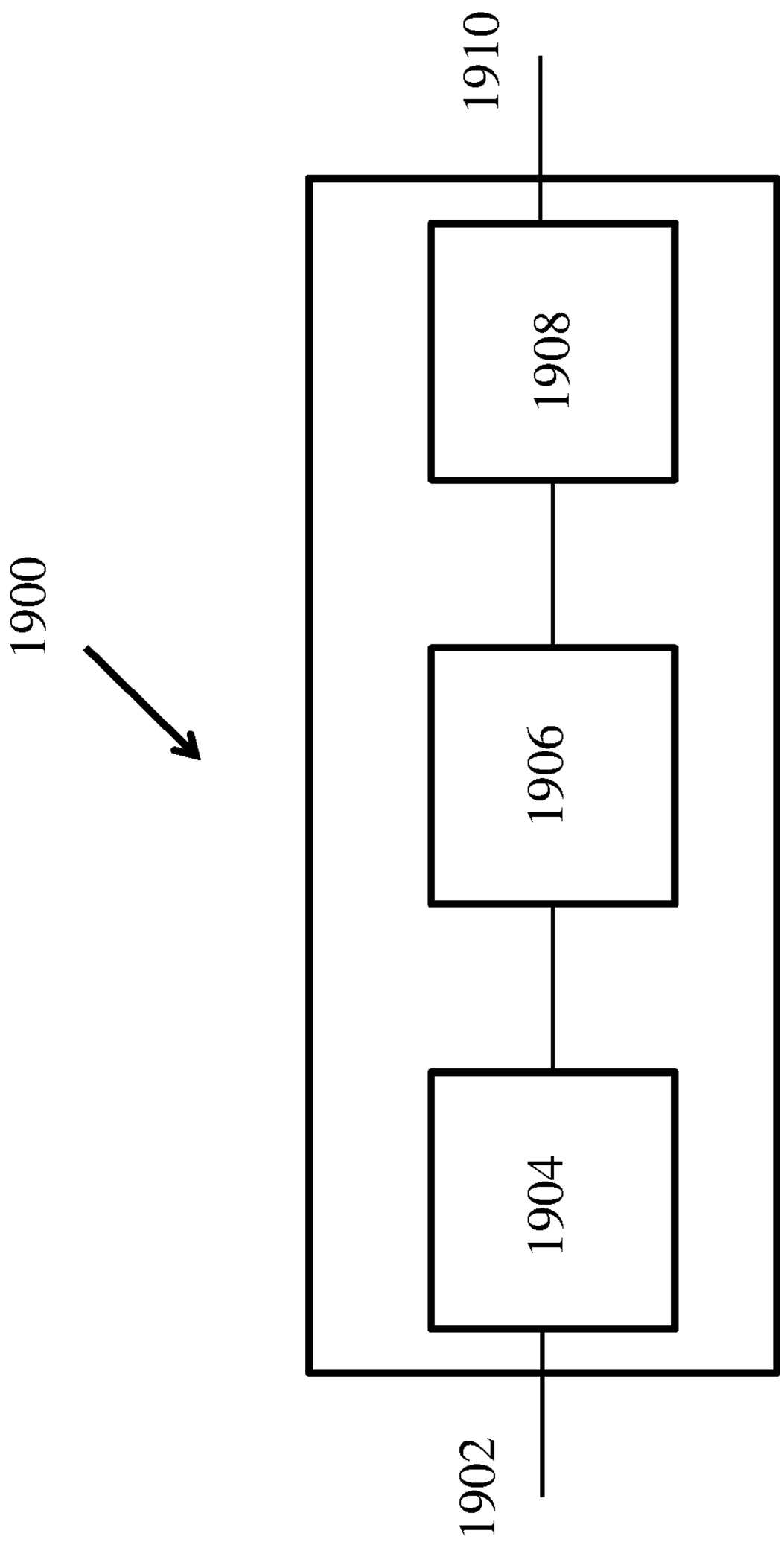


FIG. 32

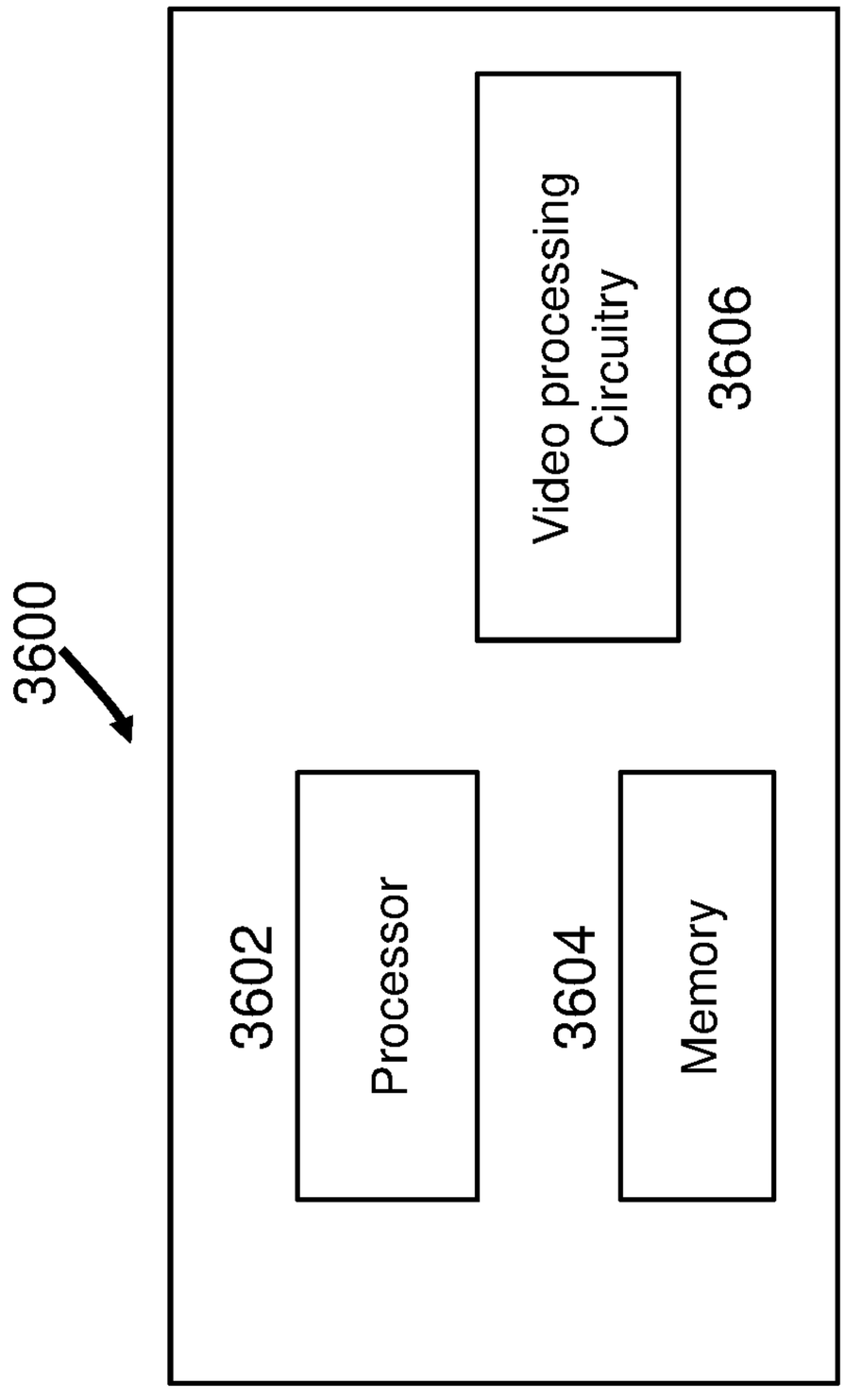

3600
Video processing Circuitry
3606
3602
Processor
3604
Memory

FIG. 33

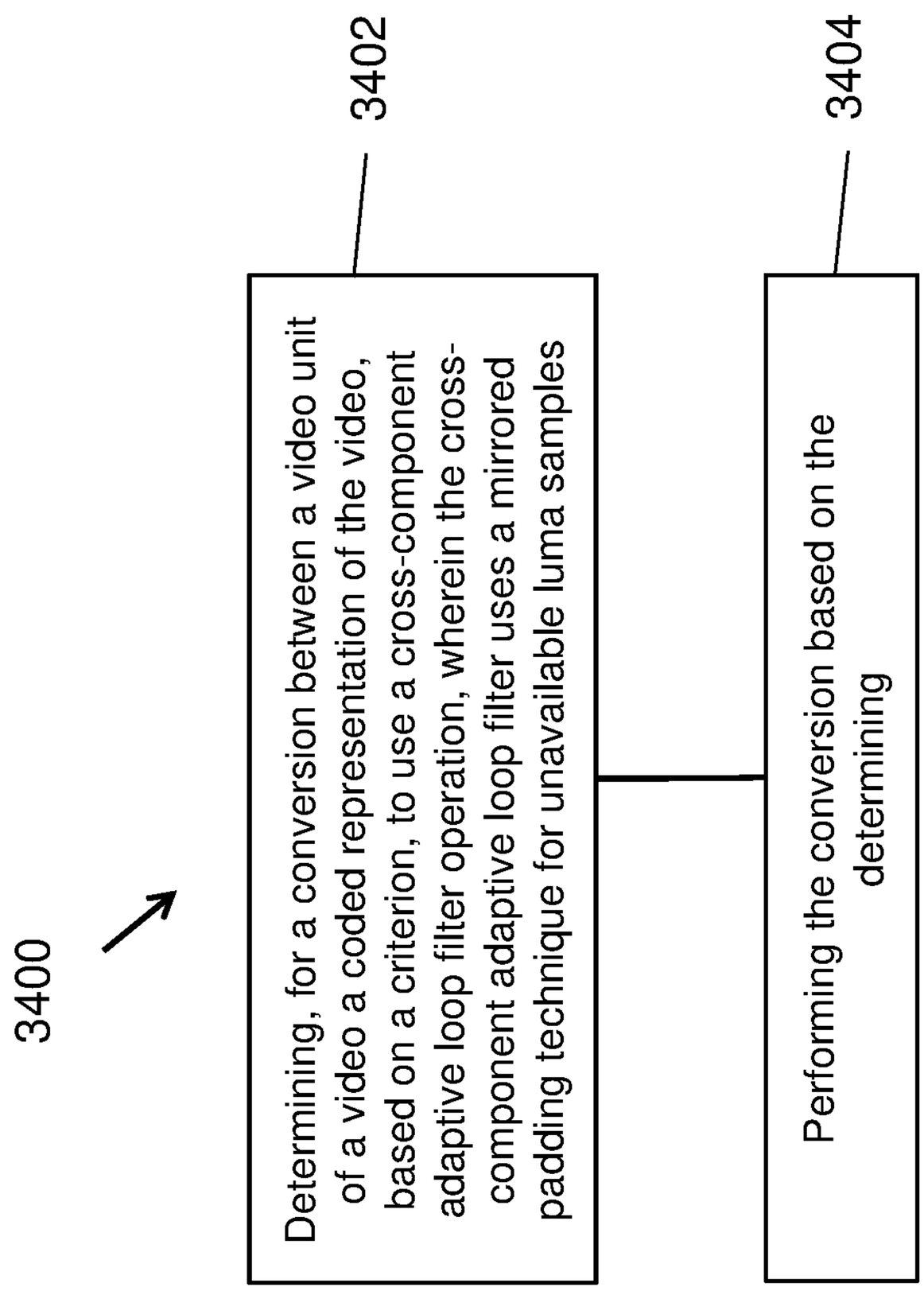

3400
Determining, for a conversion between a video unit of a video a coded representation of the video, based on a criterion, to use a cross-component adaptive loop filter operation, wherein the cross-component adaptive loop filter uses a mirrored padding technique for unavailable luma samples
3402
Performing the conversion based on the determining
3404

FIG. 34

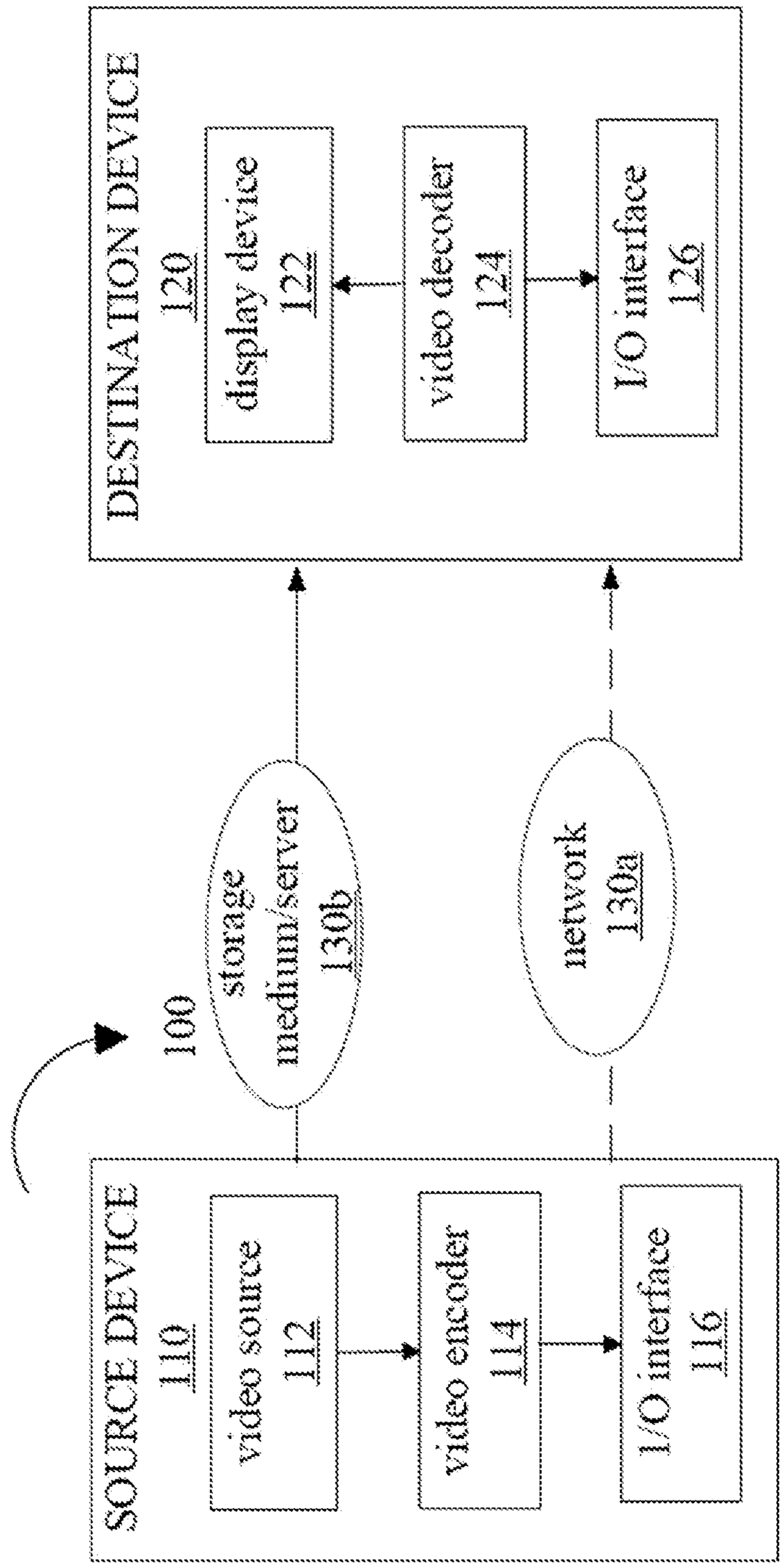

100
SOURCE DEVICE
110
video source
112
video encoder
114
I/O interface
116
storage medium/server
130b
network
130a
DESTINATION DEVICE
120
display device
122
video decoder
124
I/O interface
126

FIG. 35

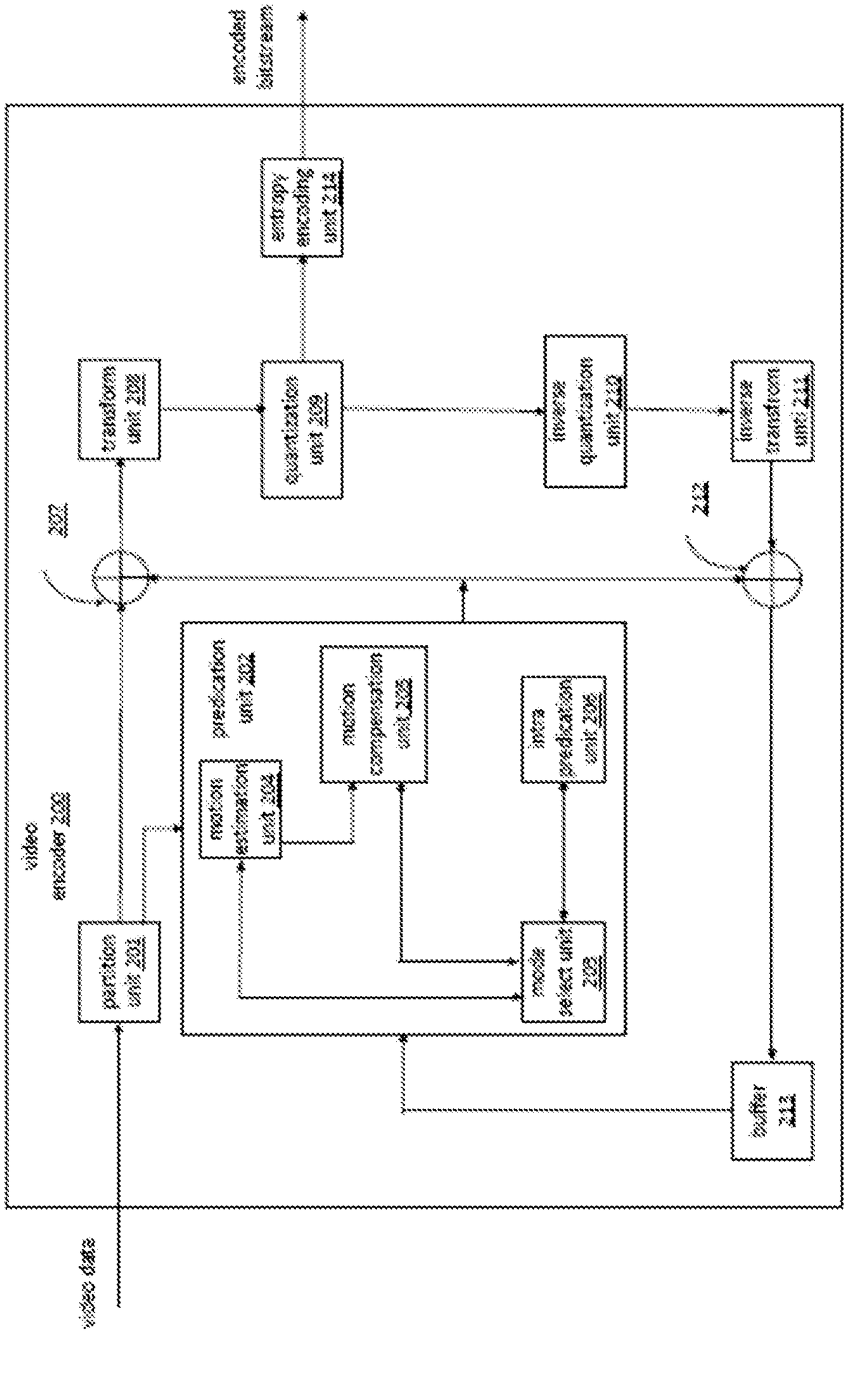

video data
video encoder 200
partition unit 201
prediction unit 202
motion estimation unit 204
motion compensation unit 205
intra prediction unit 206
mode select unit 203
202
residual unit 207
transform unit 208
quantization unit 209
entropy encoding unit 214
inverse quantization unit 210
inverse transform unit 211
212
buffer 213

FIG. 36

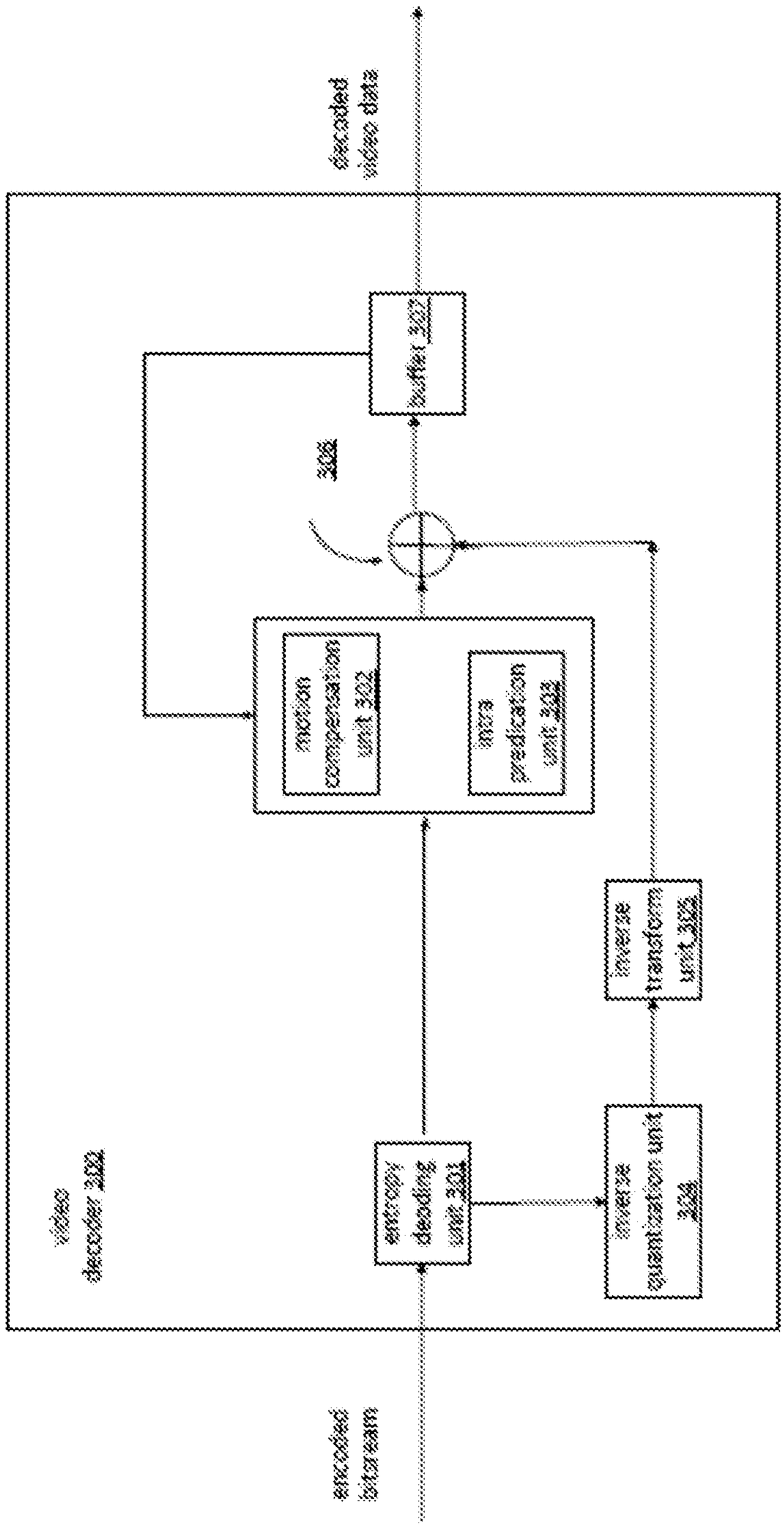

encoded bitstream
video decoder 300
entropy decoding unit 301
inverse quantization unit 303
inverse transform unit 305
reconstruction unit 307
buffer 309
loop filter 311
motion compensation unit 313
intra prediction unit 315
switch 317

FIG. 37

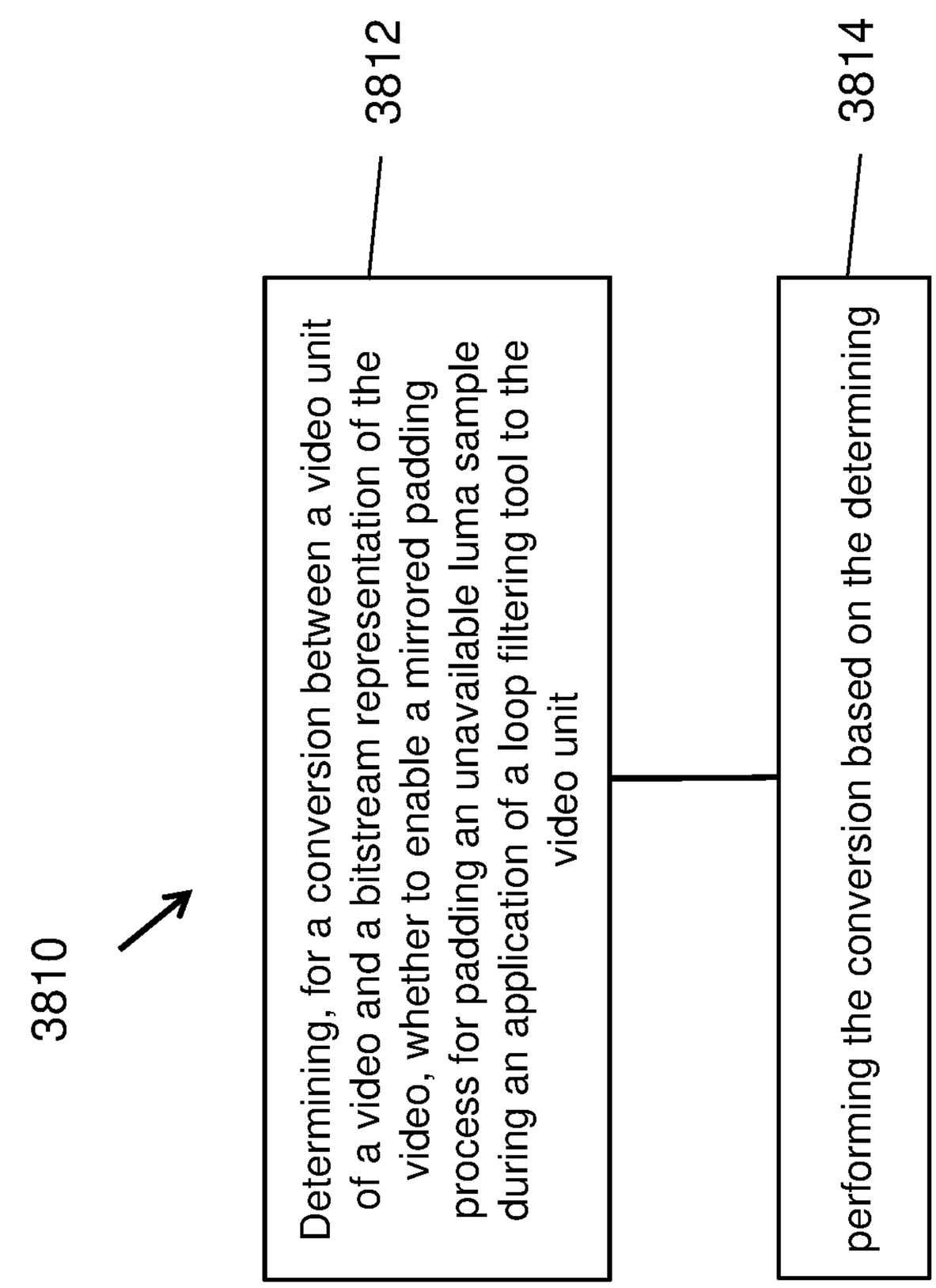

3810
Determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit
3812
performing the conversion based on the determining
3814

FIG. 38A

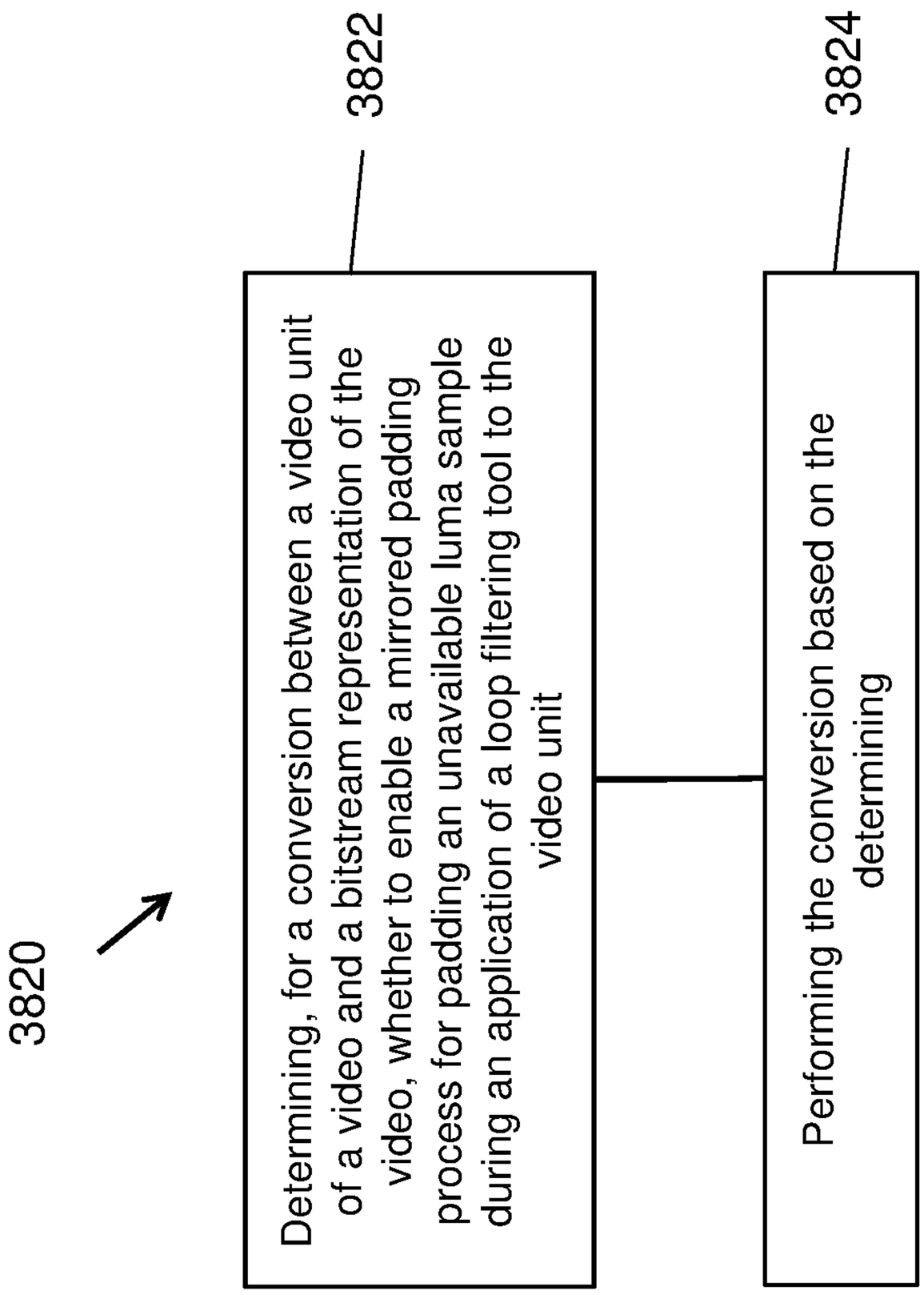

3820
Determining, for a conversion between a video unit of a video and a bitstream representation of the video, whether to enable a mirrored padding process for padding an unavailable luma sample during an application of a loop filtering tool to the video unit
3822
Performing the conversion based on the determining
3824

FIG. 38B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/135134

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/117(2014.01)i; H04N 19/82(2014.01)i

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)

H04N

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

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

CNPAT, CNKI, WPI, EPODOC, IEEE, JVET :adaptive loop filter, ALF, mirrored, symmetric, 2 side?, repetitive, asymmetric, 1 side, padding, sample, unavailable

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CHEN, Jianle et al. "JVET-P2002-v1:Algorithm description for Versatile Video Coding and Test Model 7 (VTM 7)" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 16th Meeting: Geneva, CH, 1–11 Oct. 2019*, 11 November 2019 (2019-11-11), section 3.7, Figures 49-50	1-58, 62-68
X	MISRA, K. et al. "JVET-P1008-v2: CE5-related: On the design of CC-ALF" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 16th Meeting: Geneva, CH, 1–11 Oct. 2019*, 11 October 2019 (2019-10-11), abstract, section 1	1-58, 62-68
X	LIU, Hongbin et al. "JVET-O0625:Non-CE5: Padding method for samples at variant boundaries in ALF" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 15th Meeting: Gothenburg, SE, 3–12 July 2019*, 12 July 2019 (2019-07-12), sections 1-2, figures 1-2	1-58, 62-68
X	ZHOU, Minhua. "JVET-P0053-v1:AHG16/HLS: A clean-up for the ALF sample padding" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 16th Meeting: Geneva, CH, 1–11 October 2019*, 11 October 2019 (2019-10-11), abstract, sections 1-3	1-58, 62-68

☑ 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
"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
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 January 2021	**03 March 2021**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **HE,Yanjuan** Telephone No. **86-(10)-53961817**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/135134

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2013058876 A1 (QUALCOMM INC. et al.) 25 April 2013 (2013-04-25) the whole document	1-68
A	CN 109600611 A (BEIJING DAJIA INTERCONNECTION INFORMATION TECH. CO., LTD.) 09 April 2019 (2019-04-09) the whole document	1-68

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/135134

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
WO 2013058876 A1	25 April 2013	US 2013101016 A1	25 April 2013
		ES 2677100 T3	30 July 2018
		EP 2769545 B1	18 April 2018
		CN 104247434 B	02 October 2018
		KR 20140081882 A	01 July 2014
		CN 104247434 A	24 December 2014
		JP 5876156 B2	02 March 2016
		EP 2769545 A1	27 August 2014
		KR 101607788 B1	30 March 2016
		US 9462298 B2	04 October 2016
		HU E038103 T2	28 September 2018
		JP 2014533008 A	08 December 2014
CN 109600611 A	09 April 2019	None	