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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 November 2022 (17.11.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/240436 A1

(51) International Patent Classification:
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/US2021/052051

(22) International Filing Date:
24 September 2021 (24.09.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/187,213 11 May 2021 (11.05.2021) US
17/448,469 22 September 2021 (22.09.2021) US

(71) Applicant: **TENCENT AMERICA LLC** [US/US]; 2747 Park Boulevard, Palo Alto, California 94306 (US).

(72) Inventors: **DU, Yixin**; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US). **LIU, Shan**; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US).

(74) Agent: **MA, Johnny** et al.; ArentFox Schiff LLP, 1717 K Street, Washington, DC 20006 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(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,

(54) **Title:** METHOD AND APPARATUS FOR BOUNDARY HANDLING IN VIDEO CODING

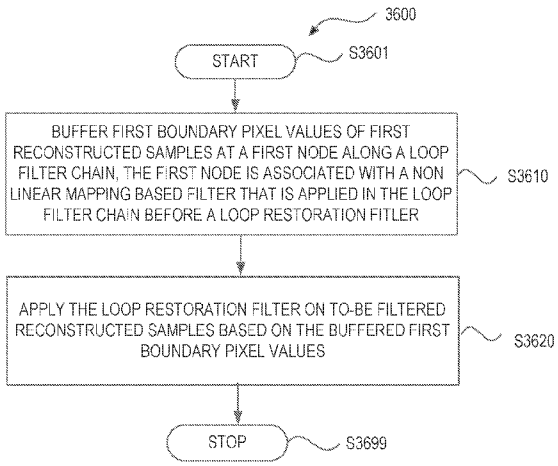

FIG. 36

(57) **Abstract:** Aspects of the disclosure provide methods and apparatuses for filtering in video encoding/decoding. In some examples, an apparatus for video coding includes processing circuitry. The processing circuitry buffers first boundary pixel values of first reconstructed samples at a first node along a loop filter chain. The first node is associated with a non linear mapping based filter that is applied in the loop filter chain before a loop restoration filter. The first boundary pixel values are values of pixels at a frame boundary. The processing circuitry applies the loop restoration filter on to-be filtered reconstructed samples based on the buffered first boundary pixel values.

WO 2022/240436 A1

[Continued on next page]

UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

METHOD AND APPARATUS FOR BOUNDARY HANDLING IN VIDEO CODING

INCORPORATION BY REFERENCE

[0001] This present application claims the benefit of priority to U.S. Patent Application No. 17/448,469, "METHOD AND APPARATUS FOR BOUNDARY HANDLING IN VIDEO CODING" filed on September 22, 2021, which claims the benefit of priority to U.S. Provisional Application No. 63/187,213, "ON LOOP RESTORATION BOUNDARY HANDLING" filed on May 11, 2021. The entire disclosure of the prior applications is hereby incorporated by reference in their entirety.

TECHNICAL FIELD

[0002] The present disclosure describes embodiments generally related to video coding.

BACKGROUND

[0003] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0004] Video coding and decoding can be performed using inter-picture prediction with motion compensation. Uncompressed digital video can include a series of pictures, each picture having a spatial dimension of, for example, 1920 x 1080 luminance samples and associated chrominance samples. The series of pictures can have a fixed or variable picture rate (informally also known as frame rate), of, for example 60 pictures per second or 60 Hz. Uncompressed video has specific bitrate requirements. For example, 1080p60 4:2:0 video at 8 bit per sample (1920x1080 luminance sample resolution at 60 Hz frame rate) requires close to 1.5 Gbit/s bandwidth. An hour of such video requires more than 600 GBytes of storage space.

[0005] One purpose of video coding and decoding can be the reduction of redundancy in the input video signal, through compression. Compression can help reduce the aforementioned bandwidth and/or storage space requirements, in some cases by two orders of magnitude or more. Both lossless compression and lossy compression, as well as a combination thereof can be employed. Lossless compression refers to techniques where an exact copy of the original signal can be reconstructed from the compressed original signal. When using lossy compression, the reconstructed signal may not be identical to the original signal, but the distortion between

original and reconstructed signals is small enough to make the reconstructed signal useful for the intended application. In the case of video, lossy compression is widely employed. The amount of distortion tolerated depends on the application; for example, users of certain consumer streaming applications may tolerate higher distortion than users of television distribution applications. The compression ratio achievable can reflect that: higher allowable/tolerable distortion can yield higher compression ratios.

[0006] A video encoder and decoder can utilize techniques from several broad categories, including, for example, motion compensation, transform, quantization, and entropy coding.

[0007] Video codec technologies can include techniques known as intra coding. In intra coding, sample values are represented without reference to samples or other data from previously reconstructed reference pictures. In some video codecs, the picture is spatially subdivided into blocks of samples. When all blocks of samples are coded in intra mode, that picture can be an intra picture. Intra pictures and their derivations such as independent decoder refresh pictures, can be used to reset the decoder state and can, therefore, be used as the first picture in a coded video bitstream and a video session, or as a still image. The samples of an intra block can be exposed to a transform, and the transform coefficients can be quantized before entropy coding. Intra prediction can be a technique that minimizes sample values in the pre-transform domain. In some cases, the smaller the DC value after a transform is, and the smaller the AC coefficients are, the fewer the bits that are required at a given quantization step size to represent the block after entropy coding.

[0008] Traditional intra coding such as known from, for example MPEG-2 generation coding technologies, does not use intra prediction. However, some newer video compression technologies include techniques that attempt, from, for example, surrounding sample data and/or metadata obtained during the encoding/decoding of spatially neighboring, and preceding in decoding order, blocks of data. Such techniques are henceforth called "intra prediction" techniques. Note that in at least some cases, intra prediction is using reference data only from the current picture under reconstruction and not from reference pictures.

[0009] There can be many different forms of intra prediction. When more than one of such techniques can be used in a given video coding technology, the technique in use can be coded in an intra prediction mode. In certain cases, modes can have submodes and/or parameters, and those can be coded individually or included in the mode codeword. Which codeword to use for a given mode/submode/parameter combination can have an impact in the

coding efficiency gain through intra prediction, and so can the entropy coding technology used to translate the codewords into a bitstream.

[0010] A certain mode of intra prediction was introduced with H.264, refined in H.265, and further refined in newer coding technologies such as joint exploration model (JEM), versatile video coding (VVC), and benchmark set (BMS). A predictor block can be formed using neighboring sample values belonging to already available samples. Sample values of neighboring samples are copied into the predictor block according to a direction. A reference to the direction in use can be coded in the bitstream or may itself be predicted.

[0011] Referring to FIG. 1A, depicted in the lower right is a subset of nine predictor directions known from H.265's 33 possible predictor directions (corresponding to the 33 angular modes of the 35 intra modes). The point where the arrows converge (101) represents the sample being predicted. The arrows represent the direction from which the sample is being predicted. For example, arrow (102) indicates that sample (101) is predicted from a sample or samples to the upper right, at a 45 degree angle from the horizontal. Similarly, arrow (103) indicates that sample (101) is predicted from a sample or samples to the lower left of sample (101), in a 22.5 degree angle from the horizontal.

[0012] Still referring to FIG. 1A, on the top left there is depicted a square block (104) of 4 x 4 samples (indicated by a dashed, boldface line). The square block (104) includes 16 samples, each labelled with an "S", its position in the Y dimension (e.g., row index) and its position in the X dimension (e.g., column index). For example, sample S21 is the second sample in the Y dimension (from the top) and the first (from the left) sample in the X dimension. Similarly, sample S44 is the fourth sample in block (104) in both the Y and X dimensions. As the block is 4 x 4 samples in size, S44 is at the bottom right. Further shown are reference samples that follow a similar numbering scheme. A reference sample is labelled with an R, its Y position (e.g., row index) and X position (column index) relative to block (104). In both H.264 and H.265, prediction samples neighbor the block under reconstruction; therefore no negative values need to be used.

[0013] Intra picture prediction can work by copying reference sample values from the neighboring samples as appropriated by the signaled prediction direction. For example, assume the coded video bitstream includes signaling that, for this block, indicates a prediction direction consistent with arrow (102)—that is, samples are predicted from a prediction sample or samples to the upper right, at a 45 degree angle from the horizontal. In that case, samples S41, S32, S23,

and S14 are predicted from the same reference sample R05. Sample S44 is then predicted from reference sample R08.

[0014] In certain cases, the values of multiple reference samples may be combined, for example through interpolation, in order to calculate a reference sample; especially when the directions are not evenly divisible by 45 degrees.

[0015] The number of possible directions has increased as video coding technology has developed. In H.264 (year 2003), nine different direction could be represented. That increased to 33 in H.265 (year 2013), and JEM/VVC/BMS, at the time of disclosure, can support up to 65 directions. Experiments have been conducted to identify the most likely directions, and certain techniques in the entropy coding are used to represent those likely directions in a small number of bits, accepting a certain penalty for less likely directions. Further, the directions themselves can sometimes be predicted from neighboring directions used in neighboring, already decoded, blocks.

[0016] FIG. 1B shows a schematic (180) that depicts 65 intra prediction directions according to JEM to illustrate the increasing number of prediction directions over time.

[0017] The mapping of intra prediction directions bits in the coded video bitstream that represent the direction can be different from video coding technology to video coding technology; and can range, for example, from simple direct mappings of prediction direction to intra prediction mode, to codewords, to complex adaptive schemes involving most probable modes, and similar techniques. In all cases, however, there can be certain directions that are statistically less likely to occur in video content than certain other directions. As the goal of video compression is the reduction of redundancy, those less likely directions will, in a well working video coding technology, be represented by a larger number of bits than more likely directions.

[0018] Motion compensation can be a lossy compression technique and can relate to techniques where a block of sample data from a previously reconstructed picture or part thereof (reference picture), after being spatially shifted in a direction indicated by a motion vector (MV henceforth), is used for the prediction of a newly reconstructed picture or picture part. In some cases, the reference picture can be the same as the picture currently under reconstruction. MVs can have two dimensions X and Y, or three dimensions, the third being an indication of the reference picture in use (the latter, indirectly, can be a time dimension).

[0019] In some video compression techniques, an MV applicable to a certain area of sample data can be predicted from other MVs, for example from those related to another area of sample data spatially adjacent to the area under reconstruction, and preceding that MV in decoding order. Doing so can substantially reduce the amount of data required for coding the MV, thereby removing redundancy and increasing compression. MV prediction can work effectively, for example, because when coding an input video signal derived from a camera (known as natural video) there is a statistical likelihood that areas larger than the area to which a single MV is applicable move in a similar direction and, therefore, can in some cases be predicted using a similar motion vector derived from MVs of neighboring area. That results in the MV found for a given area to be similar or the same as the MV predicted from the surrounding MVs, and that in turn can be represented, after entropy coding, in a smaller number of bits than what would be used if coding the MV directly. In some cases, MV prediction can be an example of lossless compression of a signal (namely: the MVs) derived from the original signal (namely: the sample stream). In other cases, MV prediction itself can be lossy, for example because of rounding errors when calculating a predictor from several surrounding MVs.

[0020] Various MV prediction mechanisms are described in H.265/HEVC (ITU-T Rec. H.265, "High Efficiency Video Coding", December 2016). Out of the many MV prediction mechanisms that H.265 offers, described here is a technique henceforth referred to as "spatial merge".

[0021] Referring to FIG. 2, a current block (201) comprises samples that have been found by the encoder during the motion search process to be predictable from a previous block of the same size that has been spatially shifted. Instead of coding that MV directly, the MV can be derived from metadata associated with one or more reference pictures, for example from the most recent (in decoding order) reference picture, using the MV associated with either one of five surrounding samples, denoted A0, A1, and B0, B1, B2 (202 through 206, respectively). In H.265, the MV prediction can use predictors from the same reference picture that the neighboring block is using.

SUMMARY

[0022] Aspects of the disclosure provide methods and apparatuses for filtering in video encoding/decoding. In some examples, an apparatus for video coding includes processing circuitry. The processing circuitry buffers first boundary pixel values of first reconstructed samples at a first node along a loop filter chain. The first node is associated with a non linear

mapping based filter that is applied in the loop filter chain before a loop restoration filter. The first boundary pixel values are values of pixels at a frame boundary in an example. The processing circuitry applies the loop restoration filter on to-be filtered reconstructed samples based on the buffered first boundary pixel values.

[0023] In some examples, the non linear mapping based filter is a cross-component sample offset (CCSO) filter. In some examples, the non linear mapping based filter is a local sample offset (LSO) filter.

[0024] In some examples, the first reconstructed samples at the first node are input of the non linear mapping based filter. In some examples, the processing circuitry buffers second boundary pixel values of second reconstructed samples at a second node along the loop filter chain. The second reconstructed samples at the second node are generated after an applying of sample offsets generated by the non linear mapping based filter. The second boundary pixel values are values of the pixels at the frame boundary. Then, the processing circuitry can apply the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

[0025] In an example, the processing circuitry combines the sample offsets generated by the non linear mapping based filter with output of a constrained directional enhanced filter to generate the second reconstructed samples. In another example, the processing circuitry combines the sample offsets generated by the non linear mapping based filter with the first reconstructed samples to generate intermediate reconstructed samples, and applies a constrained directional enhanced filter to the intermediate reconstructed samples to generate the second reconstructed samples.

[0026] In some examples, the processing circuitry buffers second boundary pixel values of second reconstructed samples at a second node along the loop filter, and combines the second reconstructed samples at the second node with sample offsets generated by the non linear mapping based filter to generate the to-be filtered reconstructed samples. The second boundary pixel values are values of the pixels at the frame boundary. Then, the processing circuitry applies the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

[0027] In some examples, the processing circuitry buffers second boundary pixel values from second reconstructed samples generated by a deblocking filter, and applies a constrained directional enhanced filter on the second reconstructed samples to generate intermediate

reconstructed samples. The second boundary pixel values are values of the pixels at the frame boundary. The processing circuitry combines the intermediate reconstructed samples with sample offsets generated by the non linear mapping based filter to generate the first reconstructed samples. Then, the loop restoration filter can be applied based on the buffered first boundary pixel values and the buffered second boundary pixel values.

[0028] In some examples, the processing circuitry clips the to-be filtered reconstructed samples before the applying of the loop restoration filter. In some examples, the processing circuitry clips intermediate reconstructed samples before a combination with sample offsets generated by the non linear mapping based filter.

[0029] Aspects of the disclosure also provide a non-transitory computer-readable medium storing instructions which when executed by a computer cause the computer to perform any of the methods for video encoding/decoding.

BRIEF DESCRIPTION OF THE DRAWINGS

[0030] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0031] FIG. 1A is a schematic illustration of an exemplary subset of intra prediction modes.

[0032] FIG. 1B is an illustration of exemplary intra prediction directions.

[0033] FIG. 2 is a schematic illustration of a current block and its surrounding spatial merge candidates in one example.

[0034] FIG. 3 is a schematic illustration of a simplified block diagram of a communication system (300) in accordance with an embodiment.

[0035] FIG. 4 is a schematic illustration of a simplified block diagram of a communication system (400) in accordance with an embodiment.

[0036] FIG. 5 is a schematic illustration of a simplified block diagram of a decoder in accordance with an embodiment.

[0037] FIG. 6 is a schematic illustration of a simplified block diagram of an encoder in accordance with an embodiment.

[0038] FIG. 7 shows a block diagram of an encoder in accordance with another embodiment.

[0039] FIG. 8 shows a block diagram of a decoder in accordance with another embodiment.

[0040] FIG. 9 shows examples of filter shapes according to embodiments of the disclosure.

[0041] FIGs. 10A-10D show examples of subsampled positions used for calculating gradients according to embodiments of the disclosure.

[0042] FIGs. 11A-11B show examples of a virtual boundary filtering process according to embodiments of the disclosure.

[0043] FIGs. 12A-12F show examples of symmetric padding operations at virtual boundaries according to embodiments of the disclosure.

[0044] FIG. 13 shows a partition example of a picture according to some embodiments of the disclosure.

[0045] FIG. 14 shows a quadtree split pattern for a picture in some examples.

[0046] FIG. 15 shows cross-component filters according to an embodiment of the disclosure.

[0047] FIG. 16 shows an example of a filter shape according to an embodiment of the disclosure.

[0048] FIG. 17 shows a syntax example for cross component filter according to some embodiments of the disclosure.

[0049] FIGs. 18A-18B show exemplary locations of chroma samples relative to luma samples according to embodiments of the disclosure.

[0050] FIG. 19 shows an example of direction search according to an embodiment of the disclosure.

[0051] FIG. 20 shows an example illustrating subspace projection in some examples.

[0052] FIG. 21 shows a table of a plurality of sample adaptive offset (SAO) types according to an embodiment of the disclosure.

[0053] FIG. 22 shows examples of patterns for pixel classification in edge offset in some examples.

[0054] FIG. 23 shows a table for pixel classification rule for edge offset in some examples.

[0055] FIG. 24 shows an example of syntaxes that may be signaled.

[0056] FIG. 25 shows an example of a filter support area according to some embodiments of the disclosure.

[0057] FIG. 26 shows an example of another filter support area according to some embodiments of the disclosure.

[0058] FIGs. 27A-27C show a table having 81 combinations according to an embodiment of the disclosure.

[0059] FIG. 28 shows 7 filter shape configurations of 3 filter taps in an example.

[0060] FIG. 29 shows a block diagram of a loop filter chain in some examples.

[0061] FIG. 30 shows an example of a loop filter chain that includes a non linear mapping based filter.

[0062] FIG. 31 shows an example of another loop filter chain that includes a non linear mapping based filter.

[0063] FIG. 32 shows an example of another loop filter chain that includes a non linear mapping based filter.

[0064] FIG. 33 shows an example of another loop filter chain that includes a non linear mapping based filter.

[0065] FIG. 34 shows an example of another loop filter chain that includes a non linear mapping based filter.

[0066] FIG. 35 shows an example of another loop filter chain that includes a non linear mapping based filter.

[0067] FIG. 36 shows a flow chart outlining a process according to an embodiment of the disclosure.

[0068] FIG. 37 is a schematic illustration of a computer system in accordance with an embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS

[0069] FIG. 3 illustrates a simplified block diagram of a communication system (300) according to an embodiment of the present disclosure. The communication system (300) includes a plurality of terminal devices that can communicate with each other, via, for example, a network (350). For example, the communication system (300) includes a first pair of terminal devices (310) and (320) interconnected via the network (350). In the FIG. 3 example, the first pair of terminal devices (310) and (320) performs unidirectional transmission of data. For example, the terminal device (310) may code video data (e.g., a stream of video pictures that are

captured by the terminal device (310)) for transmission to the other terminal device (320) via the network (350). The encoded video data can be transmitted in the form of one or more coded video bitstreams. The terminal device (320) may receive the coded video data from the network (350), decode the coded video data to recover the video pictures and display video pictures according to the recovered video data. Unidirectional data transmission may be common in media serving applications and the like.

[0070] In another example, the communication system (300) includes a second pair of terminal devices (330) and (340) that performs bidirectional transmission of coded video data that may occur, for example, during videoconferencing. For bidirectional transmission of data, in an example, each terminal device of the terminal devices (330) and (340) may code video data (e.g., a stream of video pictures that are captured by the terminal device) for transmission to the other terminal device of the terminal devices (330) and (340) via the network (350). Each terminal device of the terminal devices (330) and (340) also may receive the coded video data transmitted by the other terminal device of the terminal devices (330) and (340), and may decode the coded video data to recover the video pictures and may display video pictures at an accessible display device according to the recovered video data.

[0071] In the FIG. 3 example, the terminal devices (310), (320), (330) and (340) may be illustrated as servers, personal computers and smart phones but the principles of the present disclosure may be not so limited. Embodiments of the present disclosure find application with laptop computers, tablet computers, media players and/or dedicated video conferencing equipment. The network (350) represents any number of networks that convey coded video data among the terminal devices (310), (320), (330) and (340), including for example wireline (wired) and/or wireless communication networks. The communication network (350) may exchange data in circuit-switched and/or packet-switched channels. Representative networks include telecommunications networks, local area networks, wide area networks and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network (350) may be immaterial to the operation of the present disclosure unless explained herein below.

[0072] FIG. 4 illustrates, as an example for an application for the disclosed subject matter, the placement of a video encoder and a video decoder in a streaming environment. The disclosed subject matter can be equally applicable to other video enabled applications, including, for example, video conferencing, digital TV, storing of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[0073] A streaming system may include a capture subsystem (413), that can include a video source (401), for example a digital camera, creating for example a stream of video pictures (402) that are uncompressed. In an example, the stream of video pictures (402) includes samples that are taken by the digital camera. The stream of video pictures (402), depicted as a bold line to emphasize a high data volume when compared to encoded video data (404) (or coded video bitstreams), can be processed by an electronic device (420) that includes a video encoder (403) coupled to the video source (401). The video encoder (403) can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The encoded video data (404) (or encoded video bitstream (404)), depicted as a thin line to emphasize the lower data volume when compared to the stream of video pictures (402), can be stored on a streaming server (405) for future use. One or more streaming client subsystems, such as client subsystems (406) and (408) in FIG. 4 can access the streaming server (405) to retrieve copies (407) and (409) of the encoded video data (404). A client subsystem (406) can include a video decoder (410), for example, in an electronic device (430). The video decoder (410) decodes the incoming copy (407) of the encoded video data and creates an outgoing stream of video pictures (411) that can be rendered on a display (412) (e.g., display screen) or other rendering device (not depicted). In some streaming systems, the encoded video data (404), (407), and (409) (e.g., video bitstreams) can be encoded according to certain video coding/compression standards. Examples of those standards include ITU-T Recommendation H.265. In an example, a video coding standard under development is informally known as Versatile Video Coding (VVC). The disclosed subject matter may be used in the context of VVC.

[0074] It is noted that the electronic devices (420) and (430) can include other components (not shown). For example, the electronic device (420) can include a video decoder (not shown) and the electronic device (430) can include a video encoder (not shown) as well.

[0075] FIG. 5 shows a block diagram of a video decoder (510) according to an embodiment of the present disclosure. The video decoder (510) can be included in an electronic device (530). The electronic device (530) can include a receiver (531) (e.g., receiving circuitry). The video decoder (510) can be used in the place of the video decoder (410) in the FIG. 4 example.

[0076] The receiver (531) may receive one or more coded video sequences to be decoded by the video decoder (510); in the same or another embodiment, one coded video sequence at a

time, where the decoding of each coded video sequence is independent from other coded video sequences. The coded video sequence may be received from a channel (501), which may be a hardware/software link to a storage device which stores the encoded video data. The receiver (531) may receive the encoded video data with other data, for example, coded audio data and/or ancillary data streams, that may be forwarded to their respective using entities (not depicted). The receiver (531) may separate the coded video sequence from the other data. To combat network jitter, a buffer memory (515) may be coupled in between the receiver (531) and an entropy decoder / parser (520) ("parser (520)" henceforth). In certain applications, the buffer memory (515) is part of the video decoder (510). In others, it can be outside of the video decoder (510) (not depicted). In still others, there can be a buffer memory (not depicted) outside of the video decoder (510), for example to combat network jitter, and in addition another buffer memory (515) inside the video decoder (510), for example to handle playout timing. When the receiver (531) is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer memory (515) may not be needed, or can be small. For use on best effort packet networks such as the Internet, the buffer memory (515) may be required, can be comparatively large and can be advantageously of adaptive size, and may at least partially be implemented in an operating system or similar elements (not depicted) outside of the video decoder (510).

[0077] The video decoder (510) may include the parser (520) to reconstruct symbols (521) from the coded video sequence. Categories of those symbols include information used to manage operation of the video decoder (510), and potentially information to control a rendering device such as a render device (512) (e.g., a display screen) that is not an integral part of the electronic device (530) but can be coupled to the electronic device (530), as was shown in FIG. 5. The control information for the rendering device(s) may be in the form of Supplemental Enhancement Information (SEI messages) or Video Usability Information (VUI) parameter set fragments (not depicted). The parser (520) may parse / entropy-decode the coded video sequence that is received. The coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow various principles, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser (520) may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameter corresponding to the group. Subgroups can include Groups of Pictures (GOPs), pictures, tiles,

slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The parser (520) may also extract from the coded video sequence information such as transform coefficients, quantizer parameter values, motion vectors, and so forth.

[0078] The parser (520) may perform an entropy decoding / parsing operation on the video sequence received from the buffer memory (515), so as to create symbols (521).

[0079] Reconstruction of the symbols (521) can involve multiple different units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. Which units are involved, and how, can be controlled by the subgroup control information that was parsed from the coded video sequence by the parser (520). The flow of such subgroup control information between the parser (520) and the multiple units below is not depicted for clarity.

[0080] Beyond the functional blocks already mentioned, the video decoder (510) can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these units interact closely with each other and can, at least partly, be integrated into each other. However, for the purpose of describing the disclosed subject matter, the conceptual subdivision into the functional units below is appropriate.

[0081] A first unit is the scaler / inverse transform unit (551). The scaler / inverse transform unit (551) receives a quantized transform coefficient as well as control information, including which transform to use, block size, quantization factor, quantization scaling matrices, etc. as symbol(s) (521) from the parser (520). The scaler / inverse transform unit (551) can output blocks comprising sample values, that can be input into aggregator (555).

[0082] In some cases, the output samples of the scaler / inverse transform (551) can pertain to an intra coded block; that is: a block that is not using predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit (552). In some cases, the intra picture prediction unit (552) generates a block of the same size and shape of the block under reconstruction, using surrounding already reconstructed information fetched from the current picture buffer (558). The current picture buffer (558) buffers, for example, partly reconstructed current picture and/or fully reconstructed current picture. The aggregator (555), in some cases, adds, on a per sample

basis, the prediction information the intra prediction unit (552) has generated to the output sample information as provided by the scaler / inverse transform unit (551).

[0083] In other cases, the output samples of the scaler / inverse transform unit (551) can pertain to an inter coded, and potentially motion compensated block. In such a case, a motion compensation prediction unit (553) can access reference picture memory (557) to fetch samples used for prediction. After motion compensating the fetched samples in accordance with the symbols (521) pertaining to the block, these samples can be added by the aggregator (555) to the output of the scaler / inverse transform unit (551) (in this case called the residual samples or residual signal) so as to generate output sample information. The addresses within the reference picture memory (557) from where the motion compensation prediction unit (553) fetches prediction samples can be controlled by motion vectors, available to the motion compensation prediction unit (553) in the form of symbols (521) that can have, for example X, Y, and reference picture components. Motion compensation also can include interpolation of sample values as fetched from the reference picture memory (557) when sub-sample exact motion vectors are in use, motion vector prediction mechanisms, and so forth.

[0084] The output samples of the aggregator (555) can be subject to various loop filtering techniques in the loop filter unit (556). Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video sequence (also referred to as coded video bitstream) and made available to the loop filter unit (556) as symbols (521) from the parser (520), but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered sample values.

[0085] The output of the loop filter unit (556) can be a sample stream that can be output to the render device (512) as well as stored in the reference picture memory (557) for use in future inter-picture prediction.

[0086] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future prediction. For example, once a coded picture corresponding to a current picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, the parser (520)), the current picture buffer (558) can become a part of the reference picture memory (557), and a fresh current picture buffer can be reallocated before commencing the reconstruction of the following coded picture.

[0087] The video decoder (510) may perform decoding operations according to a predetermined video compression technology in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that the coded video sequence adheres to both the syntax of the video compression technology or standard and the profiles as documented in the video compression technology or standard. Specifically, a profile can select certain tools as the only tools available for use under that profile from all the tools available in the video compression technology or standard. Also necessary for compliance can be that the complexity of the coded video sequence is within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size, maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[0088] In an embodiment, the receiver (531) may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder (510) to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[0089] FIG. 6 shows a block diagram of a video encoder (603) according to an embodiment of the present disclosure. The video encoder (603) is included in an electronic device (620). The electronic device (620) includes a transmitter (640) (e.g., transmitting circuitry). The video encoder (603) can be used in the place of the video encoder (403) in the FIG. 4 example.

[0090] The video encoder (603) may receive video samples from a video source (601) (that is not part of the electronic device (620) in the FIG. 6 example) that may capture video image(s) to be coded by the video encoder (603). In another example, the video source (601) is a part of the electronic device (620).

[0091] The video source (601) may provide the source video sequence to be coded by the video encoder (603) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, ...), any color space (for example, BT.601 Y CrCB,

RGB, ...), and any suitable sampling structure (for example Y CrCb 4:2:0, Y CrCb 4:4:4). In a media serving system, the video source (601) may be a storage device storing previously prepared video. In a videoconferencing system, the video source (601) may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual pictures that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, etc. in use. A person skilled in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[0092] According to an embodiment, the video encoder (603) may code and compress the pictures of the source video sequence into a coded video sequence (643) in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed is one function of a controller (650). In some embodiments, the controller (650) controls other functional units as described below and is functionally coupled to the other functional units. The coupling is not depicted for clarity. Parameters set by the controller (650) can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, ...), picture size, group of pictures (GOP) layout, maximum motion vector search range, and so forth. The controller (650) can be configured to have other suitable functions that pertain to the video encoder (603) optimized for a certain system design.

[0093] In some embodiments, the video encoder (603) is configured to operate in a coding loop. As an oversimplified description, in an example, the coding loop can include a source coder (630) (e.g., responsible for creating symbols, such as a symbol stream, based on an input picture to be coded, and a reference picture(s)), and a (local) decoder (633) embedded in the video encoder (603). The decoder (633) reconstructs the symbols to create the sample data in a similar manner as a (remote) decoder also would create (as any compression between symbols and coded video bitstream is lossless in the video compression technologies considered in the disclosed subject matter). The reconstructed sample stream (sample data) is input to the reference picture memory (634). As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the content in the reference picture memory (634) is also bit exact between the local encoder and remote encoder. In other words, the prediction part of an encoder "sees" as reference picture samples exactly the same sample values as a decoder would "see" when using prediction during decoding. This fundamental principle of

reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is used in some related arts as well.

[0094] The operation of the "local" decoder (633) can be the same as of a "remote" decoder, such as the video decoder (510), which has already been described in detail above in conjunction with FIG. 5. Briefly referring also to FIG. 5, however, as symbols are available and encoding/decoding of symbols to a coded video sequence by an entropy coder (645) and the parser (520) can be lossless, the entropy decoding parts of the video decoder (510), including the buffer memory (515), and parser (520) may not be fully implemented in the local decoder (633).

[0095] An observation that can be made at this point is that any decoder technology except the parsing/entropy decoding that is present in a decoder also necessarily needs to be present, in substantially identical functional form, in a corresponding encoder. For this reason, the disclosed subject matter focuses on decoder operation. The description of encoder technologies can be abbreviated as they are the inverse of the comprehensively described decoder technologies. Only in certain areas a more detail description is required and provided below.

[0096] During operation, in some examples, the source coder (630) may perform motion compensated predictive coding, which codes an input picture predictively with reference to one or more previously coded picture from the video sequence that were designated as "reference pictures." In this manner, the coding engine (632) codes differences between pixel blocks of an input picture and pixel blocks of reference picture(s) that may be selected as prediction reference(s) to the input picture.

[0097] The local video decoder (633) may decode coded video data of pictures that may be designated as reference pictures, based on symbols created by the source coder (630). Operations of the coding engine (632) may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in FIG. 6), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder (633) replicates decoding processes that may be performed by the video decoder on reference pictures and may cause reconstructed reference pictures to be stored in the reference picture cache (634). In this manner, the video encoder (603) may store copies of reconstructed reference pictures locally that have common content as the reconstructed reference pictures that will be obtained by a far-end video decoder (absent transmission errors).

[0098] The predictor (635) may perform prediction searches for the coding engine (632). That is, for a new picture to be coded, the predictor (635) may search the reference picture memory (634) for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor (635) may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by search results obtained by the predictor (635), an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory (634).

[0099] The controller (650) may manage coding operations of the source coder (630), including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0100] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder (645). The entropy coder (645) translates the symbols as generated by the various functional units into a coded video sequence, by lossless compressing the symbols according to technologies such as Huffman coding, variable length coding, arithmetic coding, and so forth.

[0101] The transmitter (640) may buffer the coded video sequence(s) as created by the entropy coder (645) to prepare for transmission via a communication channel (660), which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter (640) may merge coded video data from the video coder (603) with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[0102] The controller (650) may manage operation of the video encoder (603). During coding, the controller (650) may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following picture types:

[0103] An Intra Picture (I picture) may be one that may be coded and decoded without using any other picture in the sequence as a source of prediction. Some video codecs allow for different types of intra pictures, including, for example Independent Decoder Refresh ("IDR") Pictures. A person skilled in the art is aware of those variants of I pictures and their respective applications and features.

[0104] A predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[0105] A bi-directionally predictive picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can use more than two reference pictures and associated metadata for the reconstruction of a single block.

[0106] Source pictures commonly may be subdivided spatially into a plurality of sample blocks (for example, blocks of 4x4, 8x8, 4x8, or 16x16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks' respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference picture. Blocks of B pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures.

[0107] The video encoder (603) may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video encoder (603) may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data, therefore, may conform to a syntax specified by the video coding technology or standard being used.

[0108] In an embodiment, the transmitter (640) may transmit additional data with the encoded video. The source coder (630) may include such data as part of the coded video sequence. Additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, SEI messages, VUI parameter set fragments, and so on.

[0109] A video may be captured as a plurality of source pictures (video pictures) in a temporal sequence. Intra-picture prediction (often abbreviated to intra prediction) makes use of spatial correlation in a given picture, and inter-picture prediction makes uses of the (temporal or

other) correlation between the pictures. In an example, a specific picture under encoding/decoding, which is referred to as a current picture, is partitioned into blocks. When a block in the current picture is similar to a reference block in a previously coded and still buffered reference picture in the video, the block in the current picture can be coded by a vector that is referred to as a motion vector. The motion vector points to the reference block in the reference picture, and can have a third dimension identifying the reference picture, in case multiple reference pictures are in use.

[0110] In some embodiments, a bi-prediction technique can be used in the inter-picture prediction. According to the bi-prediction technique, two reference pictures, such as a first reference picture and a second reference picture that are both prior in decoding order to the current picture in the video (but may be in the past and future, respectively, in display order) are used. A block in the current picture can be coded by a first motion vector that points to a first reference block in the first reference picture, and a second motion vector that points to a second reference block in the second reference picture. The block can be predicted by a combination of the first reference block and the second reference block.

[0111] Further, a merge mode technique can be used in the inter-picture prediction to improve coding efficiency.

[0112] According to some embodiments of the disclosure, predictions, such as inter-picture predictions and intra-picture predictions are performed in the unit of blocks. For example, according to the HEVC standard, a picture in a sequence of video pictures is partitioned into coding tree units (CTU) for compression, the CTUs in a picture have the same size, such as 64x64 pixels, 32x32 pixels, or 16x16 pixels. In general, a CTU includes three coding tree blocks (CTBs), which are one luma CTB and two chroma CTBs. Each CTU can be recursively quadtree split into one or multiple coding units (CUs). For example, a CTU of 64x64 pixels can be split into one CU of 64x64 pixels, or 4 CUs of 32x32 pixels, or 16 CUs of 16x16 pixels. In an example, each CU is analyzed to determine a prediction type for the CU, such as an inter prediction type or an intra prediction type. The CU is split into one or more prediction units (PUs) depending on the temporal and/or spatial predictability. Generally, each PU includes a luma prediction block (PB), and two chroma PBs. In an embodiment, a prediction operation in coding (encoding/decoding) is performed in the unit of a prediction block. Using a luma prediction block as an example of a prediction block, the prediction block includes a matrix of

values (e.g., luma values) for pixels, such as 8x8 pixels, 16x16 pixels, 8x16 pixels, 16x8 pixels, and the like.

[0113] FIG. 7 shows a diagram of a video encoder (703) according to another embodiment of the disclosure. The video encoder (703) is configured to receive a processing block (e.g., a prediction block) of sample values within a current video picture in a sequence of video pictures, and encode the processing block into a coded picture that is part of a coded video sequence. In an example, the video encoder (703) is used in the place of the video encoder (403) in the FIG. 4 example.

[0114] In an HEVC example, the video encoder (703) receives a matrix of sample values for a processing block, such as a prediction block of 8x8 samples, and the like. The video encoder (703) determines whether the processing block is best coded using intra mode, inter mode, or bi-prediction mode using, for example, rate-distortion optimization. When the processing block is to be coded in intra mode, the video encoder (703) may use an intra prediction technique to encode the processing block into the coded picture; and when the processing block is to be coded in inter mode or bi-prediction mode, the video encoder (703) may use an inter prediction or bi-prediction technique, respectively, to encode the processing block into the coded picture. In certain video coding technologies, merge mode can be an inter picture prediction submode where the motion vector is derived from one or more motion vector predictors without the benefit of a coded motion vector component outside the predictors. In certain other video coding technologies, a motion vector component applicable to the subject block may be present. In an example, the video encoder (703) includes other components, such as a mode decision module (not shown) to determine the mode of the processing blocks.

[0115] In the FIG. 7 example, the video encoder (703) includes the inter encoder (730), an intra encoder (722), a residue calculator (723), a switch (726), a residue encoder (724), a general controller (721), and an entropy encoder (725) coupled together as shown in FIG. 7.

[0116] The inter encoder (730) is configured to receive the samples of the current block (e.g., a processing block), compare the block to one or more reference blocks in reference pictures (e.g., blocks in previous pictures and later pictures), generate inter prediction information (e.g., description of redundant information according to inter encoding technique, motion vectors, merge mode information), and calculate inter prediction results (e.g., predicted block) based on the inter prediction information using any suitable technique. In some examples,

the reference pictures are decoded reference pictures that are decoded based on the encoded video information.

[0117] The intra encoder (722) is configured to receive the samples of the current block (e.g., a processing block), in some cases compare the block to blocks already coded in the same picture, generate quantized coefficients after transform, and in some cases also intra prediction information (e.g., an intra prediction direction information according to one or more intra encoding techniques). In an example, the intra encoder (722) also calculates intra prediction results (e.g., predicted block) based on the intra prediction information and reference blocks in the same picture.

[0118] The general controller (721) is configured to determine general control data and control other components of the video encoder (703) based on the general control data. In an example, the general controller (721) determines the mode of the block, and provides a control signal to the switch (726) based on the mode. For example, when the mode is the intra mode, the general controller (721) controls the switch (726) to select the intra mode result for use by the residue calculator (723), and controls the entropy encoder (725) to select the intra prediction information and include the intra prediction information in the bitstream; and when the mode is the inter mode, the general controller (721) controls the switch (726) to select the inter prediction result for use by the residue calculator (723), and controls the entropy encoder (725) to select the inter prediction information and include the inter prediction information in the bitstream.

[0119] The residue calculator (723) is configured to calculate a difference (residue data) between the received block and prediction results selected from the intra encoder (722) or the inter encoder (730). The residue encoder (724) is configured to operate based on the residue data to encode the residue data to generate the transform coefficients. In an example, the residue encoder (724) is configured to convert the residue data from a spatial domain to a frequency domain, and generate the transform coefficients. The transform coefficients are then subject to quantization processing to obtain quantized transform coefficients. In various embodiments, the video encoder (703) also includes a residue decoder (728). The residue decoder (728) is configured to perform inverse-transform, and generate the decoded residue data. The decoded residue data can be suitably used by the intra encoder (722) and the inter encoder (730). For example, the inter encoder (730) can generate decoded blocks based on the decoded residue data and inter prediction information, and the intra encoder (722) can generate decoded blocks based on the decoded residue data and the intra prediction information. The decoded blocks are

suitably processed to generate decoded pictures and the decoded pictures can be buffered in a memory circuit (not shown) and used as reference pictures in some examples.

[0120] The entropy encoder (725) is configured to format the bitstream to include the encoded block. The entropy encoder (725) is configured to include various information according to a suitable standard, such as the HEVC standard. In an example, the entropy encoder (725) is configured to include the general control data, the selected prediction information (e.g., intra prediction information or inter prediction information), the residue information, and other suitable information in the bitstream. Note that, according to the disclosed subject matter, when coding a block in the merge submode of either inter mode or bi-prediction mode, there is no residue information.

[0121] FIG. 8 shows a diagram of a video decoder (810) according to another embodiment of the disclosure. The video decoder (810) is configured to receive coded pictures that are part of a coded video sequence, and decode the coded pictures to generate reconstructed pictures. In an example, the video decoder (810) is used in the place of the video decoder (410) in the FIG. 4 example.

[0122] In the FIG. 8 example, the video decoder (810) includes an entropy decoder (871), an inter decoder (880), a residue decoder (873), a reconstruction module (874), and an intra decoder (872) coupled together as shown in FIG. 8.

[0123] The entropy decoder (871) can be configured to reconstruct, from the coded picture, certain symbols that represent the syntax elements of which the coded picture is made up. Such symbols can include, for example, the mode in which a block is coded (such as, for example, intra mode, inter mode, bi-predicted mode, the latter two in merge submode or another submode), prediction information (such as, for example, intra prediction information or inter prediction information) that can identify certain sample or metadata that is used for prediction by the intra decoder (872) or the inter decoder (880), respectively, residual information in the form of, for example, quantized transform coefficients, and the like. In an example, when the prediction mode is inter or bi-predicted mode, the inter prediction information is provided to the inter decoder (880); and when the prediction type is the intra prediction type, the intra prediction information is provided to the intra decoder (872). The residual information can be subject to inverse quantization and is provided to the residue decoder (873).

[0124] The inter decoder (880) is configured to receive the inter prediction information, and generate inter prediction results based on the inter prediction information.

[0125] The intra decoder (872) is configured to receive the intra prediction information, and generate prediction results based on the intra prediction information.

[0126] The residue decoder (873) is configured to perform inverse quantization to extract de-quantized transform coefficients, and process the de-quantized transform coefficients to convert the residual from the frequency domain to the spatial domain. The residue decoder (873) may also require certain control information (to include the Quantizer Parameter (QP)), and that information may be provided by the entropy decoder (871) (data path not depicted as this may be low volume control information only).

[0127] The reconstruction module (874) is configured to combine, in the spatial domain, the residual as output by the residue decoder (873) and the prediction results (as output by the inter or intra prediction modules as the case may be) to form a reconstructed block, that may be part of the reconstructed picture, which in turn may be part of the reconstructed video. It is noted that other suitable operations, such as a deblocking operation and the like, can be performed to improve the visual quality.

[0128] It is noted that the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using any suitable technique. In an embodiment, the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using one or more integrated circuits. In another embodiment, the video encoders (403), (603), and (603), and the video decoders (410), (510), and (810) can be implemented using one or more processors that execute software instructions.

[0129] Aspects of the disclosure provide filtering techniques for video coding/decoding.

[0130] An adaptive loop filter (ALF) with block-based filter adaption can be applied by encoders/decoders to reduce artifacts. For a luma component, one of a plurality of filters (e.g., 25 filters) can be selected for a 4×4 luma block, for example, based on a direction and activity of local gradients.

[0131] An ALF can have any suitable shape and size. Referring to FIG. 9, ALFs (910)-(911) have a diamond shape, such as a 5×5 diamond-shape for the ALF (910) and a 7×7 diamond-shape for the ALF (911). In the ALF (910), elements (920)-(932) form a diamond shape and can be used in the filtering process. Seven values (e.g., C0-C6) can be used for the elements (920)-(932). In the ALF (911), elements (940)-(964) forms a diamond shape and can be used in the filtering process. Thirteen values (e.g., C0-C12) can be used for the elements (940)-(964).

[0132] Referring to FIG. 9, in some examples, the two ALFs (910)-(911) with the diamond filter shape are used. The 5×5 diamond-shaped filter (910) can be applied for chroma components (e.g., chroma blocks, chroma CBs), and the 7×7 diamond-shaped filter (911) can be applied for a luma component (e.g., a luma block, a luma CB). Other suitable shape(s) and size(s) can be used in the ALF. For example, a 9x9 diamond-shaped filter can be used.

[0133] Filter coefficients at locations indicated by the values (e.g., C0-C6 in (910) or C0-C12 in (920)) can be non-zero. Further, when the ALF includes a clipping function, clipping values at the locations can be non-zero.

[0134] For block classification of a luma component, a 4 × 4 block (or luma block, luma CB) can be categorized or classified as one of multiple (e.g., 25) classes. A classification index C can be derived based on a directionality parameter D and a quantized value $\hat{A}$ of an activity value A using Eq. (1).

$$C = 5D + \hat{A} \quad \text{Eq. (1)}$$

To calculate the directionality parameter D and the quantized value $\hat{A}$, gradients g_v, g_h, g_{d1}, and g_{d2} of a vertical, a horizontal, and two diagonal directions (e.g., d1 and d2), respectively, can be calculated using 1-D Laplacian as follows.

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

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

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

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

where indices i and j refer to coordinates of an upper left sample within the 4 × 4 block and R(k,l) indicates a reconstructed sample at a coordinate (k,l). The directions (e.g., d1 and d2) can refer to 2 diagonal directions.

[0135] To reduce complexity of the block classification described above, a subsampled 1-D Laplacian calculation can be applied. FIGs. 10A-10D show examples of subsampled positions used for calculating the gradients g_v, g_h, g_{d1}, and g_{d2} of the vertical (FIG. 10A), the horizontal (FIG. 10B), and the two diagonal directions d1 (FIG. 10C) and d2 (FIG. 10D), respectively. The same subsampled positions can be used for gradient calculation of the different directions. In FIG. 10A, labels 'V' show the subsampled positions to calculate the

vertical gradient g_v. In FIG. 10B, labels 'H' show the subsampled positions to calculate the horizontal gradient g_h. In FIG. 10C, labels 'D1' show the subsampled positions to calculate the d1 diagonal gradient g_{d1}. In FIG. 10D, labels 'D2' show the subsampled positions to calculate the d2 diagonal gradient g_{d2}.

[0136] A maximum value $g_{h,v}^{max}$ and a minimum value $g_{h,v}^{min}$ of the gradients of horizontal and vertical directions g_v and g_h can be set as:

$$g_{h,v}^{max} = max(g_h, g_v), \quad g_{h,v}^{min} = min(g_h, g_v) \qquad \text{Eq. (6)}$$

A maximum value $g_{d1,d2}^{max}$ and a minimum value $g_{d1,d2}^{min}$ of the gradients of two diagonal directions g_{d1} and g_{d2} can be set as:

$$g_{d1,d2}^{max} = max(g_{d1}, g_{d2}), \quad g_{d1,d2}^{min} = min(g_{d1}, g_{d2}) \qquad \text{Eq. (7)}$$

The directionality parameter D can be derived based on the above values and two thresholds t_1 and t_2 as below.

Step 1. If (1) $g_{h,v}^{max} \leq t_1 \cdot g_{h,v}^{min}$ and (2) $g_{d1,d2}^{max} \leq t_1 \cdot g_{d1,d2}^{min}$ are true, D is set to 0.

Step 2. If $g_{h,v}^{max}/g_{h,v}^{min} > g_{d1,d2}^{max}/g_{d1,d2}^{min}$, continue to Step 3; otherwise continue to 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 $g_{d1,d2}^{max} > t_2 \cdot g_{d1,d2}^{min}$, D is set to 4; otherwise D is set to 3.

[0137] The activity value A can be calculated as:

$$A = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} (V_{k,l} + H_{k,l}) = g_v + g_h \qquad \text{Eq. (8)}$$

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

[0138] For chroma components in a picture, no block classification is applied, and thus a single set of ALF coefficients can be applied for each chroma component.

[0139] Geometric transformations can be applied to filter coefficients and corresponding filter clipping values (also referred to as clipping values). Before filtering a block (e.g., a 4×4 luma block), geometric transformations such as rotation or diagonal and vertical flipping can be applied to the filter coefficients f(k, l) and the corresponding filter clipping values c(k, l), for example, depending on gradient values (e.g., g_v, g_h, g_{d1}, and/or g_{d2}) calculated for the block. The geometric transformations applied to the filter coefficients f(k, l) and the corresponding filter clipping values c(k, l) can be equivalent to applying the geometric transformations to samples in a region supported by the filter. The geometric transformations can make different blocks to which an ALF is applied more similar by aligning the respective directionality.

[0140] Three geometric transformations, including a diagonal flip, a vertical flip, and a rotation can be performed as described by Eqs. (9)-(11), respectively.

$$f_D(k,l) = f(l,k),\ c_D(k,l) = c(l,k), \quad \text{Eq. (9)}$$

$$f_V(k,l) = f(k,K-l-1),\ c_V(k,l) = c(k,K-l-1) \quad \text{Eq. (10)}$$

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

where K is a size of the ALF or the filter, and $0 \leq k,l \leq K-1$ are coordinates of coefficients. For example, a location (0,0) is at an upper left corner and a location $(K-1,K-1)$ is at a lower right corner of the filter f or a clipping value matrix (or clipping matrix) c. The transformations can be applied to the filter coefficients f (k, l) and the clipping values c(k, l) depending on the gradient values calculated for the block. An example of a relationship between the transformation and the four gradients are summarized in Table 1.

Table 1: Mapping of the gradient calculated for a block and the transformation

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

[0141] In some embodiments, ALF filter parameters are signaled in an Adaptation Parameter Set (APS) for a picture. In the APS, one or more sets (e.g., up to 25 sets) of luma filter coefficients and clipping value indexes can be signaled. In an example, a set of the one or more sets can include luma filter coefficients and one or more clipping value indexes. One or more sets (e.g., up to 8 sets) of chroma filter coefficients and clipping value indexes can be signaled. To reduce signaling overhead, filter coefficients of different classifications (e.g., having different classification indices) for luma components can be merged. In a slice header, indices of the APSs used for a current slice can be signaled.

[0142] In an embodiment, a clipping value index (also referred to as clipping index) can be decoded from the APS. The clipping value index can be used to determine a corresponding clipping value, for example, based on a relationship between the clipping value index and the corresponding clipping value. The relationship can be pre-defined and stored in a decoder. In an example, the relationship is described by a table, such as a luma table (e.g., used for a luma CB) of the clipping value index and the corresponding clipping value, a chroma table (e.g., used for a chroma CB) of the clipping value index and the corresponding clipping value. The clipping

value can be dependent of a bit depth B. The bit depth B can refer to an internal bit depth, a bit depth of reconstructed samples in a CB to be filtered, or the like. In some examples, a table (e.g., a luma table, a chroma table) is obtained using Eq. (12).

$$\text{AlfClip} = \left\{\text{round}\left(2^{B\frac{N-n+1}{N}}\right) \text{ for } n \in [1..N]\right\}, \quad \text{Eq. (12)}$$

where AlfClip is the clipping value, B is the bit depth (e.g., bitDepth), N (e.g., $N = 4$) is a number of allowed clipping values, and (n-1) is the clipping value index (also referred to as clipping index or clipIdx). Table 2 shows an example of a table obtained using Eq. (12) with $N = 4$. The clipping index (n-1) can be 0, 1, 2, and 3 in Table 2, and n can be 1, 2, 3, and 4, respectively. Table 2 can be used for luma blocks or chroma blocks.

Table 2 – AlfClip can depend on the bit depth B and clipIdx

bitDepth	**clipIdx**			
	0	**1**	**2**	**3**
8	255	64	16	4
9	511	108	23	5
10	1023	181	32	6
11	2047	304	45	7
12	4095	512	64	8
13	8191	861	91	10
14	16383	1448	128	11
15	32767	2435	181	13
16	65535	4096	256	16

[0143] In a slice header for a current slice, one or more APS indices (e.g., up to 7 APS indices) can be signaled to specify luma filter sets that can be used for the current slice. The filtering process can be controlled at one or more suitable levels, such as a picture level, a slice level, a CTB level, and/or the like. In an embodiment, the filtering process can be further controlled at a CTB level. A flag can be signaled to indicate whether the ALF is applied to a luma CTB. The luma CTB can choose a filter set among a plurality of fixed filter sets (e.g., 16 fixed filter sets) and the filter set(s) (also referred to as signaled filter set(s)) that are signaled in the APSs. A filter set index can be signaled for the luma CTB to indicate the filter set (e.g., the filter set among the plurality of fixed filter sets and the signaled filter set(s)) to be applied. The

plurality of fixed filter sets can be pre-defined and hard-coded in an encoder and a decoder, and can be referred to as pre-defined filter sets.

[0144] For a chroma component, an APS index can be signaled in the slice header to indicate the chroma filter sets to be used for the current slice. At the CTB level, a filter set index can be signaled for each chroma CTB if there is more than one chroma filter set in the APS.

[0145] The filter coefficients can be quantized with a norm equal to 128. In order to decrease the multiplication complexity, a bitstream conformance can be applied so that the coefficient value of the non-central position can be in a range of −27 to 27 − 1, inclusive. In an example, the central position coefficient is not signaled in the bitstream and can be considered as equal to 128.

[0146] In some embodiments, the syntaxes and semantics of clipping index and clipping values are defined as follows:

alf_luma_clip_idx[sfIdx][j] can be used to specify the clipping index of the clipping value to use before multiplying by the j-th coefficient of the signaled luma filter indicated by sfIdx. A requirement of bitstream conformance can include that the values of alf_luma_clip_idx[sfIdx][j] with sfIdx = 0 to alf_luma_num_filters_signalled_minus1 and j = 0 to 11 shall be in the range of 0 to 3, inclusive.

The luma filter clipping values AlfClipL[adaptation_parameter_set_id] with elements AlfClipL[adaptation_parameter_set_id][filtIdx][j], with filtIdx = 0 to NumAlfFilters − 1 and j = 0 to 11 can be derived as specified in Table 2 depending on bitDepth set equal to BitDepthY and clipIdx set equal to alf_luma_clip_idx[alf_luma_coeff_delta_idx[filtIdx]][j].

alf_chroma_clip_idx[altIdx][j] can be used to specify the clipping index of the clipping value to use before multiplying by the j-th coefficient of the alternative chroma filter with index altIdx. A requirement of bitstream conformance can include that the values of alf_chroma_clip_idx[altIdx][j] with altIdx = 0 to alf_chroma_num_alt_filters_minus1, j = 0 to 5 shall be in the range of 0 to 3, inclusive.

The chroma filter clipping values AlfClipC[adaptation_parameter_set_id][altIdx] with elements AlfClipC[adaptation_parameter_set_id][altIdx][j], with altIdx = 0 to alf_chroma_num_alt_filters_minus1, j = 0 to 5 can be derived as specified in Table 2 depending on bitDepth set equal to BitDepthC and clipIdx set equal to alf_chroma_clip_idx[altIdx][j].

[0147] In an embodiment, the filtering process can be described as below. At a decoder side, when the ALF is enabled for a CTB, a sample R(i,j) within a CU (or CB) can be filtered, resulting in a filtered sample value R'(i,j) as shown below using Eq. (13). In an example, each sample in the CU is filtered.

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

where f(k,l) denotes the decoded filter coefficients, K(x, y) is a clipping function, and c(k, l) denotes the decoded clipping parameters (or clipping values). The variables k and l can vary between -L/2 and L/2 where L denotes a filter length. The clipping function K(x, y) = min (y, max(-y, x)) corresponds to a clipping function Clip3 (-y, y, x). By incorporating the clipping function K(x, y), the loop filtering method (e.g., ALF) becomes a non-linear process, and can be referred to a nonlinear ALF.

[0148] In the nonlinear ALF, multiple sets of clipping values can be provided in Table 3. In an example, a luma set includes four clipping values {1024, 181, 32, 6}, and a chroma set includes 4 clipping values {1024, 161, 25, 4}. The four clipping values in the luma set can be selected by approximately equally splitting, in a logarithmic domain, a full range (e.g., 1024) of the sample values (coded on 10 bits) for a luma block. The range can be from 4 to 1024 for the chroma set.

Table 3 – Examples of clipping values

	INTRA/INTER tile group
LUMA	{ 1024, 181, 32, 6 }
CHROMA	{ 1024, 161, 25, 4 }

[0149] The selected clipping values can be coded in an "alf_data" syntax element as follows: a suitable encoding scheme (e.g., a Golomb encoding scheme) can be used to encode a clipping index corresponding to the selected clipping value such as shown in Table 3. The encoding scheme can be the same encoding scheme used for encoding the filter set index.

[0150] In an embodiment, a virtual boundary filtering process can be used to reduce a line buffer requirement of the ALF. Accordingly, modified block classification and filtering can be employed for samples near CTU boundaries (e.g., a horizontal CTU boundary). A virtual boundary (1130) can be defined as a line by shifting a horizontal CTU boundary (1120) by

"$N_{samples}$" samples, as shown in FIG. 11A, where $N_{samples}$ can be a positive integer. In an example, $N_{samples}$ is equal to 4 for a luma component, and $N_{samples}$ is equal to 2 for a chroma component.

[0151] Referring to FIG. 11A, a modified block classification can be applied for a luma component. In an example, for the 1D Laplacian gradient calculation of a 4x4 block (1110) above the virtual boundary (1130), only samples above the virtual boundary (1130) are used. Similarly, referring to FIG. 11B, for a 1D Laplacian gradient calculation of a 4x4 block (1111) below a virtual boundary (1131) that is shifted from a CTU boundary (1121), only samples below the virtual boundary (1131) are used. The quantization of an activity value A can be accordingly scaled by taking into account a reduced number of samples used in the 1D Laplacian gradient calculation.

[0152] For a filtering processing, a symmetric padding operation at virtual boundaries can be used for both a luma component and a chroma component. FIGs. 12A-12F illustrate examples of such modified ALF filtering for a luma component at virtual boundaries. When a sample being filtered is located below a virtual boundary, neighboring samples that are located above the virtual boundary can be padded. When a sample being filtered is located above a virtual boundary, neighboring samples that are located below the virtual boundary can be padded. Referring to FIG. 12A, a neighboring sample C0 can be padded with a sample C2 that is located below a virtual boundary (1210). Referring to FIG. 12B, a neighboring sample C0 can be padded with a sample C2 that is located above a virtual boundary (1220). Referring to FIG. 12C, neighboring samples C1-C3 can be padded with samples C5-C7, respectively, that are located below a virtual boundary (1230). Referring to FIG. 12D, neighboring samples C1-C3 can be padded with samples C5-C7, respectively, that are located above a virtual boundary (1240). Referring to FIG. 12E, neighboring samples C4-C8 can be padded with samples C10, C11, C12, C11, and C10, respectively, that are located below a virtual boundary (1250). Referring to FIG. 12F, neighboring samples C4-C8 can be padded with samples C10, C11, C12, C11, and C10, respectively, that are located above a virtual boundary (1260).

[0153] In some examples, the above description can be suitably adapted when sample(s) and neighboring sample(s) are located to the left (or to the right) and to the right (or to the left) of a virtual boundary.

[0154] According to an aspect of the disclosure, in order to improve coding efficiency, pictures can be partitioned based on filtering process. In some examples, a CTU is also referred

to as a largest coding unit (LCU). In an example, the CTU or LCU can have a size of 64×64 pixels. In some embodiments, LCU-Aligned picture quadtree splitting can be used for the filtering based partition. In some examples, the coding unit synchronous picture quadtree-based adaptive loop filter can be used. For example, the luma picture can be split into several multi-level quadtree partitions, and each partition boundary is aligned to the boundaries of the LCUs. Each partition has its own filtering process and thus be referred to as a filter unit (FU).

[0155] In some examples, a 2-pass encoding flow can be used. At a first pass of the 2-pass encoding flow, the quadtree split pattern of the picture and the best filter of each FU can be determined. In some embodiment, the determination of the quadtree split pattern of the picture and the determination of the best filters for FUs are based on filtering distortions. The filtering distortions can be estimated by fast filtering distortion estimation (FFDE) technique during the determination process. The picture is partitioned using quadtree partition. According to the determined quadtree split pattern and the selected filters of all FUs, the reconstructed picture can be filtered.

[0156] At a second pass of the 2-pass encoding flow, the CU synchronous ALF on/off control is performed. According to the ALF on/off results, the first filtered picture is partially recovered by the reconstructed picture.

[0157] Specifically, in some examples, a top-down splitting strategy is adopted to divide a picture into multi-level quadtree partitions by using a rate-distortion criterion. Each partition is called a filter unit (FU). The splitting process aligns quadtree partitions with LCU boundaries. The encoding order of FUs follows the z-scan order.

[0158] FIG. 13 shows a partition example according to some embodiments of the disclosure. In the FIG. 13 example, a picture (1300) is split into 10 FUs, and the encoding order is FU0, FU1, FU2, FU3, FU4, FU5, FU6, FU7, FU8, and FU9.

[0159] FIG. 14 shows a quadtree split pattern (1400) for the picture (1300). In the FIG. 14 example, split flags are used to indicate the picture partition pattern. For example, "1" indicates a quadtree partition is performed on the block; and "0" indicates that the block is not further partitioned. In some examples, a minimum size FU has the LCU size, and no split flag is needed for the minimum size FU. The split flags are encoded and transmitted in z-order as shown in FIG. 14.

[0160] In some examples, the filter of each FU is selected from two filter sets based on the rate-distortion criterion. The first set has 1/2-symmetric square-shaped and rhombus-shaped

filters derived for the current FU. The second set comes from time-delayed filter buffers; the time-delayed filter buffers store the filters previously derived for FUs of prior pictures. The filter with the minimum rate-distortion cost of these two sets can be chosen for the current FU. Similarly, if the current FU is not the smallest FU and can be further split into 4 children FUs, the rate-distortion costs of the 4 children FUs are calculated. By comparing the rate-distortion cost of the split and non-split cases recursively, the picture quadtree split pattern can be decided.

[0161] In some examples, a maximum quadtree split level may be used to limit the maximum number of FUs. In an example, when the maximum quadtree split level is 2, the maximum number of FUs is 16. Further, during the quadtree split determination, the correlation values for deriving Wiener coefficients of the 16 FUs at the bottom quadtree level (smallest FUs) can be reused. The rest FUs can derive their Wiener filters from the correlations of the 16 FUs at the bottom quadtree level. Therefore, in the example, only one frame buffer access is performed for deriving the filter coefficients of all FUs.

[0162] After the quadtree split pattern is decided, to further reduce the filtering distortion, the CU synchronous ALF on/off control can be performed. By comparing the filtering distortion and non-filtering distortion at each leaf CU, the leaf CU can explicitly switch ALF on/off in its local region. In some examples, the coding efficiency may be further improved by redesigning the filter coefficients according to the ALF on/off results.

[0163] A cross-component filtering process can apply cross-component filters, such as cross-component adaptive loop filters (CC-ALFs). The cross-component filter can use luma sample values of a luma component (e.g., a luma CB) to refine a chroma component (e.g., a chroma CB corresponding to the luma CB). In an example, the luma CB and the chroma CB are included in a CU.

[0164] FIG. 15 shows cross-component filters (e.g., CC-ALFs) used to generate chroma components according to an embodiment of the disclosure. In some examples, FIG. 15 shows filtering processes for a first chroma component (e.g., a first chroma CB), a second chroma component (e.g., a second chroma CB), and a luma component (e.g., a luma CB). The luma component can be filtered by a sample adaptive offset (SAO) filter (1510) to generate a SAO filtered luma component (1541). The SAO filtered luma component (1541) can be further filtered by an ALF luma filter (1516) to become a filtered luma CB (1561) (e.g., 'Y').

[0165] The first chroma component can be filtered by a SAO filter (1512) and an ALF chroma filter (1518) to generate a first intermediate component (1552). Further, the SAO filtered

luma component (1541) can be filtered by a cross-component filter (e.g., CC-ALF) (1521) for the first chroma component to generate a second intermediate component (1542). Subsequently, a filtered first chroma component (1562) (e.g., 'Cb') can be generated based on at least one of the second intermediate component (1542) and the first intermediate component (1552). In an example, the filtered first chroma component (1562) (e.g., 'Cb') can be generated by combining the second intermediate component (1542) and the first intermediate component (1552) with an adder (1522). The cross-component adaptive loop filtering process for the first chroma component can include a step performed by the CC-ALF (1521) and a step performed by, for example, the adder (1522).

[0166] The above description can be adapted to the second chroma component. The second chroma component can be filtered by a SAO filter (1514) and the ALF chroma filter (1518) to generate a third intermediate component (1553). Further, the SAO filtered luma component (1541) can be filtered by a cross-component filter (e.g., a CC-ALF) (1531) for the second chroma component to generate a fourth intermediate component (1543). Subsequently, a filtered second chroma component (1563) (e.g., 'Cr') can be generated based on at least one of the fourth intermediate component (1543) and the third intermediate component (1553). In an example, the filtered second chroma component (1563) (e.g., 'Cr') can be generated by combining the fourth intermediate component (1543) and the third intermediate component (1553) with an adder (1532). In an example, the cross-component adaptive loop filtering process for the second chroma component can include a step performed by the CC-ALF (1531) and a step performed by, for example, the adder (1532).

[0167] A cross-component filter (e.g., the CC-ALF (1521), the CC-ALF (1531)) can operate by applying a linear filter having any suitable filter shape to the luma component (or a luma channel) to refine each chroma component (e.g., the first chroma component, the second chroma component).

[0168] FIG. 16 shows an example of a filter (1600) according to an embodiment of the disclosure. The filter (1600) can include non-zero filter coefficients and zero filter coefficients. The filter (1600) has a diamond shape (1620) formed by filter coefficients (1610) (indicated by circles having black fill). In an example, the non-zero filter coefficients in the filter (1600) are included in the filter coefficients (1610), and filter coefficients not included in the filter coefficients (1610) are zero. Thus, the non-zero filter coefficients in the filter (1600) are included in the diamond shape (1620), and the filter coefficients not included in the diamond

shape (1620) are zero. In an example, a number of the filter coefficients of the filter (1600) is equal to a number of the filter coefficients (1610), which is 18 in the example shown in FIG. 16.

[0169] The CC-ALF can include any suitable filter coefficients (also referred to as the CC-ALF filter coefficients). Referring back to FIG. 15, the CC-ALF (1521) and the CC-ALF (1531) can have a same filter shape, such as the diamond shape (1620) shown in FIG. 16, and a same number of filter coefficients. In an example, values of the filter coefficients in the CC-ALF (1521) are different from values of the filter coefficients in the CC-ALF (1531).

[0170] In general, filter coefficients (e.g., non-zero filter coefficients) in a CC-ALF can be transmitted, for example, in the APS. In an example, the filter coefficients can be scaled by a factor (e.g., 2^{10}) and can be rounded for a fixed point representation. Application of a CC-ALF can be controlled on a variable block size and signaled by a context-coded flag (e.g., a CC-ALF enabling flag) received for each block of samples. The context-coded flag, such as the CC-ALF enabling flag, can be signaled at any suitable level, such as a block level. The block size along with the CC-ALF enabling flag can be received at a slice-level for each chroma component. In some examples, block sizes (in chroma samples) 16x16, 32x32, and 64x64 can be supported.

[0171] FIG. 17 shows a syntax example for CC-ALF according to some embodiments of the disclosure. In the FIG. 17 example,
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is an index to indicate whether a cross component Cb filter is used and an index of the cross component Cb filter if used. For example, when
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is equal to 0, the cross component Cb filter is not applied to block of Cb colour component samples at luma location (xCtb, yCtb); when
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is not equal to 0,
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is an index for a filter to be applied. For example,
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]-th cross component Cb filter is applied to the block of Cb colour component samples at luma location (xCtb, yCtb)

[0172] Further, in the FIG. 17 example,
alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is used to

indicate whehter a cross component Cr filter is used and index of the cross component Cr filter is used. For exmaple, when
alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is equal to 0, the cross component Cr filter is not applied to block of Cr colour component samples at luma location (xCtb, yCtb); when
alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is not equal to 0,
alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] is the index of the cross component Cr filter. For example,
alf_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]-th cross component Cr filter can be applied to the block of Cr colour component samples at luma location (xCtb, yCtb)

[0173] In some examples, chroma subsampling techniques are used, thus a number of samples in each of the chroma block(s) can be less than a number of samples in the luma block. A chroma subsampling format (also referred to as a chroma subsampling format, e.g., specified by chroma_format_idc) can indicate a chroma horizontal subsampling factor (e.g., SubWidthC) and a chroma vertical subsampling factor (e.g., SubHeightC) between each of the chroma block(s) and the corresponding luma block. In an example, the chroma subsampling format is 4:2:0, and thus the chroma horizontal subsampling factor (e.g., SubWidthC) and the chroma vertical subsampling factor (e.g., SubHeightC) are 2, as shown in FIGs. 18A-18B. In an example, the chroma subsampling format is 4:2:2, and thus the chroma horizontal subsampling factor (e.g., SubWidthC) is 2, and the chroma vertical subsampling factor (e.g., SubHeightC) is 1. In an example, the chroma subsampling format is 4:4:4, and thus the chroma horizontal subsampling factor (e.g., SubWidthC) and the chroma vertical subsampling factor (e.g., SubHeightC) are 1. A chroma sample type (also referred to as a chroma sample position) can indicate a relative position of a chroma sample in the chroma block with respect to at least one corresponding luma sample in the luma block.

[0174] FIGs. 18A-18B show exemplary locations of chroma samples relative to luma samples according to embodiments of the disclosure. Referring to FIG. 18A, the luma samples (1801) are located in rows (1811)-(1818). The luma samples (1801) shown in FIG. 18A can represent a portion of a picture. In an example, a luma block (e.g., a luma CB) includes the luma samples (1801). The luma block can correspond to two chroma blocks having the chroma

subsampling format of 4:2:0. In an example, each chroma block includes chroma samples (1803). Each chroma sample (e.g., the chroma sample (1803(1)) corresponds to four luma samples (e.g., the luma samples (1801(1))-(1801(4)). In an example, the four luma samples are the top-left sample (1801(1)), the top-right sample (1801(2)), the bottom-left sample (1801(3)), and the bottom-right sample (1801(4)). The chroma sample (e.g., (1803(1))) is located at a left center position that is between the top-left sample (1801(1)) and the bottom-left sample (1801(3)), and a chroma sample type of the chroma block having the chroma samples (1803) can be referred to as a chroma sample type 0. The chroma sample type 0 indicates a relative position 0 corresponding to the left center position in the middle of the top-left sample (1801(1)) and the bottom-left sample (1801(3)). The four luma samples (e.g., (1801(1))-(1801(4))) can be referred to as neighboring luma samples of the chroma sample (1803)(1).

[0175] In an example, each chroma block includes chroma samples (1804). The above description with reference to the chroma samples (1803) can be adapted to the chroma samples (1804), and thus detailed descriptions can be omitted for purposes of brevity. Each of the chroma samples (1804) can be located at a center position of four corresponding luma samples, and a chroma sample type of the chroma block having the chroma samples (1804) can be referred to as a chroma sample type 1. The chroma sample type 1 indicates a relative position 1 corresponding to the center position of the four luma samples (e.g., (1801(1))-(1801(4))). For example, one of the chroma samples (1804) can be located at a center portion of the luma samples (1801(1))-(1801(4)).

[0176] In an example, each chroma block includes chroma samples (1805). Each of the chroma samples (1805) can be located at a top left position that is co-located with the top-left sample of the four corresponding luma samples (1801), and a chroma sample type of the chroma block having the chroma samples (1805) can be referred to as a chroma sample type 2. Accordingly, each of the chroma samples (1805) is co-located with the top left sample of the four luma samples (1801) corresponding to the respective chroma sample. The chroma sample type 2 indicates a relative position 2 corresponding to the top left position of the four luma samples (1801). For example, one of the chroma samples (1805) can be located at a top left position of the luma samples (1801(1))-(1801(4)).

[0177] In an example, each chroma block includes chroma samples (1806). Each of the chroma samples (1806) can be located at a top center position between a corresponding top-left sample and a corresponding top-right sample, and a chroma sample type of the chroma block

having the chroma samples (1806) can be referred to as a chroma sample type 3. The chroma sample type 3 indicates a relative position 3 corresponding to the top center position between the top-left sample (and the top-right sample. For example, one of the chroma samples (1806) can be located at a top center position of the luma samples (1801(1))-(1801(4)).

[0178] In an example, each chroma block includes chroma samples (1807). Each of the chroma samples (1807) can be located at a bottom left position that is co-located with the bottom-left sample of the four corresponding luma samples (1801), and a chroma sample type of the chroma block having the chroma samples (1807) can be referred to as a chroma sample type 4. Accordingly, each of the chroma samples (1807) is co-located with the bottom left sample of the four luma samples (1801) corresponding to the respective chroma sample. The chroma sample type 4 indicates a relative position 4 corresponding to the bottom left position of the four luma samples (1801). For example, one of the chroma samples (1807) can be located at a bottom left position of the luma samples (1801(1))-(1801(4)).

[0179] In an example, each chroma block includes chroma samples (1808). Each of the chroma samples (1808) is located at a bottom center position between the bottom –left sample and the bottom -right sample, and a chroma sample type of the chroma block having the chroma samples (1808) can be referred to as a chroma sample type 5. The chroma sample type 5 indicates a relative position 5 corresponding to the bottom center position between the bottom -left sample and the bottom -right sample of the four luma samples (1801). For example, one of the chroma samples (1808) can be located between the bottom-left sample and the bottom-right sample of the luma samples (1801(1))-(1801(4)).

[0180] In general, any suitable chroma sample type can be used for a chroma subsampling format. The chroma sample types 0-5 are exemplary chroma sample types described with the chroma subsampling format 4:2:0. Additional chroma sample types may be used for the chroma subsampling format 4:2:0. Further, other chroma sample types and/or variations of the chroma sample types 0-5 can be used for other chroma subsampling formats, such as 4:2:2, 4:4:4, or the like. In an example, a chroma sample type combining the chroma samples (1805) and (1807) is used for the chroma subsampling format 4:2:2.

[0181] In an example, the luma block is considered to have alternating rows, such as the rows (1811)-(1812) that include the top two samples (e.g., (1801(1))-(180)(2))) of the four luma samples (e.g., (1801(1))-(1801(4))) and the bottom two samples (e.g., (1801(3))-(1801(4))) of the four luma samples (e.g., (1801(1)-(1801(4))), respectively. Accordingly, the rows (1811),

(1813), (1815), and (1817) can be referred to as current rows (also referred to as a top field), and the rows (1812), (1814), (1816), and (1818) can be referred to as next rows (also referred to as a bottom field). The four luma samples (e.g., (1801(1))-(1801(4))) are located at the current row (e.g., (1811)) and the next row (e.g., (1812)). The relative positions 2-3 are located in the current rows, the relative positions 0-1 are located between each current row and the respective next row, and the relative positions 4-5 are located in the next rows.

[0182] The chroma samples (1803), (1804), (1805), (1806), (1807), or (1808) are located in rows (1851)-(1854) in each chroma block. Specific locations of the rows (1851)-(1854) can depend on the chroma sample type of the chroma samples. For example, for the chroma samples (1803)-(1804) having the respective chroma sample types 0-1, the row (1851) is located between the rows (1811)-(1812). For the chroma samples (1805)-(1806) having the respective the chroma sample types 2-3, the row (1851) is co-located with the current row (1811). For the chroma samples (1807)-(1808) having the respective the chroma sample types 4-5, the row (1851) is co-located with the next row (1812). The above descriptions can be suitably adapted to the rows (1852)-(1854), and the detailed descriptions are omitted for purposes of brevity.

[0183] Any suitable scanning method can be used for displaying, storing, and/or transmitting the luma block and the corresponding chroma block(s) described above in FIG. 18A. In an example, progressive scanning is used.

[0184] An interlaced scan can be used, as shown in FIG. 18B. As described above, the chroma subsampling format is 4:2:0 (e.g., chroma_format_idc is equal to 1). In an example, a variable chroma location type (e.g., ChromaLocType) indicates the current rows (e.g., ChromaLocType is chroma_sample_loc_type_top_field) or the next rows (e.g., ChromaLocType is chroma_sample_loc_type_bottom_field). The current rows (1811), (1813), (1815), and (1817) and the next rows (1812), (1814), (1816), and (1818) can be scanned separately, for example, the current rows (1811), (1813), (1815), and (1817) can be scanned first followed by the next rows (1812), (1814), (1816), and (1818) being scanned. The current rows can include the luma samples (1801) while the next rows can include the luma samples (1802).

[0185] Similarly, the corresponding chroma block can be interlaced scanned. The rows (1851) and (1853) including the chroma samples (1803), (1804), (1805), (1806), (1807), or (1808) with no fill can be referred to as current rows (or current chroma rows), and the rows (1852) and (1854) including the chroma samples (1803), (1804), (1805), (1806), (1807), or (1808) with gray fill can be referred to as next rows (or next chroma rows). In an example,

during the interlaced scan, the rows (1851) and (1853) are scanned first followed by scanning the rows (1852) and (1854).

[0186] In some examples, constrained directional enhancement filtering techniques can be used. The use of an in-loop constrained directional enhancement filter (CDEF) can filter out coding artifacts while retaining the details of the image. In an example (e.g., HEVC), sample adaptive offset (SAO) algorithm can achieves a similar goal by defining signal offsets for different classes of pixels. Unlike SAO, CDEF is a non-linear spatial filter. In some examples, CDEF can be constrained to be easily vectorizable (i.e. implementable with single instruction multiple data (SIMD) operations). It is noted that other non-linear filters, such as a median filter, a bilateral filter cannot be handled in the same manner.

[0187] In some cases, the amount of ringing artifacts in a coded image tends to be roughly proportional to the quantization step size. The amount of detail is a property of the input image, but the smallest detail retained in the quantized image tends to also be proportional to the quantization step size. For a given quantization step size, the amplitude of the ringing is generally less than the amplitude of the details.

[0188] CDEF can be used to identify the direction of each block and then adaptively filter along the identified direction and filter to a lesser degree along directions rotated 45 degrees from the identified direction. In some examples, an encoder can search for the filter strengths and the filter strengths can be signaled explicitly, which allows a high degree of control over the blurring.

[0189] Specifically, in some examples, the direction search is performed on the reconstructed pixels, just after the deblocking filter. Since those pixels are available to the decoder, the directions can be searched by the decoder, and thus the directions require no signaling in an example. In some examples, the direction search can operate on certain block size, such 8×8 blocks, which are small enough to adequately handle non-straight edges, while being large enough to reliably estimate directions when applied to a quantized image. Also, having a constant direction over an 8×8 region makes vectorization of the filter easier. In some examples, each block (e.g., 8×8) can be compared to perfectly directional blocks to determine difference. A perfectly directional block is a block where all of the pixels along a line in one direction have the same value. In an example, a difference measure of the block and each of the perfectly directional blocks, such as sum of squared differences (SSD), root mean square (RMS) error can be calculated. Then, a perfectly directional block with minimum difference (e.g.,

minimum SSD, minimum RMS, and the like) can be determined and the direction of the determined perfectly directional block can be is direction that best matches the pattern in the block.

[0190] FIG. 19 shows an example of direction search according to an embodiment of the disclosure. In an example, a block (1910) is an 8×8 block that is reconstructed, and output from a deblocking filter. In the FIG. 19 example, the direction search can determine a direction from 8 directions shown by (1920) for the block (1910). 8 perfectly directional blocks (1930) are formed respectively corresponding to the 8 directions (1920). A perfectly directional block corresponding to a direction is a block where pixels along a line of the direction have the same value. Further, a difference measure, such as SSD, RMS error and the like, of the block (1910) and each of the perfectly directional blocks (1930) can be calculated. In the FIG. 19 example, the RMS errors are shown by (1940). As shown by (1943), the RMS error of the block (1910) and the perfectly directional block (1933) is the smallest, thus the direction (1923) is the direction that best matches the pattern in the block (1910).

[0191] After the direction of the block is identified, a non-linear low pass directional filter can be determined. For example, the filter taps of the non-linear low pass directional filter can be aligned along the identified direction to reduce ringing while preserving the directional edges or patterns. However, in some examples, directional filtering alone sometimes cannot sufficiently reduce ringing. In an example, extra filter taps are also used on pixels that do not lie along the identified direction. To reduce the risk of blurring, the extra filter taps are treated more conservatively. For this reason, CDEF includes primary filter taps and secondary filter taps. In an example, a complete 2-D CDEF filter can be expressed as Eq.(14):

$$y(i,j) = x(i,j) + \text{round}\Big(\sum_{m,n} w_{d,m,n}^{(p)} f\left(x(m,n) - x(i,j), S^{(p)}, D\right) + \sum_{m,n} w_{d,m,n}^{(s)} f\left(x(m,n) - x(i,j), S^{(s)}, D\right) \Big), \quad \text{Eq. (14)}$$

where D denotes a damping parameter, $S^{(p)}$ denotes the strength of the primary filter taps, $S^{(s)}$ denotes the strength of the secondary filter taps, round(·) denotes an operation that rounds ties away from zero, w denote the filter weights and $f(d, S, D)$ is a constraint function operating on the difference between the filtered pixel and each of the neighboring pixels. In an example, for small differences, the function $f(d, S, D)$is equal to D, that can make the filter to behave like a

linear filter; when the difference is large, the function $f(d, S, D)$ is equal to 0, that can effectively ignores the filter taps.

[0192] In some examples, in-loop restoration schemes are used in video coding post deblocking to generally denoise and enhance the quality of edges, beyond the deblocking operation. In an example, the in-loop restoration schemes are switchable within a frame per suitably sized tile. The in-loop restoration schemes are based on separable symmetric Wiener filters, dual self-guided filters with subspace projection, and domain transform recursive filters. Because content statistics can vary substantially within a frame, in-loop restoration schemes are integrated within a switchable framework where different schemes can be triggered in different regions of the frame.

[0193] According to an aspect of the disclosure, the in-loop restoration (LR) (also referred to as LR filter) can use neighboring pixels during LR filtering, such as a window of pixels around a pixel to be filtered. To be able to apply the LR filter at frame edge, in some examples, before the filtering of LR is applied, some boundary pixel values of pixels at the frame boundary are copied into a boundary buffer. In some examples, two copies of boundary pixels are stored in the boundary buffers. In an example, a first copying step is applied after the deblocking filter and before the CDEF, and the copy of the boundary pixel values of pixels at the frame boundary by the first copying step is denoted as COPY0; the second copying step is applied after the CDEF and before the LR filter, and the copy of the boundary pixel values of pixels at the frame boundary by the second copying step is denoted as COPY1. The boundary buffer is then used for padding of boundary pixels during LR filtering process, for example, when apply the LR filter at the frame edge. The process of copying boundary pixel values into the boundary buffer is denoted as boundary handling.

[0194] Separable symmetric Wiener filter can be one of the in-loop restoration schemes. In some examples, every pixel in a degraded frame can be reconstructed as a non-causal filtered version of the pixels within a $w \times w$ window around it where $w = 2r + 1$ is odd for integer r. If the 2D filter taps are denoted by a $w^2 \times 1$ element vector F in column-vectorized form, a straightforward LMMSE optimization leads to filter parameters being given by $F = H^{-1} M$, where $H = E[XX^T]$ is the autocovariance of x, the column-vectorized version of the w^2 samples in the $w \times w$ window around a pixel, and $M = E[YX^T]$ is the cross correlation of x with the scalar source sample y, to be estimated. In an example, the encoder can estimate H and M from realizations in the deblocked frame and the source and can send the resultant filter F to the decoder. However,

that would not only incur a substantial bit rate cost in transmitting w^2 taps, but also non-separable filtering will make decoding prohibitively complex. In some embodiments, several additional constraints are imposed on the nature of F. For the first constraint, F is constrained to be separable so that the filtering can be implemented as separable horizontal and vertical w-tap convolutions. For the second constraint, each of the horizontal and vertical filters are constrained to be symmetric. For the third constraint, the sum of both the horizontal and vertical filter coefficients is assumed to sum to 1.

[0195] Dual self-guided filtering with subspace projection can be one of the in-loop restoration schemes. Guided filtering is an image filtering technique where a local linear model shown by Eq. (15):

$$y = F x + G \qquad \text{Eq. (15)}$$

is used to compute the filtered output y from an unfiltered sample x, where F and G are determined based on the statistics of the degraded image and a guidance image in the neighborhood of the filtered pixel. If the guide image is the same as the degraded image, the resultant so-called self-guided filtering has the effect of edge preserving smoothing. In an example, a specific form of self-guided filtering can be used. The specific form of self-guided filtering depends on two parameters: a radius r and a noise parameter e, and is enumerated as follows steps:

1. Obtain mean μ and variance σ^2 of pixels in a $(2r + 1) \times (2r + 1)$ window around every pixel. This step can be implemented efficiently with box filtering based on integral imaging.
2. Compute for every pixel: $f = \sigma^2/(\sigma^2 + e)$; $g = (1 - f)\mu$
3. Compute F and G for every pixel as averages of f and g values in a 3×3 window around the pixel for use.

[0196] The specific form of self-guided filter is controlled by r and e, where a higher r implies a higher spatial variance and a higher e implies a higher range variance.

[0197] FIG. 20 shows an example illustrating subspace projection in some examples. As shown in FIG. 20, even though none of the restorations X_1, X_2 are close to the source Y, appropriate multipliers $\{\alpha, \beta\}$ can bring them much closer to the source Y as long as they are moving somewhat in the right direction.

[0198] In some examples, a technique that is referred to as frame super-resolution (FSR) is used in order to improve the perceptual quality of decoded pictures. Generally, FSR process

is applied at low bit-rates, and includes four steps. In a first step, at the encoder side, the source video is downscaled as a non-normative procedure. In a second step, the downscaled video is encoded, followed by filtering processes of a deblocking filter and a CDEF. In a third step, a linear upscaling process is applied as a normative procedure to bring the encoded video back to its original spatial resolution. In a fourth step, the loop restoration filter is applied to resolve part of the high frequency lost. In an example, the last two steps together can be referred to as super-resolving process. Similarly, at the decoder side, processes of decoding, the deblocking filter and CDEF can be applied at lower spatial resolution. Then, the frames go through the super-resolving process. In some examples, in order to reduce overheads in line-buffers with respect to hardware implementation, the upscaling and downscaling process are applied to horizontal dimension only.

[0199] In some examples (e.g., HEVC), a filtering technique that is referred to as sample adaptive offset (SAO) can be used. In some examples, SAO is applied to the reconstruction signal after a deblocking filter. SAO can use the offset values given in the slice header. In some examples, for luma samples, the encoder can decide whether to apply (enable) SAO on a slice. When SAO is enabled, the current picture allows recursive splitting of a coding unit into four sub-regions and each sub-region can select an SAO type from multiple SAO types based on features in the sub-region.

[0200] FIG. 21 shows a table (2100) of a plurality of SAO types according to an embodiment of the disclosure. In the table (2100), SAO types 0-6 are shown. It is noted that SAO type 0 is used to indicate no SAO application. Further, each SAO type of SAO type 1 to SAO type 6 includes multiple categories. SAO can classify reconstructed pixels of a sub-region into categories and reduce the distortion by adding an offset to pixels of each category in the sub-region. In some examples, edge properties can be used for pixel classification in SAO types 1-4, and pixel intensity can be used for pixel classification in SAO types 5-6.

[0201] Specifically, in an embodiment, such as SAO types 5-6, band offset (BO) can be used to classify all pixels of a sub-region into multiple bands. Each band of the multiple bands includes pixels in the same intensity interval. In some examples, the intensity range is equally divided into a plurality of intervals, such as 32 intervals from zero to the maximum intensity value (e.g. 255 for 8-bit pixels), and each interval is associated with an offset. Further, in an example, the 32 bands are divided into two groups, such as a first group and a second group. The first group includes the central 16 bands (e.g., 16 intervals that are in the middle of the

intensity range), while the second group includes of the rest 16 bands (e.g., 8 intervals that are at the low side of the intensity range and 8 intervals that are at the high side of the intensity range). In an example, only offsets of one of the two groups are transmitted. In some embodiments, when the pixel classification operation in BO is used, the five most significant bits of each pixel can be directly used as the band index.

[0202] Further, in an embodiment, such as SAO types 1-4, edge offset (EO) can be used for pixel classification and determination of offsets. For example, pixel classification can be determined based on 1-dimensional 3-pixel patterns with consideration of edge directional information.

[0203] FIG. 22 shows examples of 3-pixel patterns for pixel classification in edge offset in some examples. In the FIG. 22 example, a first pattern (2210) (as shown by 3 grey pixels) is referred to as 0-degree pattern (horizontal direction is associated with the 0-degree pattern), a second pattern (2220) (as shown by 3 grey pixels) is referred to as 90-degree pattern (vertical direction is associated with the 90-degree pattern), a third pattern (2230) (as shown by 3 grey pixels) is referred to as 135-degree pattern (135 degree diagonal direction is associated with the 135-degree pattern) and a fourth pattern (2240) (as shown by 3 grey pixels) is referred to as 45-degree pattern (45 degree diagonal direction is associated with the 45-degree pattern). In an example, one of the four directional patterns shown in FIG. 22 can be selected considering edge directional information for a sub-region. The selection can be sent in the coded video bitstream as side information in an example. Then, pixels in the sub-region can be classified into multiple categories by comparing each pixel with its two neighboring pixels on the direction associated with the directional pattern.

[0204] FIG. 23 shows a table (2300) for pixel classification rule for edge offset in some examples. Specifically, a pixel c (also shown in each pattern of FIG. 22) is compared with two neighboring pixels (also shown by grey color in each pattern of FIG. 22), and the pixel c can be classified into one of category 0-4 based on the comparison according to the pixel classification rule shown in FIG. 23.

[0205] In some embodiments, the SAO on the decoder side can be operated independently of largest coding unit (LCU) (e.g., CTU), so that the line buffers can be saved. In some examples, pixels of the top and bottom rows in each LCU are not SAO processed when the 90-degree, 135-degree, and 45-degree classification patterns are chosen; pixels of the leftmost

and rightmost columns in each LCU are not SAO processed when the 0-degree, 135-degree, and 45-degree patterns are chosen.

[0206] FIG. 24 shows an example (2400) of syntaxes that may need to be signaled for a CTU if the parameters are not merged from neighboring CTU. For example, a syntax element sao_type_idx[cIdx][rx][ry] can be signaled to indicate the SAO type of a sub-region. The SAO type may be BO (band offset) or EO (edge offset). When sao_type_idx[cIdx][rx][ry] has a value of 0, it indicates that SAO is OFF; a value of one to four indicates that one of the four EO categories corresponding to 0°, 90°, 135°, and 45° is used; and a value of five indicates that BO is used. In the FIG. 24 example, each of the BO and EO types has four SAO offset values (sao_offset[cIdx][rx][ry][0] to sao_offset[cIdx][rx][ry][3]) that are signaled.

[0207] Generally, a filtering process can use the reconstructed samples of a first color component as input (e.g., Y or Cb or Cr, or R or G or B) to generate an output, and the output of the filtering process is applied on a second color component that can be the same one as the first color component or that can be another color component that is different from the first color component.

[0208] In a related example of cross-component filtering (CCF), filter coefficients are derived based on some mathematical equations. The derived filter coefficients are signaled from encoder side to the decoder side, and the derived filter coefficients are used to generate offsets using linear combinations. The generated offsets are then added to reconstructed samples as a filtering process. For example, the offsets are generated based on linear combinations of the filtering coefficients with luma samples, and the generated offsets are added to the reconstructed chroma samples. The related example of CCF is based on an assumption of a linear mapping relationship between the reconstructed luma sample values and the delta values between the original and reconstructed chroma samples. However, the mapping between the reconstructed luma sample values and the delta values between the original and reconstructed chroma samples does not necessarily follow a linear mapping process, thus the coding performance of CCF may be limited under the linear mapping relationship assumption.

[0209] In some examples, non linear mapping techniques can be used in cross-component filtering and/or same color component filtering without significant signaling overheads. In an example, the non linear mapping techniques can be used in the cross-component filtering to generate cross-component sample offset. In another example, the non

linear mapping techniques can be used in the same color component filtering to generate local sample offset.

[0210] For convenience, a filtering process that uses the non linear mapping techniques can be referred to as sample offset by non linear mapping (SO-NLM). SO-NLM in the cross-component filtering process can be referred to as a cross-component sample offset (CCSO). SO-NLM in the same color component filtering can be referred to as local sample offset (LSO). Filters that use the non linear mapping techniques can be referred to as non linear mapping based filters. The non linear mapping based filters can include CCSO filters, LSO filters and the like.

[0211] In an example, CCSO and LSO can be used as loop filtering to reduce distortion of reconstructed samples. CCSO and LSO do not rely on the linear mapping assumption used in the related example CCF. For example, CCSO does not rely on the assumption of linear mapping relationship between the luma reconstructed sample values and the delta values between the original chroma samples and chroma reconstructed samples. Similarly, LSO does not rely on the assumption of linear mapping relationship between the reconstructed sample values of a color component and the delta values between the original samples of the color component and reconstructed samples of the color component.

[0212] In the following description, SO-NLM filtering process is described that use the reconstructed samples of a first color component as input (e.g., Y or Cb or Cr, or R or G or B) to generate an output, and the output of the filtering process is applied on a second color component. When the second color component is the same color component as the first color component, the description is applicable for LSO; and when the second color component is different from the first color component, the description is applicable for CCSO.

[0213] In SO-NLM, a non linear mapping is derived at encoder side. A non linear mapping is between reconstructed samples of a first color component in the filter support region and offsets to be added to a second color component in the filter support region. When the second color component is the same as the first color component, the non linear mapping is used in LSO; when the second color component is different from the first color component, the non linear mapping is used in CCSO. The domain of the non linear mapping is determined by different combinations of processed input reconstructed samples (also referred to as combinations of possible reconstructed sample values).

[0214] Techniques of SO-NLM can be illustrated using a specific example. In the specific example, reconstructed samples from a first color component located in a filter support

area (also referred to as "filter support region") are determined. The filter support area is an area within which the filter can be applied, and the filter support area can have any suitable shape.

[0215] FIG. 25 shows an example of a filter support area (2500) according to some embodiments of the disclosure. The filter support area (2500) includes four reconstructed samples: P0, P1, P2 and P3 of a first color component. In the FIG. 25 example, the four reconstructed samples can form a cross-shape in the vertical direction and the horizontal direction, and the center location of the cross-shape is the location for the sample to be filtered. A sample at the center location and of the same color component as P0-P3 is denoted by C. A sample at the center location and of a second color component is denoted by F. The second color component can be the same as the first color component of P0-P3 or can be different from the first color component of P0-P3.

[0216] FIG. 26 shows an example of another filter support area (2600) according to some embodiments of the disclosure. The filter support area (2600) includes four reconstructed samples P0, P1, P2 and P3 of a first color component that form a square shape. In the FIG. 26 example, the center location of the square shape is the location of the sample to be filtered. A sample at the center location and of the same color component as P0-P3 is denoted by C. A sample at the center location and of a second color component is denoted by F. The second color component can be the same as the first color component of P0-P3 or can be different from the first color component of P0-P3.

[0217] The reconstructed samples are input to the SO-NLM filter, and are suitably processed to form filter taps. In an example, the location of a reconstructed sample that is an input to the SO-NLM filter is referred to as filter tap location. In a specific example, the reconstructed samples are processed in following two steps.

[0218] In a first step, the delta values respectively between P0-P3 and C are computed. For example, m0 denotes the delta value between P0 to C; m1 denotes the delta value between P1 to C; m2 denotes the delta value between P2 to C; m3 denotes the delta value between P3 to C.

[0219] In a second step, the delta values m0-m3 are further quantized, the quantized values are denoted as d0, d1, d2, d3. In an example, the quantized value can be one of -1, 0, 1 based on a quantization process. For example, a value m can be quantized to -1 when m is smaller than –N (N is a positive value and is referred to as quantization step size); the value m

can be quantized to 0 when m is in a range of [-N, N]; and the value m can be quantized to 1 when m is greater than N. In some examples, the quantization step size N can be one of 4, 8, 12, 16 and the like.

[0220] In some embodiments, the quantized values d0-d3 are filter taps and can be used to identify one combination in the filter domain. For example, the filter taps d0-d3 can form a combination in the filter domain. Each filter tap can have three quantized values, thus when four filter taps are used, the filter domain includes 81 (3×3×3×3) combinations.

[0221] FIGs. 27A-27C show a table (2700) having 81 combinations according to an embodiment of the disclosure. The table (2700) includes 81 rows corresponding to 81 combinations. In each row corresponding to a combination, the first column includes an index of the combinations; the second column includes the value of filter tap d0 for the combination; the third column includes the value of filter tap d1 for the combination; the fourth column includes the value of filter tap d2 for the combination; the fifth column includes the value of filter tap d3 for the combination; the six column includes the offset value associated with the combination for the non linear mapping. In an example, when the filter taps d0-d3 are determined, the offset value (denoted by s) associated with the combination of d0-d3 can be determined according to the table (2700). In an example, offset values s0-s80 are integers, such as 0, 1, -1, 3, -3, 5, -5, -7, and the like.

[0222] In some embodiments, the final filtering process of SO-NLM can be applied as shown in Eq. (16):

$$f' = clip(f + s) \quad \text{Eq. (16)}$$

where f is the reconstructed sample of the second color component to be filtered, and s is the offset value determined according to filter taps that are processing results of reconstructed samples of first color component, such as using table (2700). The sum of the reconstructed sample F and the offset value s is further clipped into the range associated with bit-depth to determine the final filtered sample f' of the second color component.

[0223] It is noted that, in the case of LSO, the second color component in the above description is the same as the first color component; and, in the case of CCSO, the second color component in the above description can be different from the first color component.

[0224] It is noted that, the above description can be adjusted for other embodiments of the present disclosure.

[0225] In some examples, at the encoder side, the encoding device can derive a mapping between reconstructed samples of a first color component in a filter support region and the offsets to be added to reconstructed samples of a second color component. The mapping can be any suitable linear or non-linear mapping. Then, the filtering process can be applied at the encoder side and/or the decoder side based on the mapping. For example, the mapping is suitably informed to the decoder (e.g., the mapping is included in a coded video bitstream that is transmitted from the encoder side to the decoder side), and then the decoder can perform the filtering process based on the mapping.

[0226] Performance of the non linear mapping based filters, such as the CCSO filters, LSO filters and the like, depends on filter shape configuration. A filter shape configuration (also referred to as filter shape) of a filter can refer to properties of a pattern formed by filter tap positions. The pattern can be defined by various parameters, such as a number of filter taps, a geometry shape of the filter tap positions, a distance of filter tap positions to a center of the pattern, and the like. Using a fixed filter shape configuration may limit the performance of the non linear mapping based filters.

[0227] As shown by FIG. 24 and FIG. 25 and FIG. 27A-27C, some examples use 5-tap filter design for the filter shape configurations of the non linear mapping based filters. The 5-tap filter design can use tap locations at P0, P1, P2, P3 and C. The 5-tap filter design for the filter shape configuration can result in a look up table (LUT) with 81 entries as shown in FIG. 27A-27C. The LUT of sample offsets need to be signaled from the encoder side to the decoder side, and the signaling of the LUT can contribute to a majority of the signaling overhead and affect the coding efficiency of using the non linear mapping based filters. According to some aspects the disclosure, the number of filter taps can be different from 5. In some examples, the number of filter taps can be reduced, the information in the filter support area can still be captured, and the coding efficiency can be improved.

[0228] In some examples, the filter shape configurations in the group for the non linear mapping based filter respectively have 3 filter taps.

[0229] FIG. 28 shows 7 filter shape configurations of 3 filter taps in an example. Specifically, a first filter shape configuration includes 3 filter taps at positions that are labeled as "1" and "c", the position "c" is the center position of the positions "1"; a second filter shape configuration includes 3 filter taps at positions that are labeled as "2" and the position "c", the position "c" is the center position of the positions "2"; a third filter shape configuration includes

3 filter taps at positions that are labeled as "3" and the position "c", the position "c" is the center position of the positions "3"; a fourth filter shape configuration includes 3 filter taps at positions that are labeled as "4" and the position "c", the position "c" is the center position of the positions "4"; a fifth filter shape configuration includes 3 filter taps at positions that are labeled as "5" and "c", the position "c" is the center position of the positions "5"; a sixth filter shape configuration includes 3 filter taps at positions that are labeled as "6" and the position "c", the position "c" is the center position of the positions "6"; a seventh filter shape configuration includes 3 filter taps at positions that are labeled as "7" and the position "c", the position "c" is the center position of the positions "7".

[0230] Aspects of the present disclosure provide techniques to integrate video processing tools, such as filtering, boundary handling, and clipping tools. In some examples, a non linear mapping based filter (e.g., CCSO, LSO) or other tools (e.g., clipping module) is after the deblocking filter and before the LR filter, boundary handling processes that store two copies of boundary pixels are also applied after the deblocking filter and before the LR filter. The present disclosure provide various configurations for including the non linear mapping based filter(s), boundary handling module(s), and/or clipping module(s) after the deblocking filter and before the LR filter.

[0231] FIG. 29 shows a block diagram of a loop filter chain (2900) in some examples. The loop filter chain (2900) includes a plurality of filters connected in series in a filter chain. The loop filter chain (2900) can be used as in loop filtering unit, such as the loop filter unit (556) in an example. The loop filter chain (2900) can be used in the encoding or decoding loops before storing the reconstructed picture in a decoded picture buffer, such as the reference picture memory (557). The loop filter chain (2900) receives input reconstructed samples from a prior processing module, applies filters on the reconstructed samples to generate output reconstructed samples.

[0232] The loop filter chain (2900) can include any suitable filters. In the FIG. 29 example, the loop filter chain (2900) includes a deblocking filter (labeled as deblocking), a constrained directional enhancement filter (labeled as CDEF) and an in-loop restoration filter (labeled as LR) connected in a chain. The loop filter chain (2900) has an input node (2901), an output node (2909) and a plurality of intermediate nodes (2902)-(2903). The input node (2901) of the loop filter chain (2900) receives the input reconstructed samples from the prior processing module, and the input reconstructed samples are provided to the deblocking filter. The

intermediate node (2902) receives reconstructed samples (after being processed by the deblocking filter) from the deblocking filter, and provides the reconstructed samples to the CDEF for further filter processing. The intermediate node (2903) receives reconstructed samples (after being processed by the CDEF) from the CDEF , and provides the reconstructed samples to the LR filter for further filter processing. The output node (2909) receives output reconstructed samples from the LR filter (after being processed by the LR filter). The output reconstructed samples can be provided to other processing modules, such as a post processing module for further processing.

[0233] It is noted that the following description illustrates techniques of using a non linear mapping based filter in the loop filter chain (2900). The techniques of using the non linear mapping based filter can be used in other suitable loop filter chain.

[0234] According to some aspects of the disclosure, when a non linear mapping based filter is used in a loop filter chain, at least one of the copies (COPY0 and COPY1) of boundary pixel values that are used by boundary handling in the LR filter is associated with the non linear mapping based filter. In some examples, reconstructed samples from where the boundary pixel values are obtained and buffered can be the input of the non linear mapping based filter. In some examples, reconstructed samples from where the boundary pixel values are obtained and buffered can be a result of applying the non linear mapping based filter. In some examples, reconstructed samples from where the boundary pixel values are obtained and buffered can be combined with sample offsets generated by the non linear mapping based filter.

[0235] In some examples, pixels after the deblocking filter and before the CDEF or the non linear mapping based filter are used for COPY0, pixels after applying CDEF or the non linear mapping based filter, and before LR filter are used for COPY1.

[0236] FIG. 30 shows an example of a loop filter chain (3000) that includes a non linear mapping based filter and a CDEF between the input and the output of the non linear mapping based filter. The loop filter chain (3000) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3000), the deblocking filter generates first intermediate reconstructed samples at a first intermediate node (3011), and the first intermediate reconstructed samples are input to the CDEF and a non linear mapping based filter (labeled as SO-NLM). The CDEF is applied on the first intermediate reconstructed samples and generates second intermediate reconstructed samples at a second intermediate node (3012). The non linear mapping based filter generates sample offsets SO based on the first

intermediate reconstructed samples. The sample offsets SO are combined with the second intermediate reconstructed samples to generate third intermediate reconstructed samples at a third intermediate node (3013). Then, the LR filter is applied on the third intermediate reconstructed samples to generate an output of the loop filter chain (3000). Further, the first intermediate reconstructed samples at the first intermediate node (3011) are used to obtain the first copy COPY0 of the boundary pixels for boundary handling in the LR filter, and third intermediate reconstructed samples at the third intermediate node (3013) are used to obtain the second copy COPY1 of the boundary pixels for boundary handling in the LR filter.

[0237] In some examples, pixels after the deblocking filter and before applying of the CDEF or the non linear mapping based filter are used for COPY0, pixels after applying CDEF and before applying of the non linear mapping based filter are used for COPY1.

[0238] FIG. 31 shows an example of a loop filter chain (3100) that includes a non linear mapping based filter and a CDEF between the input and the output of the non linear mapping based filter. The loop filter chain (3100) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3100), the deblocking filter generates first intermediate reconstructed samples at a first intermediate node (3111), and the first intermediate reconstructed samples are input to the CDEF and a non linear mapping based filter (labeled as SO-NLM). The CDEF is applied on the first intermediate reconstructed samples and generates second intermediate reconstructed samples at a second intermediate node (3112). The non linear mapping based filter generates sample offsets SO based on the first intermediate reconstructed samples. The sample offsets SO are combined with the second intermediate reconstructed samples to generate third intermediate reconstructed samples at a intermediate third node (3113). Then, the LR filter is applied on the third intermediate reconstructed samples to generate an output of the loop filter chain (3100). Further, the first intermediate reconstructed samples at the first intermediate node (3111) are used to obtain the first copy COPY0 of the boundary pixels for boundary handling in the LR filter, and second intermediate reconstructed samples at the second intermediate node (3012) are used to obtain the second copy COPY1 of the boundary pixels for boundary handling in the LR filter.

[0239] In some examples, the non linear mapping based filter is connected in series with other filters in the loop filter chain, and no other filter is between the input and output of the non linear mapping based filter. In an example, the non linear mapping based filter is applied after

the CDEF. The pixels after the deblocking filter and before the CDEF are used for COPY0, pixels after non linear mapping based filter and before the LR filter are used for COPY1.

[0240] FIG. 32 shows an example of a loop filter chain (3200) in an example. The loop filter chain (3200) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3200), the CDEF is between a first intermediate node (3211) and a second intermediate node (3212), and a non linear mapping based filter (labeled as SO-NLM) is applied at the second intermediate node (3212) of the loop filter chain (3200). Specifically, the deblocking filter generates first intermediate reconstructed samples at the first intermediate node (3211), the CDEF is applied on the first intermediate reconstructed samples and generates second intermediates reconstructed samples at the second intermediate node (3212). The second intermediates reconstructed samples are the input of the non linear mapping based filter. Based on the input, the non linear mapping based filter generates sample offsets (SO). The sample offsets are combined with the second intermediate reconstructed samples at the second intermediate node (3212) to generate third intermediate reconstructed samples at a third intermediate node (3213). The LR filter is applied to the third intermediate reconstructed samples to generate the output of the loop filter chain (3200). Further, the first intermediate reconstructed samples at the first intermediate node (3211) are used to obtain the first copy COPY0 of the boundary pixels for boundary handling in the LR filter, and third intermediate reconstructed samples at the third node (3213) are used to obtain the second copy COPY1 of the boundary pixels for boundary handling in the LR filter.

[0241] In an example, the non linear mapping based filter is applied before the CDEF. The pixels after the deblocking filter and before the non linear mapping based filter are used for COPY0, pixels after the CDEF and before the LR filter are used for COPY1.

[0242] FIG. 33 shows an example of a loop filter chain (3300) in an example. The loop filter chain (3300) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3300), a non linear mapping based filter (labeled as SO-NLM) is applied at a first intermediate node (3311) of the loop filter chain (3300), the CDEF is between a second intermediate node (3312) and a third intermediate node (3313). Specifically, the deblocking filter generates first intermediate reconstructed samples at the first intermediate node (3311). The first intermediates reconstructed samples are the input of the non linear mapping based filter. Based on the input, the non linear mapping based filter generates sample offsets (SO). The sample offsets are combined with the first intermediate reconstructed

samples at the first intermediate node (3311) to generate second intermediate reconstructed samples at the second intermediate node (3312). The CDEF is applied on the second intermediate reconstructed samples and generates third intermediates reconstructed samples at the third intermediate node (3313). The LR filter is applied to the third intermediate reconstructed samples to generate the output of the loop filter chain (3300). Further, the first intermediate reconstructed samples at the first intermediate node (3311) are used to obtain the first copy COPY0 of the boundary pixels for boundary handling in the LR filter, and third intermediate reconstructed samples at the third intermediate node (3313) are used to obtain the second copy COPY1 of the boundary pixels for boundary handling in the LR filter.

[0243] According to an aspect of the disclosure, the enabling/disabling of boundary handling (e.g., obtain COPY0 and COPY1 and use COPY0 and COPY1 in the LR filter) of LR filter depends on the enabling/disabling of other filtering tools, such as non linear mapping based filter, the CDEF and/or FSR in the loop filtering chain.

[0244] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of non linear mapping based filter, and/or CDEF, and/or FSR. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-SO-NLM is used to indicate the enabling (e.g., the flag En-SO-NLM has binary 1) or disabling (e.g., the flag En-SO-NLM has binary 0) of applying a non linear mapping based filter; a flag En-CDEF is used to indicate the enabling (e.g., the flag En-CDEF has binary 1) or disabling (e.g., the flag En-CDEF has binary 0) of applying the CDEF; a flag En-FSR is used to indicate the enabling (e.g., the flag En-FSR has binary 1) or disabling (e.g., the flag En-FSR has binary 0) of applying the FSR. Then, in an exmaple, the flag En-Boundary is a logic combination of the flag En-SO-NLM, the flag En-CDEF and the flag En-FSR. It is noted that any suitable logic operators, such as AND, OR, NOT, and the like can be used.

[0245] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of non linear mapping based fitler, and/or CDEF. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-SO-NLM is used to indicate the enabling (e.g., the flag En-SO-NLM has binary 1) or disabling (e.g., the flag En-SO-NLM has binary 0) of applying a non linear mapping based filter; a flag En-CDEF is used to indicate the enabling (e.g., the flag En-CDEF has binary 1) or disabling (e.g., the flag En-CDEF

has binary 0) of applying the CDEF. Then, in an exmaple, the flag En-Boundary is a logic combination of the flag En-SO-NLM, and the flag En-CDEF. It is noted that any suitable logic operators, such as AND, OR, NOT, and the like can be used.

[0246] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of non linear mapping based filter, and/or FSR. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-SO-NLM is used to indicate the enabling (e.g., the flag En-SO-NLM has binary 1) or disabling (e.g., the flag En-SO-NLM has binary 0) of applying a non linear mapping based filter; a flag En-FSR is used to indicate the enabling (e.g., the flag En-FSR has binary 1) or disabling (e.g., the flag En-FSR has binary 0) of applying the FSR. Then, in an exmaple, the flag En-Boundary is a logic combination of the flag En-SO-NLM, and the flag En-FSR. It is noted that any suitable logic operators, such as AND, OR, NOT, and the like can be used.

[0247] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of non linear mapping based filter. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-SO-NLM is used to indicate the enabling (e.g., the flag En-SO-NLM has binary 1) or disabling (e.g., the flag En-SO-NLM has binary 0) of applying a non linear mapping based filter. Then, in an example, the flag En-Boundary can be the flag En-SO-NLM or can be logic NOT of the flag En-SO-NLM.

[0248] In one embodiment, the enabling/disabling of boundary handling depends on enabling/disabling of CDEF, and/or FSR.

[0249] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of the CDEF, and/or FSR. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-CDEF is used to indicate the enabling (e.g., the flag En-CDEF has binary 1) or disabling (e.g., the flag En-CDEF has binary 0) of applying the CDEF; a flag En-FSR is used to indicate the enabling (e.g., the flag En-FSR has binary 1) or disabling (e.g., the flag En-FSR has binary 0) of applying the FSR. Then, in an example, the flag En-Boundary is a logic combination of the flag En-CDEF and the flag En-FSR. It is noted that any suitable logic operators, such as AND, OR, NOT, and the like can be used.

[0250] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of the CDEF. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En-CDEF is used to indicate the enabling (e.g., the flag En- CDEF has binary 1) or disabling (e.g., the flag En- CDEF has binary 0) of applying a non linear mapping based filter. Then, in an example, the flag En-Boundary can be the flag En-CDEF or can be logic NOT of the flag En- CDEF.

[0251] In one embodiment, the enabling/disabling of boundary handling depends on enabling/disabling of FSR.

[0252] In some examples, the enabling/disabling of boundary handling depends on enabling/disabling of FSR. For example, a flag En-Boundary is used to indicate the enabling (e.g., the flag En-Boundary has binary 1) or disabling (e.g., the flag En-Boundary has binary 0) of the boundary handling; a flag En- FSR is used to indicate the enabling (e.g., the flag En- FSR has binary 1) or disabling (e.g., the flag En- FSR has binary 0) of applying a non linear mapping based filter. Then, in an example, the flag En-Boundary can be the flag En- FSR or can be logic NOT of the flag En- FSR.

[0253] According to some aspects of the disclosure, pixels values are clipped before the LR filter. In some examples, the pixels after applying the CDEF are clipped first, denote as CLIP0. Then, the pixels after applying the non linear mapping based filter are clipped, denoted as CLIP1. It is noted that, in some examples, the pixel values are clipped into a suitable range that is meaningful for pixel values. In an example, when pixel values are represented by 8 bits, the pixel values can be clipped to a range of [0,255].

[0254] It is noted that in the present description, the non linear mapping based filter is used as an example to illustrate the techniques to buffer boundary pixel values at different nodes along a loop filter chain and/or to clip pixel vales at different nodes along a loop filter chain. The techniques can be used when other coding tool, such as a cross component sample adaptive offset (CCSAO) tool, and the like, is applied after the deblocking filter and before the LR filter.

[0255] FIG. 34 shows an example of a loop filter chain (3400) that includes a non linear mapping based filter with a CDEF between the input and the output of the non linear mapping based filter. The loop filter chain (3400) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3400), a non linear mapping based filter (labeled as SO-NLM) is applied between two intermediate nodes. The deblocking

filter generates first intermediate reconstructed samples at a first intermediate node (3411). The first intermediate reconstructed samples are input of both the CDEF and the non linear mapping based filter. Based on the first intermediate reconstructed samples, the CDEF is applied and generate second intermediate reconstructed samples at a second intermediate node (3412). Also based on the first intermediate reconstructed samples, the non linear mapping based filter generates sample offsets (SO). The second intermediate reconstructed samples are clipped to generate CLIP0. The CLIP0 is combined with the sample offsets to generate third intermediate reconstructed samples at a third intermediate node (3413), the third intermediate reconstructed samples are clipped to generate CLIP1. The CLIP1 is provided to the LR filter for further filtering.

[0256] FIG. 35 shows an example of a loop filter chain (3500) that includes a non linear mapping based filter with a CDEF between the input and the output of the non linear mapping based filter. The loop filter chain (3500) can be used in the place of the loop filter chain (2900) in an encoding device or a decoding device. In the loop filter chain (3500), a non linear mapping based filter (labeled as SO-NLM) is applied between two intermediate nodes. The deblocking filter generates first intermediate reconstructed samples at a first intermediate node (3511). The first intermediate reconstructed samples are input of both the CDEF and the non linear mapping based filter. Based on the first intermediate reconstructed samples, the CDEF is applied and generate second intermediate reconstructed samples at a second intermediate node (3512). Also based on the first intermediate reconstructed samples, the non linear mapping based filter generates sample offsets (SO). The second intermediate reconstructed samples are combined with the sample offsets to generate third intermediate reconstructed samples at a third intermediate node (3513), the third intermediate reconstructed samples are clipped to generate CLIP0. The CLIP0 is provided to the LR filter for further filtering.

[0257] FIG. 36 shows a flow chart outlining a process (3600) according to an embodiment of the disclosure. The process (3600) can be used for video filtering. When the term block is used, block may be interpreted as a prediction block, a coding unit, a luma block, a chroma block, or the like. In various embodiments, the process (3600) are executed by processing circuitry, such as the processing circuitry in the terminal devices (310), (320), (330) and (340), the processing circuitry that performs functions of the video encoder (403), the processing circuitry that performs functions of the video decoder (410), the processing circuitry that performs functions of the video decoder (510), the processing circuitry that performs

functions of the video encoder (603), and the like. In some embodiments, the process (3600) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (3600). The process starts at (S3601) and proceeds to (S3610).

[0258] At (S3610), first boundary pixel values of first reconstructed samples at a first node along a loop filter chain are buffered. The first node is associated with a non linear mapping based filter that is applied in the loop filter chain before a loop restoration filter.

[0259] In some examples, the non linear mapping based filter is a cross-component sample offset (CCSO) filter. In some examples, the non linear mapping based filter is a local sample offset (LSO) filter.

[0260] At (S3620), the loop restoration filter is applied on to-be filtered reconstructed samples based on the buffered first boundary pixel values.

[0261] In some examples, the first reconstructed samples at the first node are input of the non linear mapping based filter.

[0262] In the FIG. 30, FIG. 31, and FIG. 33 examples, the first node can be the first intermediate node (3011)/(3111)/(3311) in the respective description of FIG. 30, FIG. 31 and FIG. 33, the first reconstructed samples can be the first intermediate reconstructed samples in the respective description of FIG. 30, FIG. 31 and FIG. 33.

[0263] Then, in the examples of FIG. 30 and FIG. 33, second boundary pixel values of second reconstructed samples at a second node along the loop filter chain can be buffered, the second reconstructed samples at the second node are generated after an applying of sample offsets generated by the non linear mapping based filter. The loop restoration filter is applied on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

[0264] In the example of FIG. 30, the sample offsets generated by the non linear mapping based filter are combined with output (e.g., the second intermediate reconstructed samples in the description of FIG. 30) of a constrained directional enhanced filter to generate the second reconstructed samples (e.g., the third intermediate reconstructed samples in the description of FIG. 30).

[0265] In the example of FIG. 33, the sample offsets generated by the non linear mapping based filter are combined with the first reconstructed samples (e.g., the first intermediate reconstructed samples in the description of FIG. 33) to generate intermediate reconstructed

samples (e.g., the second intermediate reconstructed samples in the description of FIG. 33), and then a constrained directional enhanced filter is applied to the intermediate reconstructed samples to generate the second reconstructed samples (e.g., the third intermediate reconstructed samples in the description of FIG. 33).

[0266] In the example of FIG. 31, the second boundary pixel values of second reconstructed samples at a second node (e.g., the second intermediate reconstructed samples at the second intermediate node (3112) in the description of FIG. 33) along the loop filter are buffered. The second reconstructed samples at the second node are combined with sample offsets generated by the non linear mapping based filter to generate the to-be filtered reconstructed samples. The loop restoration filter is applied on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

[0267] In the FIG. 32 example, the first reconstructed samples can be the third intermediate reconstructed samples at the third intermediate node (3213) in the description of FIG. 32, which is the result of applying the non linear mapping based filter. Then, second boundary pixel values of second reconstructed samples (e.g., the first intermediate reconstructed samples in the description of FIG. 32) generated by the deblocking filter are buffered. The constrained directional enhanced filter is applied on the second reconstructed samples to generate intermediate reconstructed samples (e.g., second intermediate reconstructed samples in the description of FIG. 32). The intermediate reconstructed samples are combined with sample offsets generated by the non linear mapping based filter to generate the first reconstructed samples (e.g., the third intermediate reconstructed samples in the description of FIG. 32). Then, the loop restoration filter can be applied based on the buffered first boundary pixel values and the buffered second boundary pixel values.

[0268] In some examples, the to-be filtered reconstructed samples are clipped into a suitable range, such as [0,255] in the case of 8 bits, before the applying of the loop restoration filter, such as the examples in FIG. 34 and FIG. 35. In some examples (e.g., in FIG. 34), intermediate reconstructed samples (e.g., second intermediate reconstructed samples in the description of FIG. 34) are clipped into a suitable range, such as [0,255] in the case of 8 bits, before a combination with sample offsets generated by the non linear mapping based filter.

[0269] The process (3600) proceeds to (S3699), and terminates.

[0270] It is noted that, in some examples, the non linear mapping based filter is a cross-component sample offset (CCSO) filter, and in some other examples, the non linear mapping based filter is a local sample offset (LSO) filter.

[0271] The process (3600) can be suitably adapted. Step(s) in the process (3600) can be modified and/or omitted. Additional step(s) can be added. Any suitable order of implementation can be used.

[0272] Embodiments in the disclosure may be used separately or combined in any order. Further, each of the methods (or embodiments), an encoder, and a decoder may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium.

[0273] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media. For example, FIG. 37 shows a computer system (3700) suitable for implementing certain embodiments of the disclosed subject matter.

[0274] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by one or more computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0275] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[0276] The components shown in FIG. 37 for computer system (3700) are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system (3700).

[0277] Computer system (3700) may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio

input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human interface devices can also be used to capture certain media not necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[0278] Input human interface devices may include one or more of (only one of each depicted): keyboard (3701), mouse (3702), trackpad (3703), touch screen (3710), data-glove (not shown), joystick (3705), microphone (3706), scanner (3707), camera (3708).

[0279] Computer system (3700) may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen (3710), data-glove (not shown), or joystick (3705), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (3709), headphones (not depicted)), visual output devices (such as screens (3710) to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[0280] Computer system (3700) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (3720) with CD/DVD or the like media (3721), thumb-drive (3722), removable hard drive or solid state drive (3723), legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[0281] Those skilled in the art should also understand that term "computer readable media" as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[0282] Computer system (3700) can also include an interface (3754) to one or more communication networks (3755). Networks can for example be wireless, wireline, optical. Networks can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks include local area networks such as Ethernet, wireless

LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and terrestrial broadcast TV, vehicular and industrial to include CANBus, and so forth. Certain networks commonly require external network interface adapters that attached to certain general purpose data ports or peripheral buses (3749) (such as, for example USB ports of the computer system (3700)); others are commonly integrated into the core of the computer system (3700) by attachment to a system bus as described below (for example Ethernet interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks, computer system (3700) can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[0283] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (3740) of the computer system (3700).

[0284] The core (3740) can include one or more Central Processing Units (CPU) (3741), Graphics Processing Units (GPU) (3742), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (3743), hardware accelerators for certain tasks (3744), graphics adapter (3750), and so forth. These devices, along with Read-only memory (ROM) (3745), Random-access memory (3746), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (3747), may be connected through a system bus (3748). In some computer systems, the system bus (3748) can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus (3748), or through a peripheral bus (3749). In an example, a display (3710) can be connected to the graphics adapter (3750). Architectures for a peripheral bus include PCI, USB, and the like.

[0285] CPUs (3741), GPUs (3742), FPGAs (3743), and accelerators (3744) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (3745) or RAM (3746). Transitional data can be also be stored in RAM (3746), whereas permanent data can be stored for example, in the internal mass storage (3747). Fast storage and retrieve to any of the memory devices can be enabled through

the use of cache memory, that can be closely associated with one or more CPU (3741), GPU (3742), mass storage (3747), ROM (3745), RAM (3746), and the like.

[0286] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[0287] As an example and not by way of limitation, the computer system having architecture (3700), and specifically the core (3740) can provide functionality as a result of processor(s) (including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core (3740) that are of non-transitory nature, such as core-internal mass storage (3747) or ROM (3745). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (3740). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (3740) and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM (3746) and modifying such data structures according to the processes defined by the software. In addition or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator (3744)), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

Appendix A: Acronyms

JEM: joint exploration model

VVC: versatile video coding

BMS: benchmark set

MV: Motion Vector

HEVC: High Efficiency Video Coding

MPM: most probable mode

WAIP: Wide-Angle Intra Prediction

SEI: Supplementary Enhancement Information

VUI: Video Usability Information

GOPs: Groups of Pictures

TUs: Transform Units,

PUs: Prediction Units

CTUs: Coding Tree Units

CTBs: Coding Tree Blocks

PBs: Prediction Blocks

HRD: Hypothetical Reference Decoder

SDR: standard dynamic range

SNR: Signal Noise Ratio

CPUs: Central Processing Units

GPUs: Graphics Processing Units

CRT: Cathode Ray Tube

LCD: Liquid-Crystal Display

OLED: Organic Light-Emitting Diode

CD: Compact Disc

DVD: Digital Video Disc

ROM: Read-Only Memory

RAM: Random Access Memory

ASIC: Application-Specific Integrated Circuit

PLD: Programmable Logic Device

LAN: Local Area Network

GSM: Global System for Mobile communications

LTE: Long-Term Evolution

CANBus: Controller Area Network Bus

USB: Universal Serial Bus

PCI: Peripheral Component Interconnect

FPGA: Field Programmable Gate Areas

SSD: solid-state drive

IC: Integrated Circuit

CU: Coding Unit

PDPC: Position Dependent Prediction Combination

ISP: Intra Sub-Partitions

SPS: Sequence Parameter Setting

HDR: high dynamic range

SDR: standard dynamic range

JVET: Joint Video Exploration Team

MPM: most probable mode

WAIP: Wide-Angle Intra Prediction

CU: Coding Unit

PU: Prediction Unit

TU: Transform Unit

CTU: Coding Tree Unit

PDPC: Position Dependent Prediction Combination

ISP: Intra Sub-Partitions

SPS: Sequence Parameter Setting

PPS: Picture Parameter Set

APS: Adaptation Parameter Set

VPS: Video Parameter Set

DPS: Decoding Parameter Set

ALF: Adaptive Loop Filter

SAO: Sample Adaptive Offset

CC-ALF: Cross-Component Adaptive Loop Filter

CDEF: Constrained Directional Enhancement Filter

CCSO: Cross-Component Sample Offset

CCSAO: Cross-Component Sample Adaptive Offset

LSO: Local Sample Offset

LR: Loop Restoration Filter

FSR: Frame Super-Resolution

AV1: AOMedia Video 1

AV2: AOMedia Video 2

[0288] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

WHAT IS CLAIMED IS:

1. A method for filtering in video coding, comprising:

buffering, by a processor, first boundary pixel values of first reconstructed samples at a first node along a loop filter chain, the first node being associated with a non linear mapping based filter that is applied in the loop filter chain before a loop restoration filter, the first boundary pixel values being values of pixels at a frame boundary; and

applying, by the processor, the loop restoration filter on to-be filtered reconstructed samples based on the buffered first boundary pixel values.

2. The method of claim 1, wherein the non linear mapping based filter is at least one of a cross-component sample offset (CCSO) filter and a local sample offset (LSO) filter.

3. The method of claim 1, wherein the first reconstructed samples at the first node are input of the non linear mapping based filter.

4. The method of claim 3, further comprising:

buffering, by the processor, second boundary pixel values of second reconstructed samples at a second node along the loop filter chain, the second reconstructed samples at the second node being generated after an applying of sample offsets generated by the non linear mapping based filter, the second boundary pixel values being values of the pixels at the frame boundary; and

applying, by the processor, the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

5. The method of claim 4, further comprising:

combining, by the processor, the sample offsets generated by the non linear mapping based filter with output of a constrained directional enhanced filter to generate the second reconstructed samples.

6. The method of claim 4, further comprising:

combining, by the processor, the sample offsets generated by the non linear mapping based filter with the first reconstructed samples to generate intermediate reconstructed samples; and

applying, by the processor, a constrained directional enhanced filter to the intermediate reconstructed samples to generate the second reconstructed samples.

7. The method of claim 3, further comprising:

buffering, by the processor, second boundary pixel values of second reconstructed samples at a second node along the loop filter, the second boundary pixel values being values of the pixels at the frame boundary;

combining, by the processor, the second reconstructed samples at the second node with sample offsets generated by the non linear mapping based filter to generate the to-be filtered reconstructed samples; and

applying, by the processor, the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

8. The method of claim 1, further comprising:

buffering second boundary pixel values of second reconstructed samples generated by a deblocking filter, the second boundary pixel values being values of the pixels at the frame boundary;

applying a constrained directional enhanced filter on the second reconstructed samples to generate intermediate reconstructed samples; and

combining the intermediate reconstructed samples with sample offsets generated by the non linear mapping based filter to generate the first reconstructed samples.

9. The method of claim 1, further comprising:

clipping the to-be filtered reconstructed samples before the applying of the loop restoration filter.

10. The method of claim 9, further comprising:

clipping intermediate reconstructed samples before a combination with sample offsets generated by the non linear mapping based filter.

11. An apparatus for video coding, comprising processing circuitry configured to:

buffer first boundary pixel values of first reconstructed samples at a first node along a loop filter chain, the first node being associated with a non linear mapping based filter that is applied in the loop filter chain before a loop restoration filter, the first boundary pixel values being values of pixels at a frame boundary; and

apply the loop restoration filter on to-be filtered reconstructed samples based on the buffered first boundary pixel values.

12. The apparatus of claim 11, wherein the non linear mapping based filter is at least one of a cross-component sample offset (CCSO) filter and a local sample offset (LSO) filter.

13. The apparatus of claim 11, wherein the first reconstructed samples at the first node are input of the non linear mapping based filter.

14. The apparatus of claim 13, wherein the processing circuitry is configured to:

buffer second boundary pixel values of second reconstructed samples at a second node along the loop filter chain, the second reconstructed samples at the second node being generated after an applying of sample offsets generated by the non linear mapping based filter, the second boundary pixel values being values of the pixels at the frame boundary; and

apply the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

15. The apparatus of claim 14, wherein the processing circuitry is configured to:

combine the sample offsets generated by the non linear mapping based filter with output of a constrained directional enhanced filter to generate the second reconstructed samples.

16. The apparatus of claim 14, wherein the processing circuitry is configured to:

combine the sample offsets generated by the non linear mapping based filter with the first reconstructed samples to generate intermediate reconstructed samples; and

apply a constrained directional enhanced filter to the intermediate reconstructed samples to generate the second reconstructed samples.

17. The apparatus of claim 13, wherein the processing circuitry is configured to:

buffer second boundary pixel values of second reconstructed samples at a second node along the loop filter, the second boundary pixel values being values of the pixels at the frame boundary;

combine the second reconstructed samples at the second node with sample offsets generated by the non linear mapping based filter to generate the to-be filtered reconstructed samples; and

apply the loop restoration filter on the to-be filtered reconstructed samples based on the buffered first boundary pixel values, and the buffered second boundary pixel values.

18. The apparatus of claim 11, wherein the processing circuitry is configured to:

buffer second boundary pixel values of second reconstructed samples generated by a deblocking filter, the second boundary pixel values being values of the pixels at the frame boundary;

apply a constrained directional enhanced filter on the second reconstructed samples to generate intermediate reconstructed samples; and

combine the intermediate reconstructed samples with sample offsets generated by the non linear mapping based filter to generate the first reconstructed samples.

19. The apparatus of claim 11, wherein the processing circuitry is configured to:

clip the to-be filtered reconstructed samples before the applying of the loop restoration filter.

20. The apparatus of claim 19, wherein the processing circuitry is configured to:

clip intermediate reconstructed samples before a combination with sample offsets generated by the non linear mapping based filter.

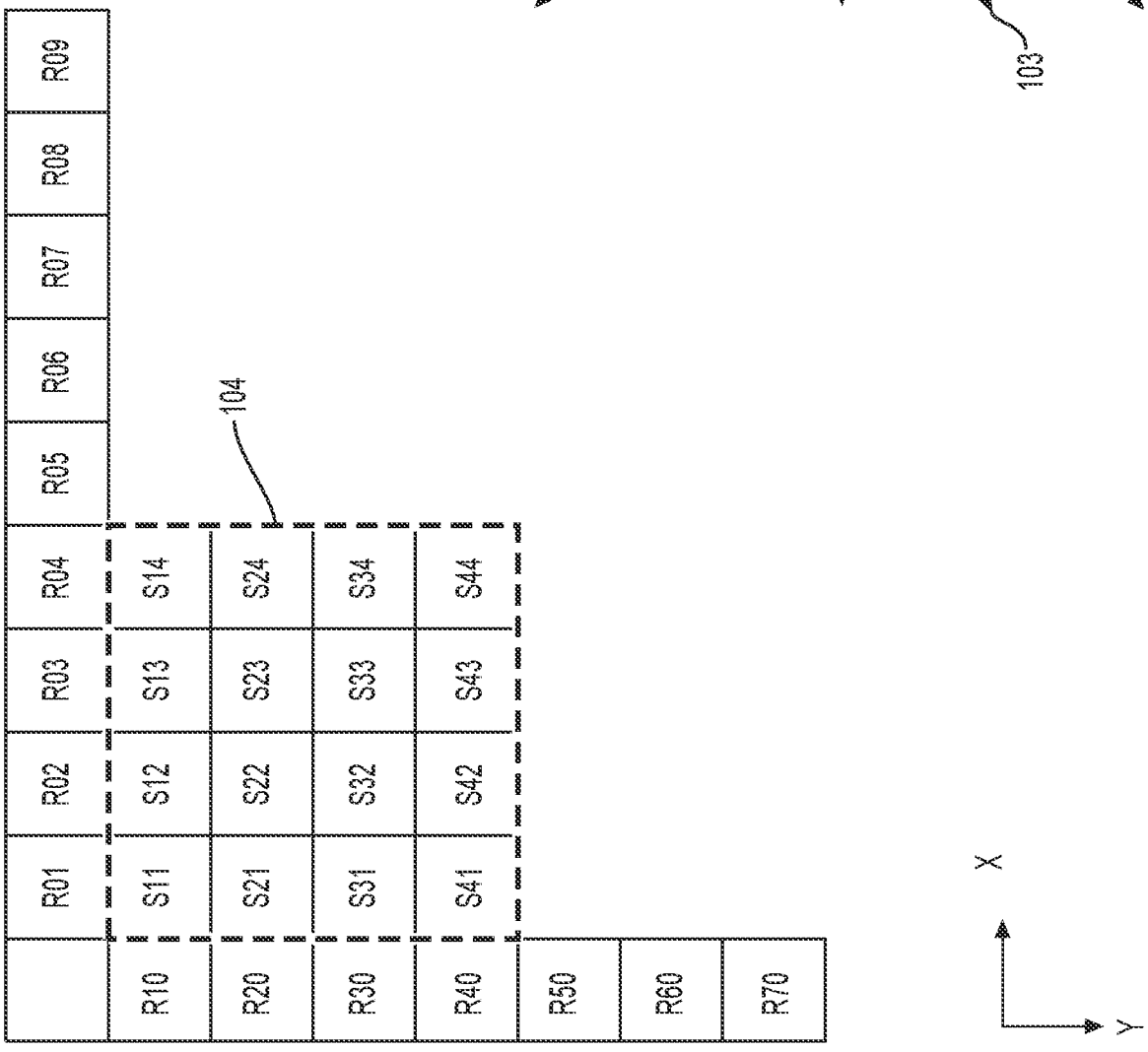

FIG. 1A
(Related art)

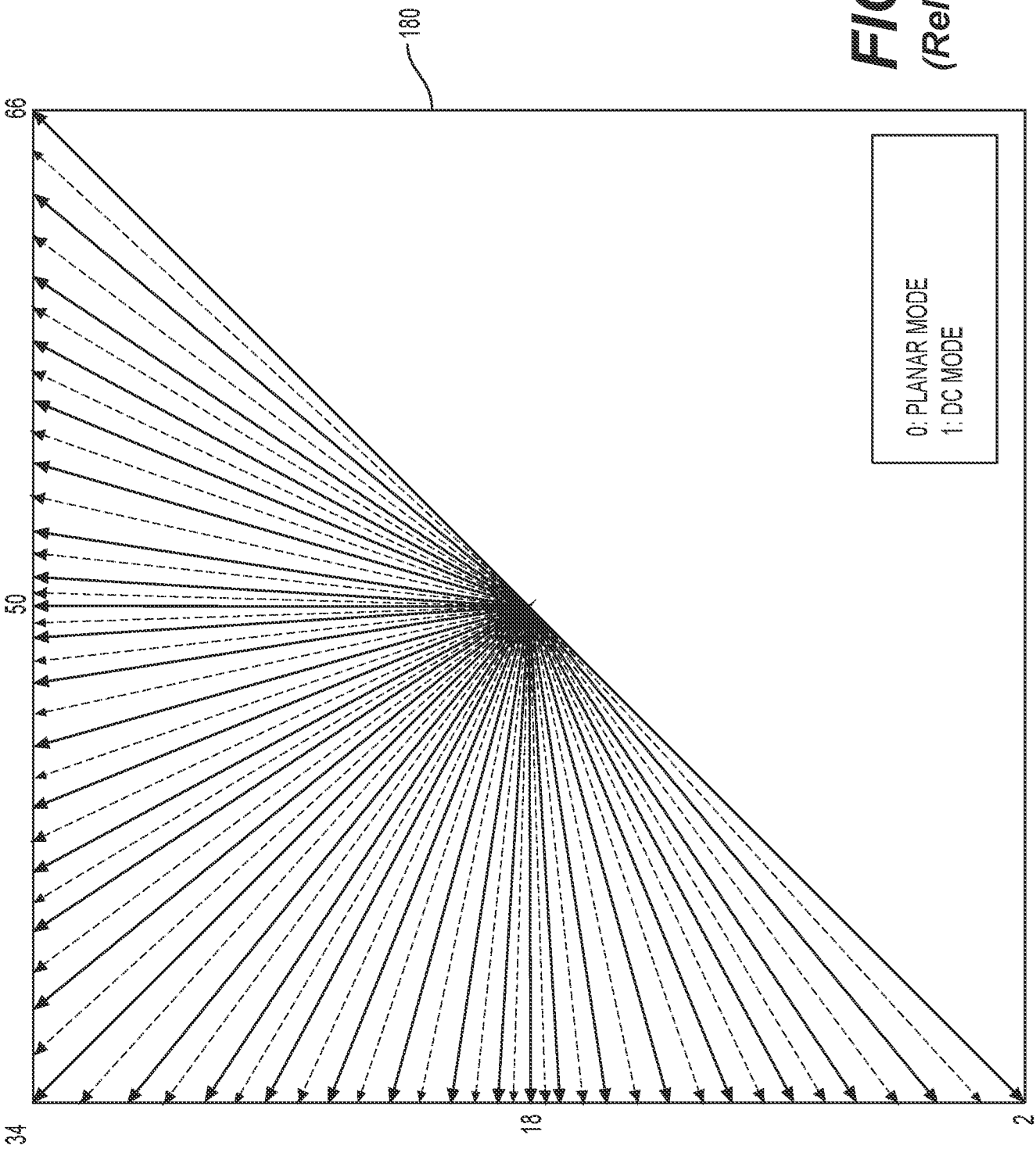

FIG. 1B
(Related art)

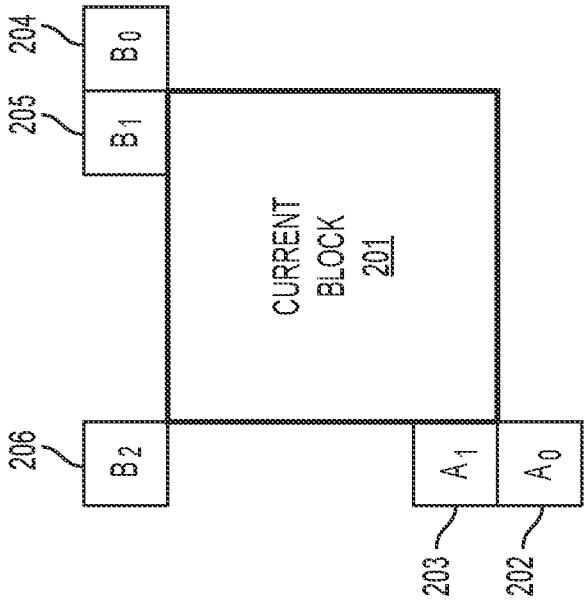

FIG. 2
(Related Art)

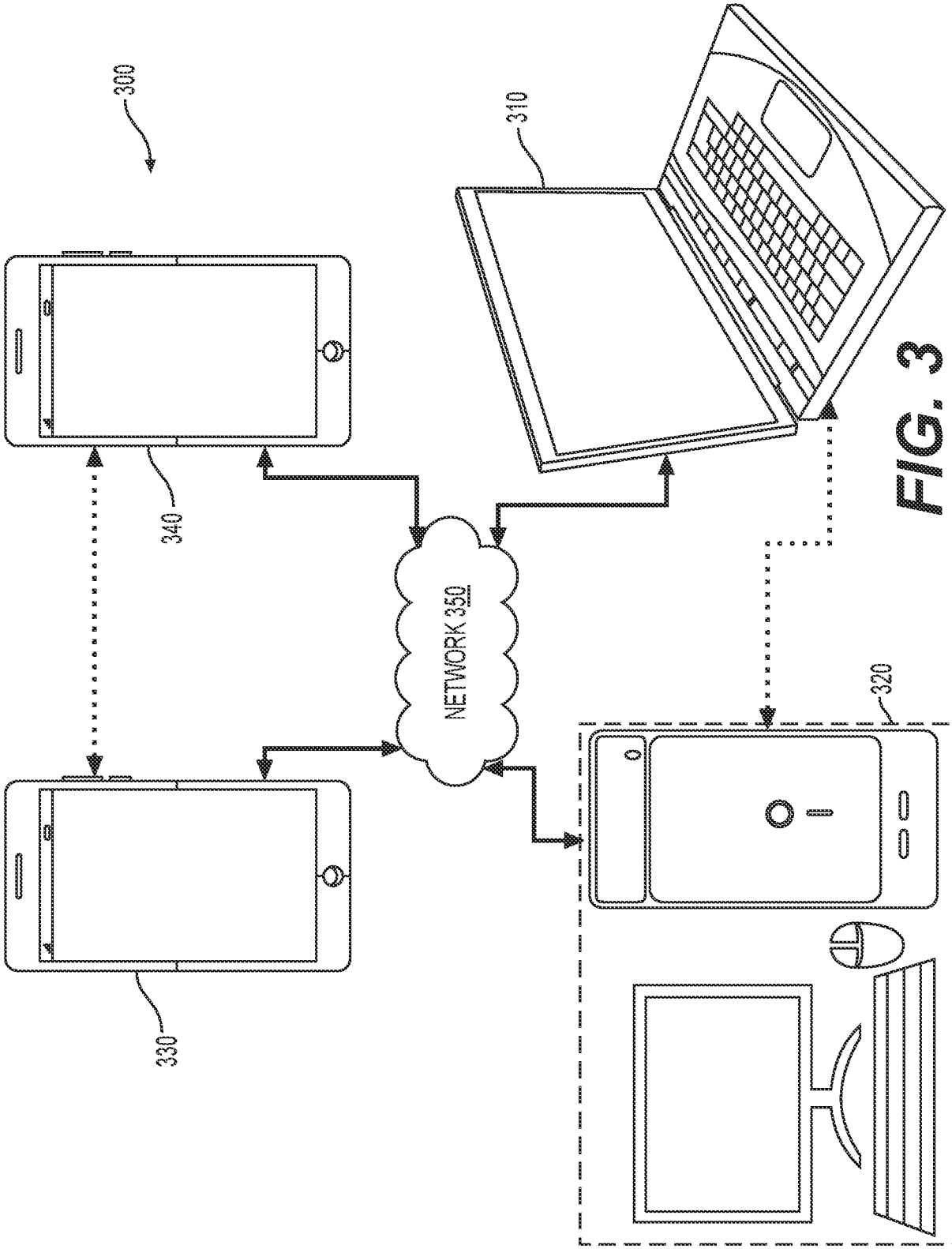

FIG. 3

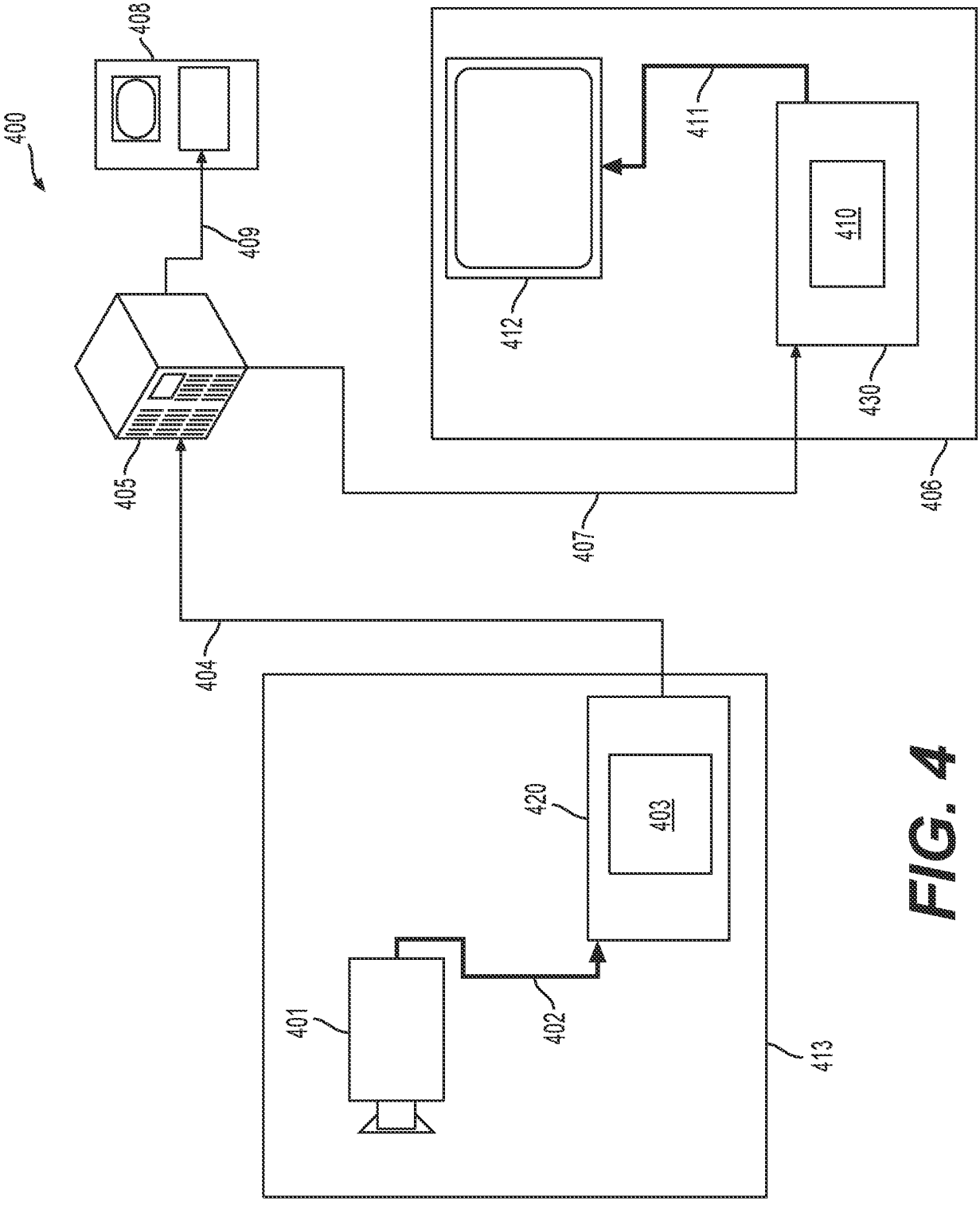

FIG. 4

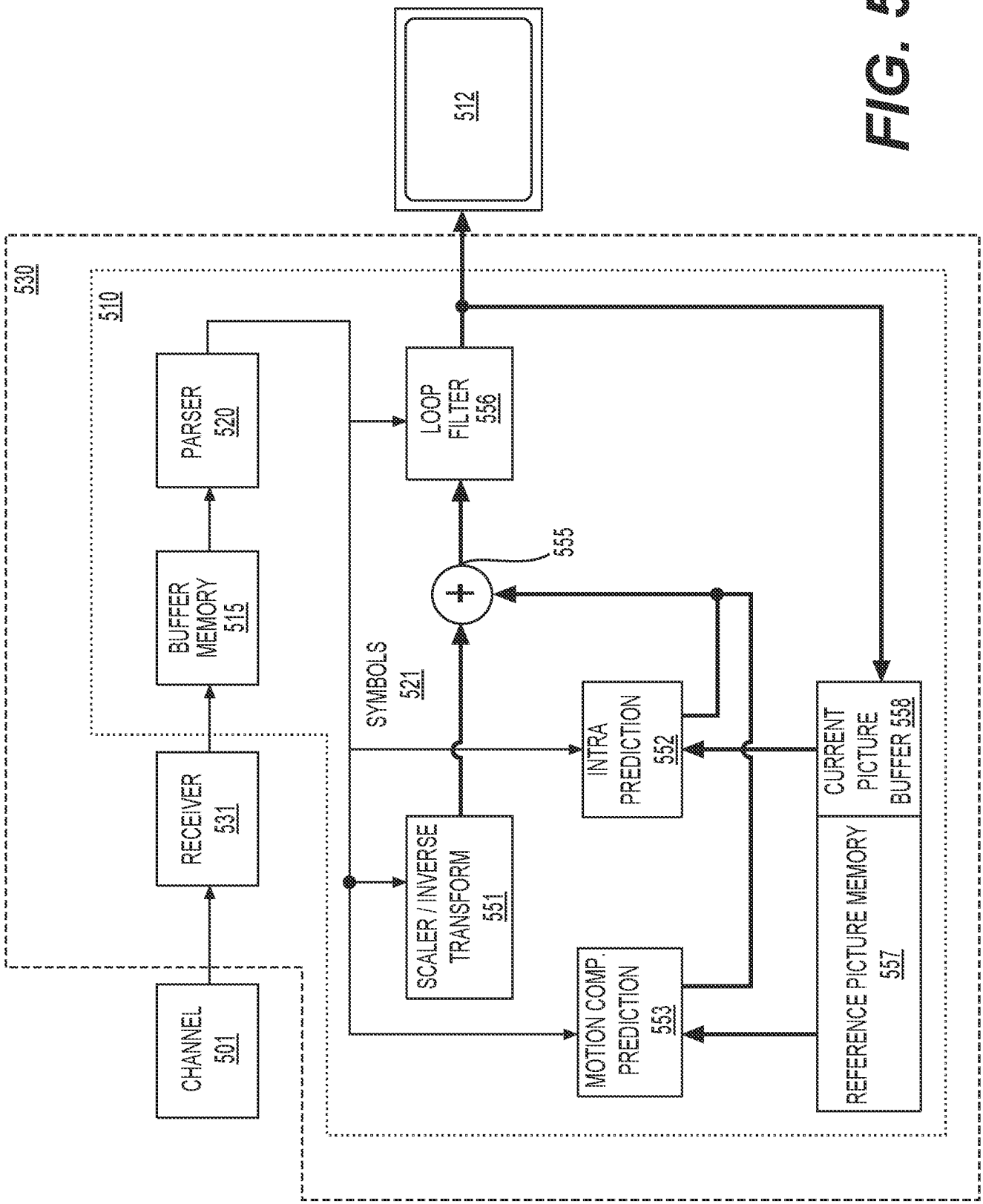
512
530
510
PARSER 520
BUFFER MEMORY 515
RECEIVER 531
CHANNEL 501
LOOP FILTER 556
555
SYMBOLS 521
SCALER / INVERSE TRANSFORM 551
INTRA PREDICTION 552
MOTION COMP. PREDICTION 553
CURRENT PICTURE BUFFER 558
REFERENCE PICTURE MEMORY 557

FIG. 5

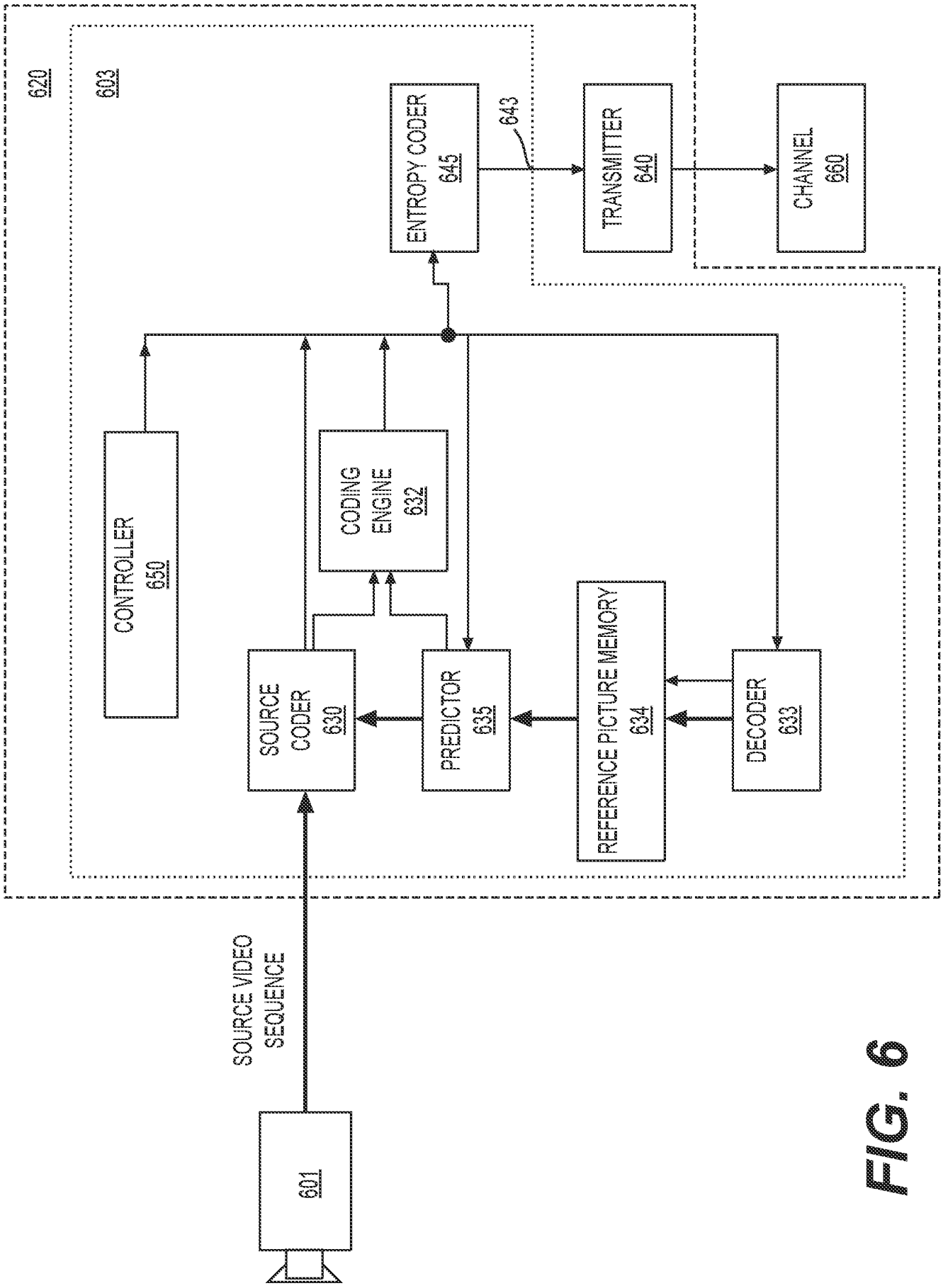
620
603
ENTROPY CODER 645
643
TRANSMITTER 640
CHANNEL 660
CONTROLLER 650
CODING ENGINE 632
SOURCE CODER 630
PREDICTOR 635
REFERENCE PICTURE MEMORY 634
DECODER 633
SOURCE VIDEO SEQUENCE
601

FIG. 6

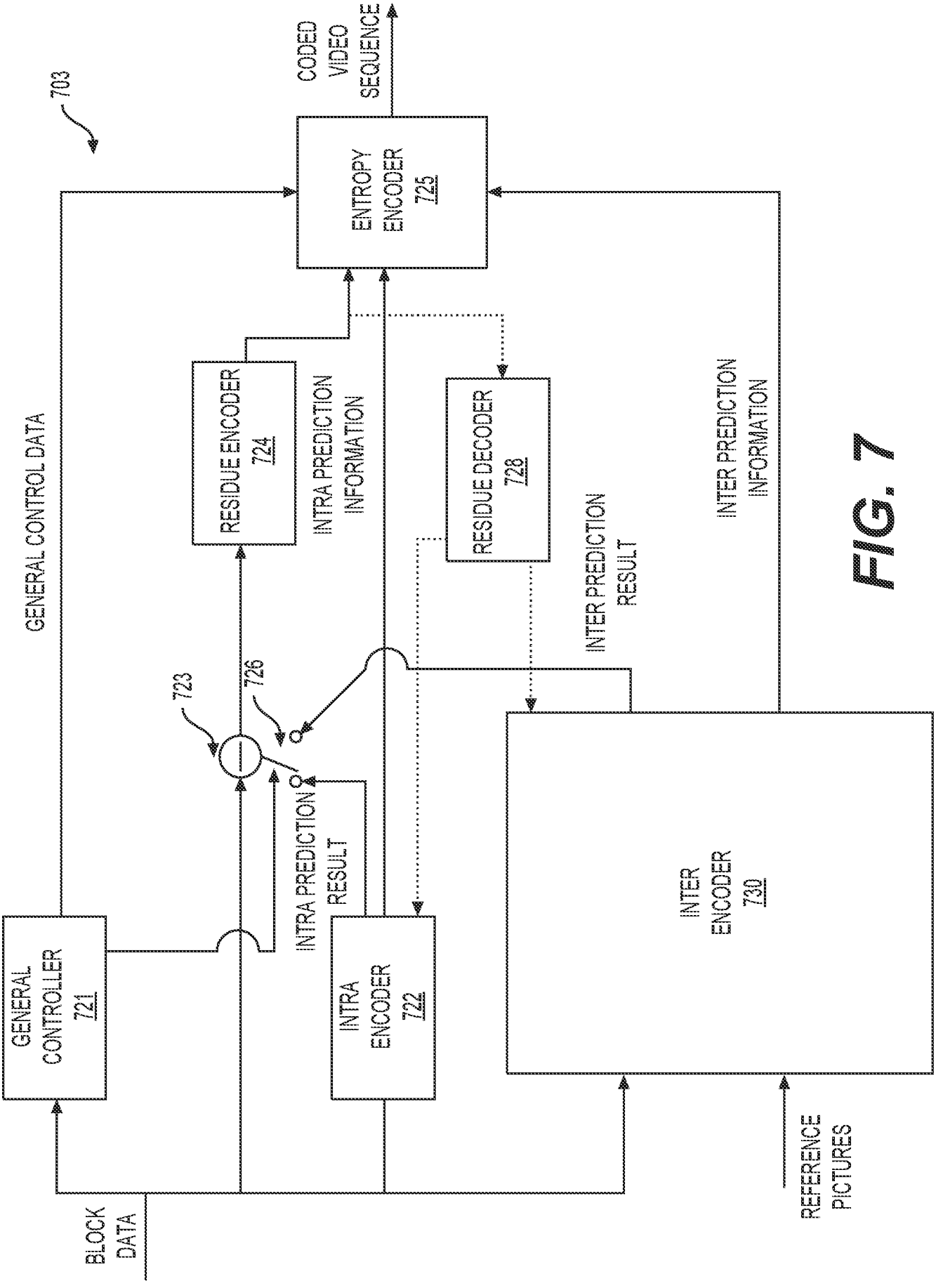
703
CODED VIDEO SEQUENCE
ENTROPY ENCODER 725
GENERAL CONTROL DATA
RESIDUE ENCODER 724
INTRA PREDICTION INFORMATION
RESIDUE DECODER 728
INTER PREDICTION INFORMATION
INTER PREDICTION RESULT
723
726
INTRA PREDICTION RESULT
GENERAL CONTROLLER 721
INTRA ENCODER 722
INTER ENCODER 730
BLOCK DATA
REFERENCE PICTURES

FIG. 7

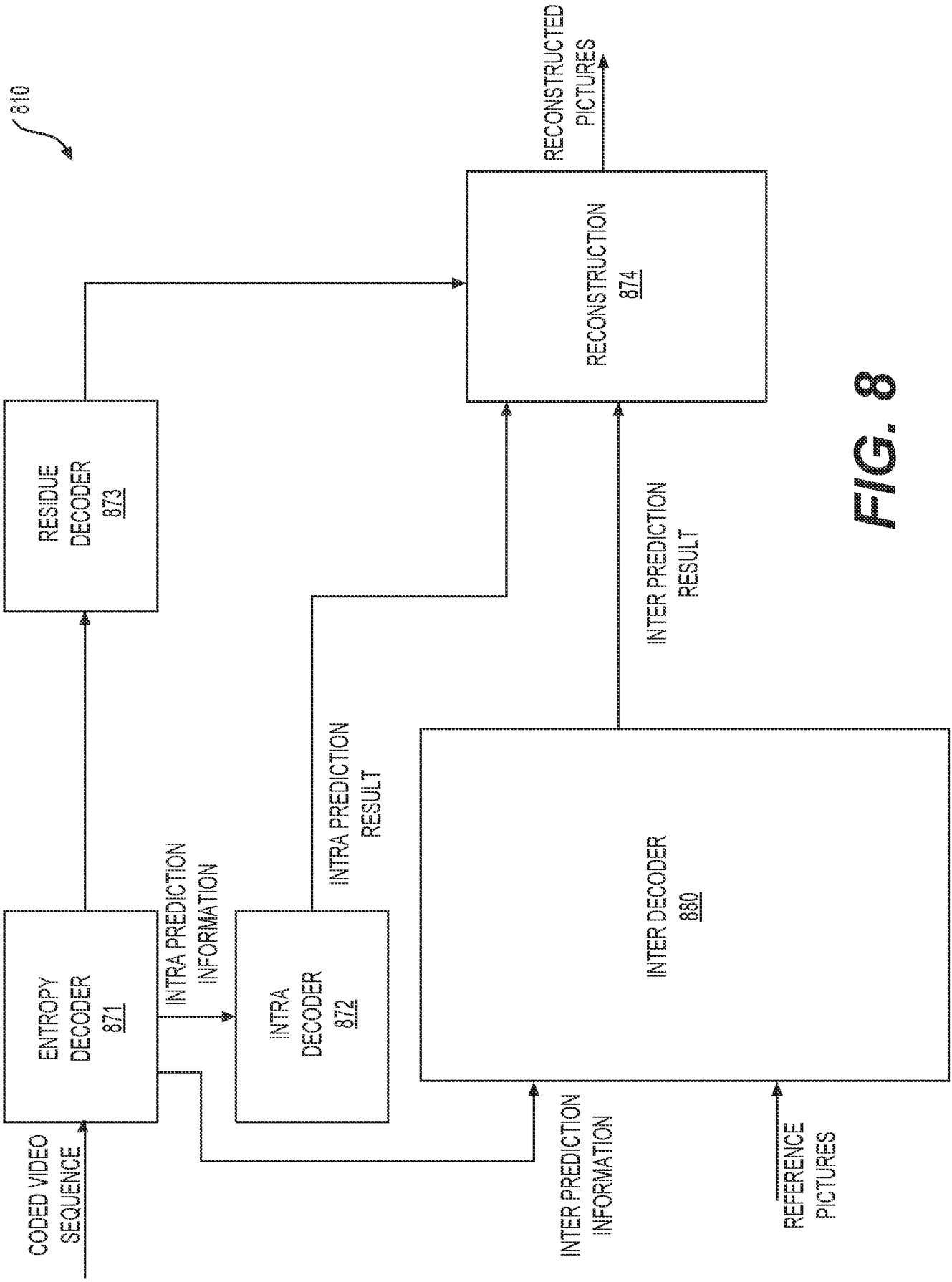
810
RECONSTRUCTED PICTURES
RECONSTRUCTION 874
RESIDUE DECODER 873
INTER PREDICTION RESULT
INTRA PREDICTION RESULT
INTER DECODER 880
ENTROPY DECODER 871
INTRA PREDICTION INFORMATION
INTRA DECODER 872
CODED VIDEO SEQUENCE
INTER PREDICTION INFORMATION
REFERENCE PICTURES

FIG. 8

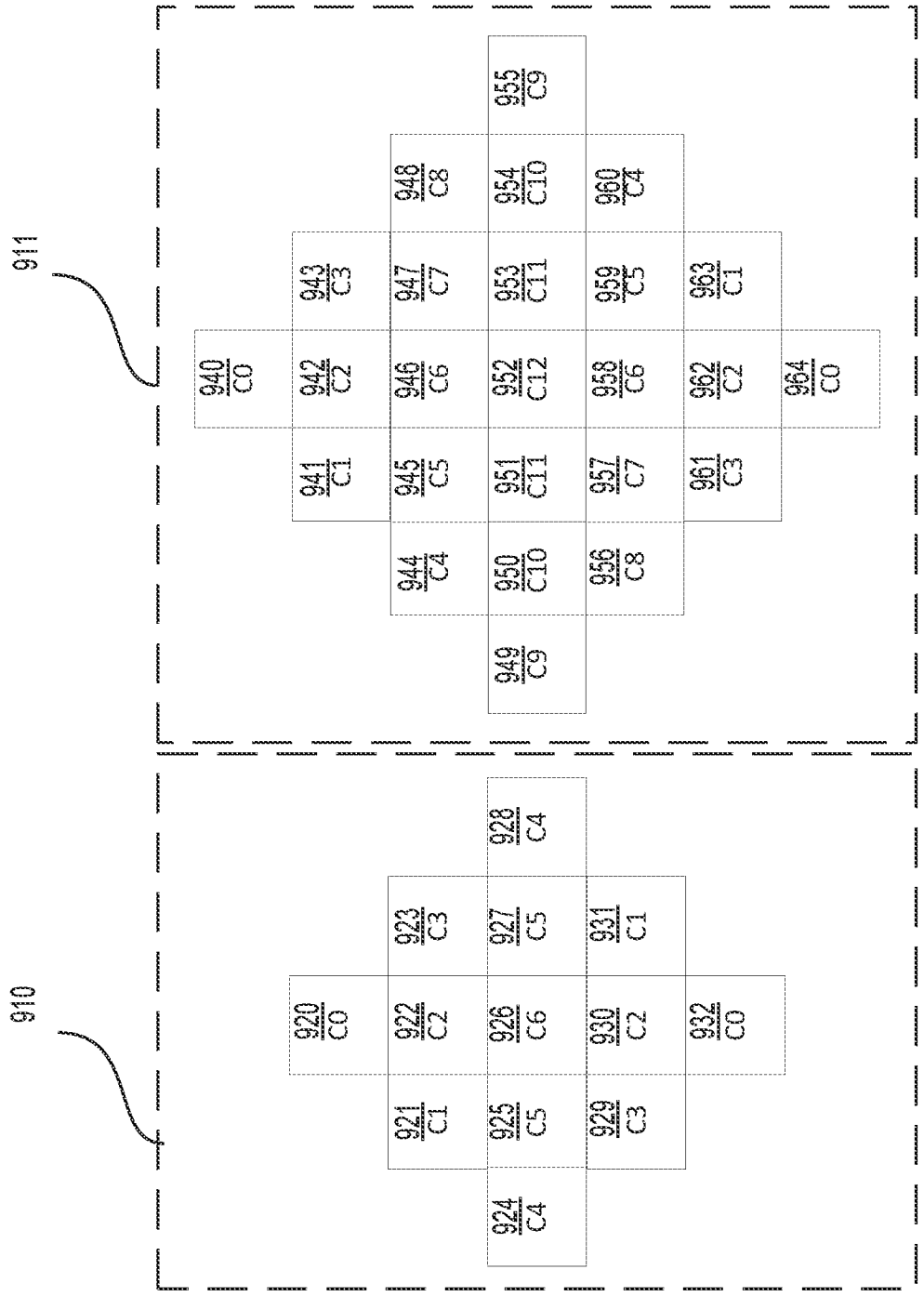
911
940 C0
941 C1
942 C2
943 C3
944 C4
945 C5
946 C6
947 C7
948 C8
949 C9
950 C10
951 C11
952 C12
953 C11
954 C10
955 C9
956 C8
957 C7
958 C6
959 C5
960 C4
961 C3
962 C2
963 C1
964 C0
910
920 C0
921 C1
922 C2
923 C3
924 C4
925 C5
926 C6
927 C5
928 C4
929 C3
930 C2
931 C1
932 C0

FIG. 9

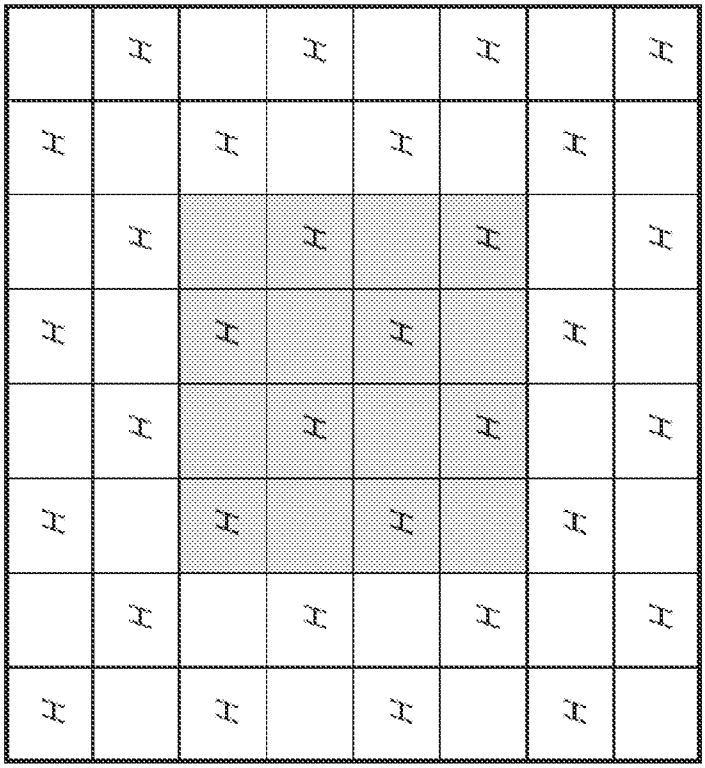

FIG. 10B

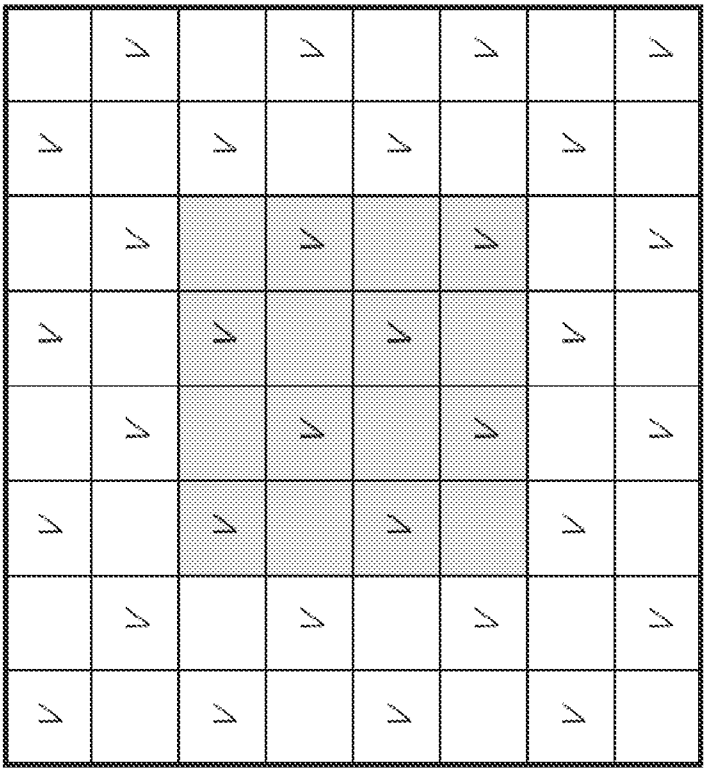

FIG. 10A

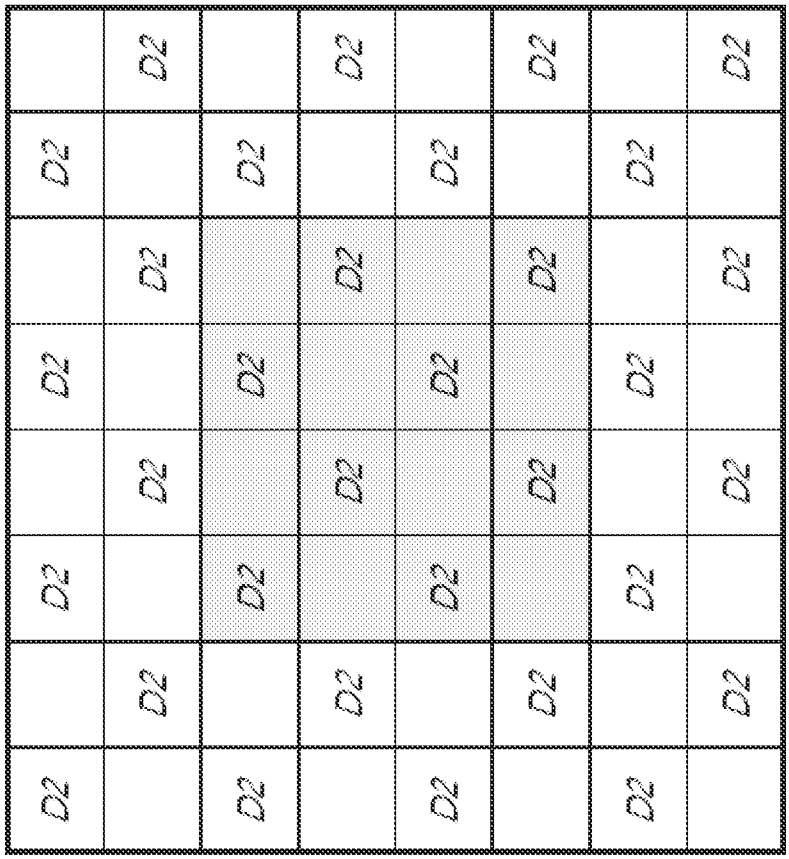

FIG. 10D

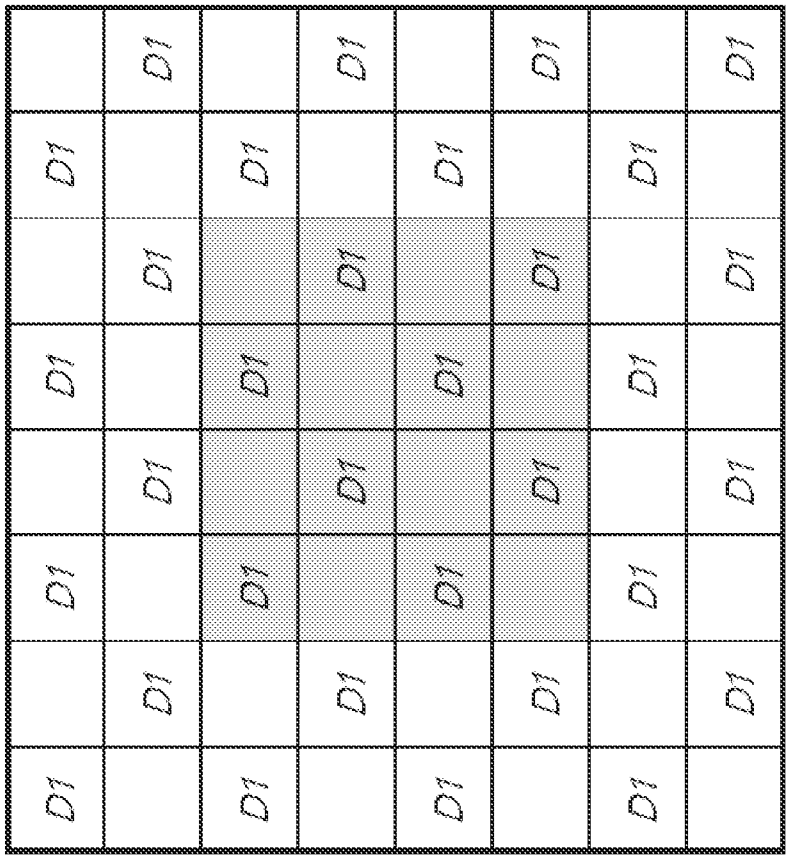

FIG. 10C

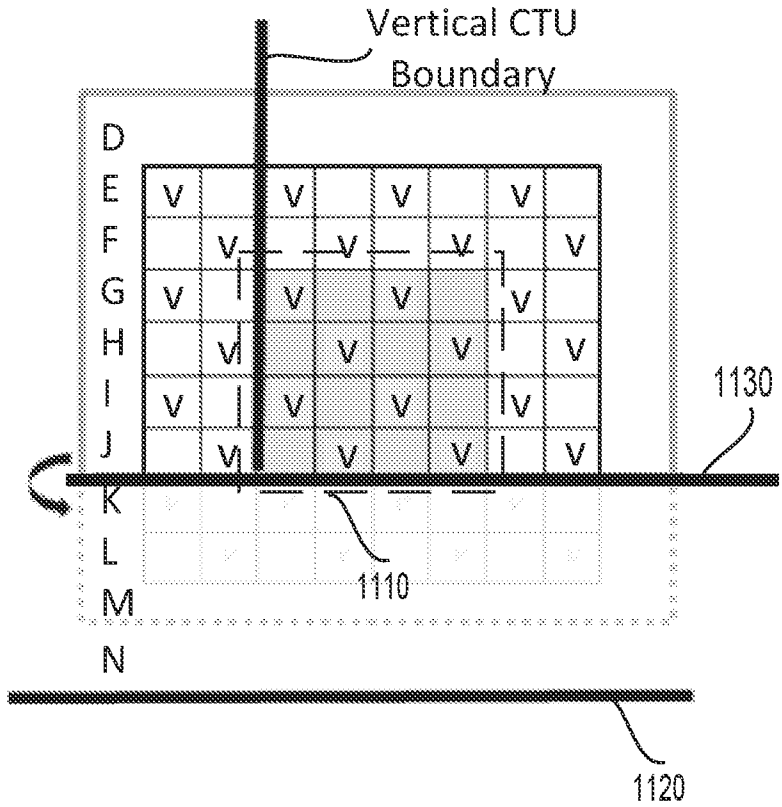

FIG. 11A

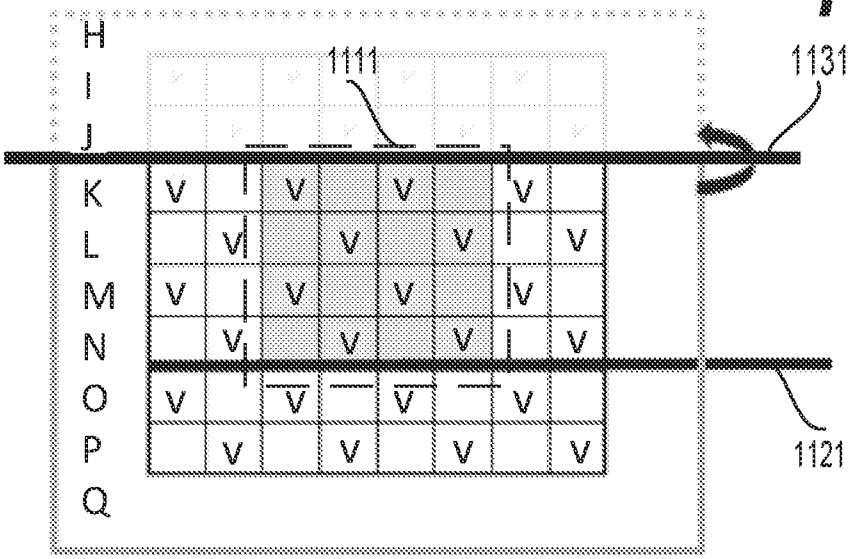

FIG. 11B

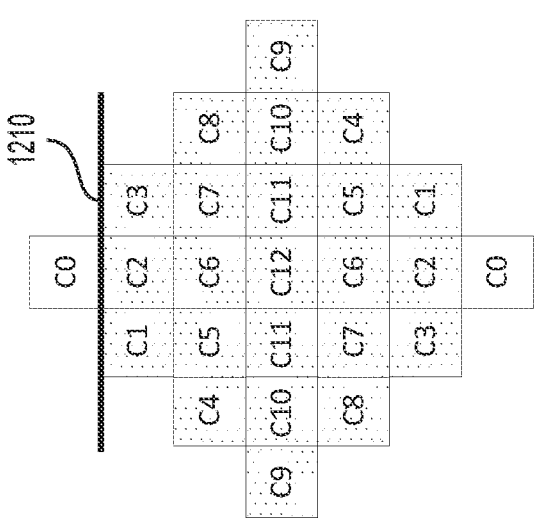

FIG. 12A

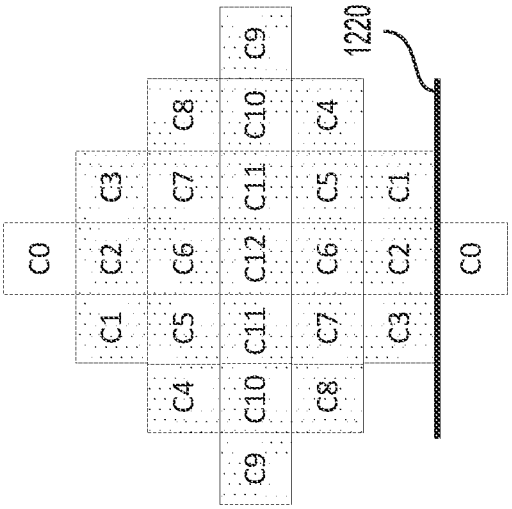

FIG. 12B

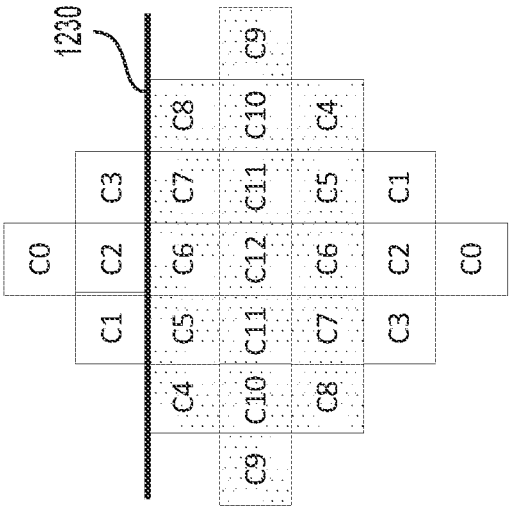

FIG. 12C

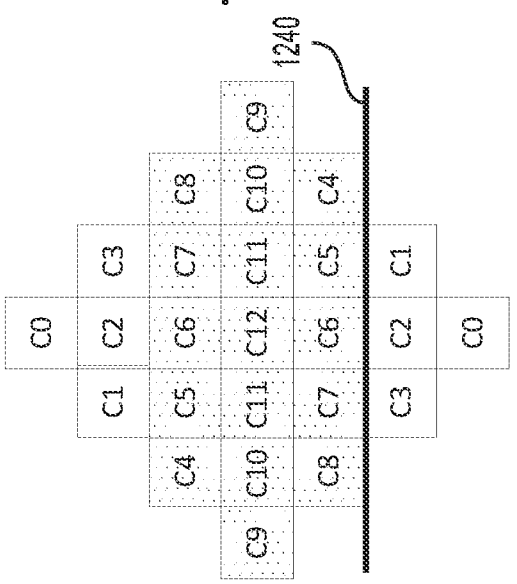

FIG. 12D

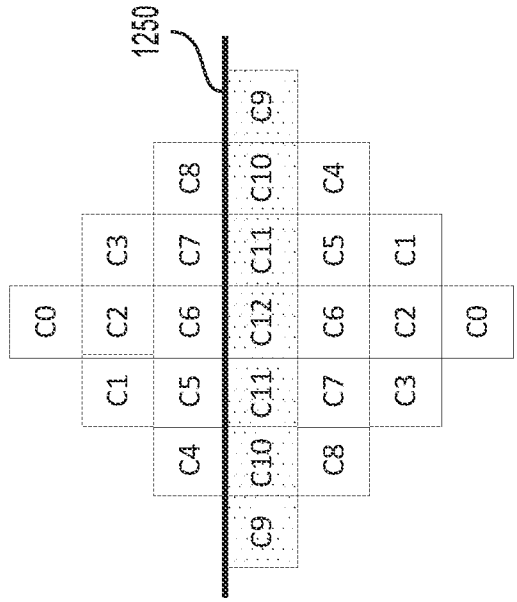

FIG. 12E

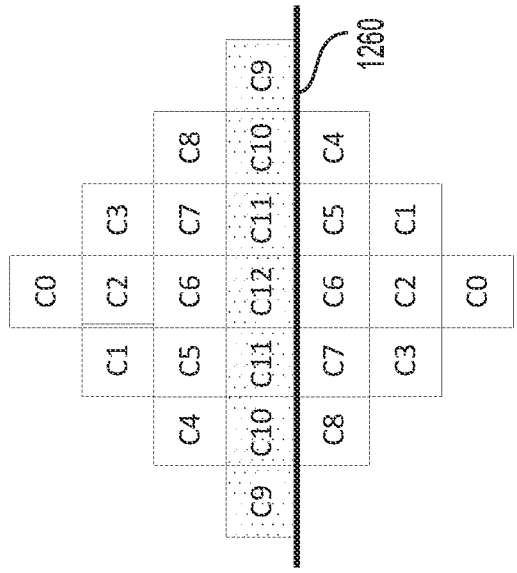

FIG. 12F

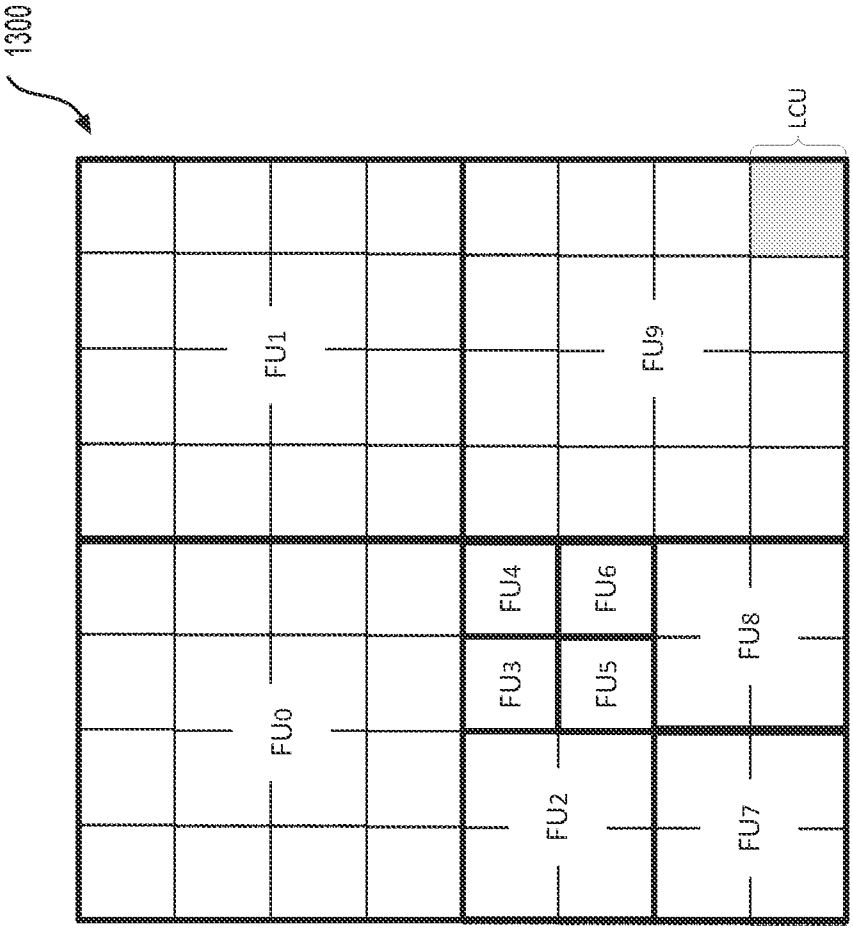

FIG. 13

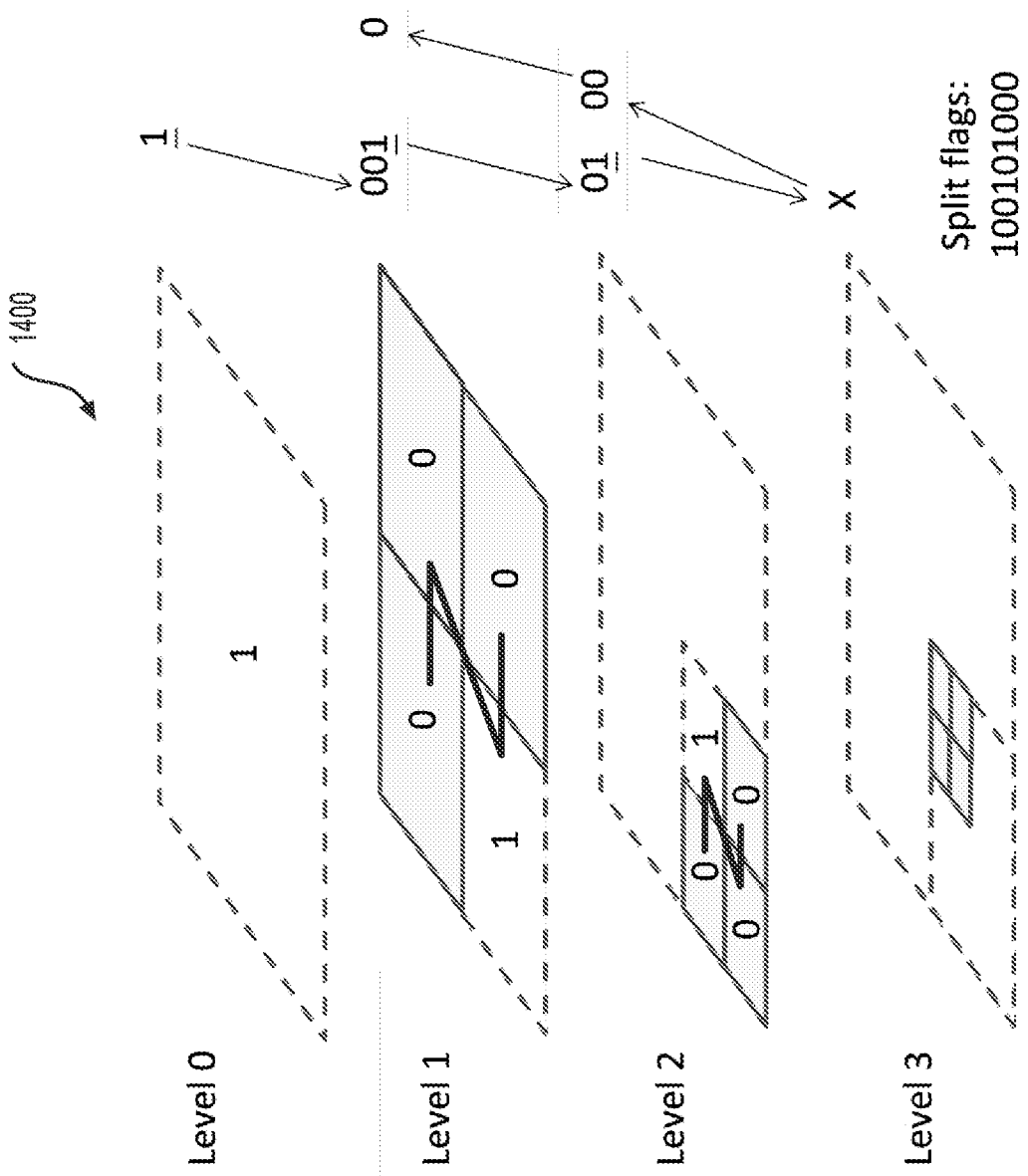
1400
1
001
01
0
00
X
Split flags:
10010100
Level 0
Level 1
Level 2
Level 3

FIG. 14

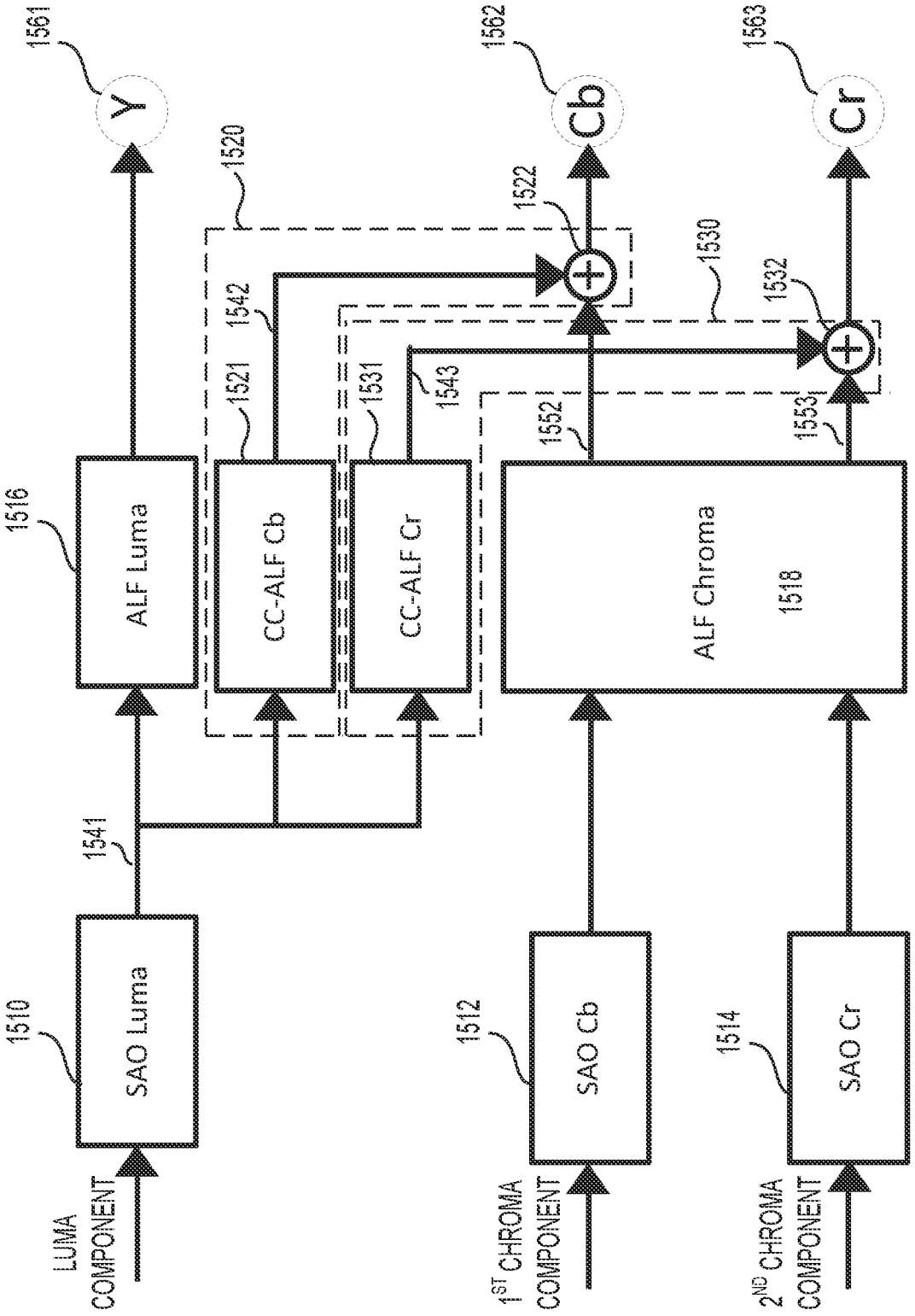
LUMA COMPONENT
1ST CHROMA COMPONENT
2ND CHROMA COMPONENT
SAO Luma
SAO Cb
SAO Cr
ALF Luma
CC-ALF Cb
CC-ALF Cr
ALF Chroma
1518
Y
Cb
Cr
1510
1512
1514
1516
1520
1521
1522
1530
1531
1532
1541
1542
1543
1552
1553
1561
1562
1563

FIG. 15

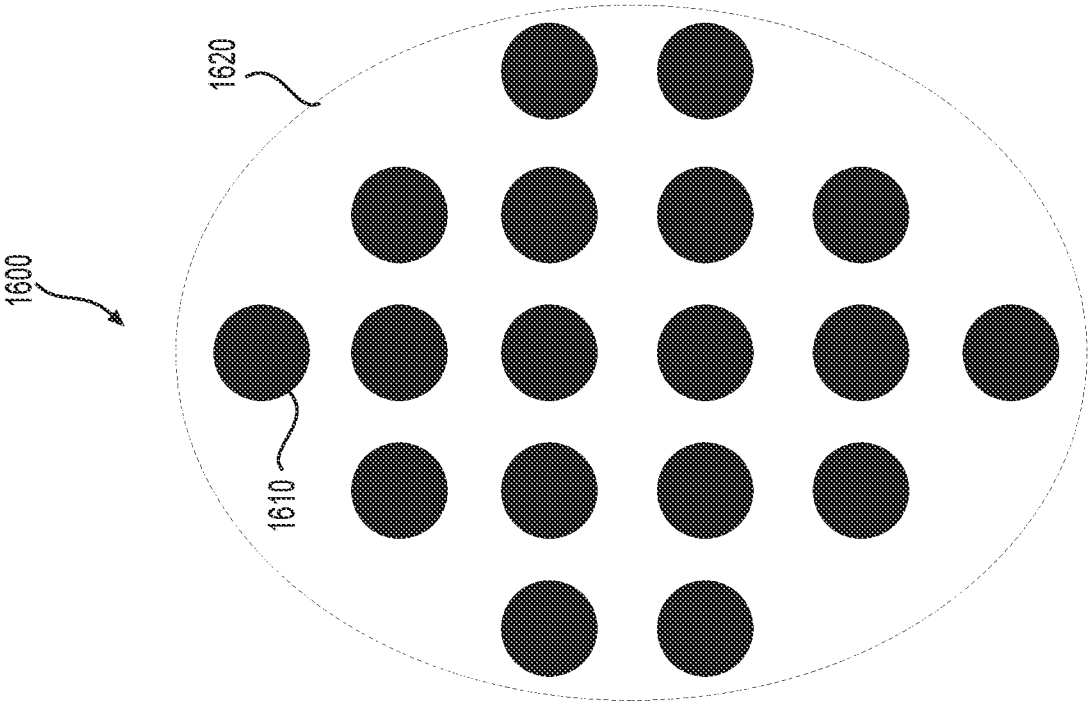

FIG. 16

if (slice_cross_component_alf_cb_enabled_flag)	
alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(slice_cross_component_alf_cb_enabled_flag = = 0 || alf_ctb_cross_component_cb_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] == 0)	
if(slice_alf_chroma_idc = = 1 || slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(alf_ctb_flag[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] && aps_alf_chroma_num_alt_filters_minus1 > 0)	
alf_ctb_filter_alt_idx[0][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
}	
if (slice_cross_component_alf_cr_enabled_flag)	
alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(slice_cross_component_alf_cr_enabled_flag = = 0 || alf_ctb_cross_component_cr_idc[xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] == 0)	
if(slice_alf_chroma_idc = = 2 || slice_alf_chroma_idc = = 3) {	
alf_ctb_flag[2][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
if(alf_ctb_flag[2][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY] && aps_alf_chroma_num_alt_filters_minus1 > 0)	
alf_ctb_filter_alt_idx[1][xCtb >> CtbLog2SizeY][yCtb >> CtbLog2SizeY]	ae(v)
}	

FIG. 17

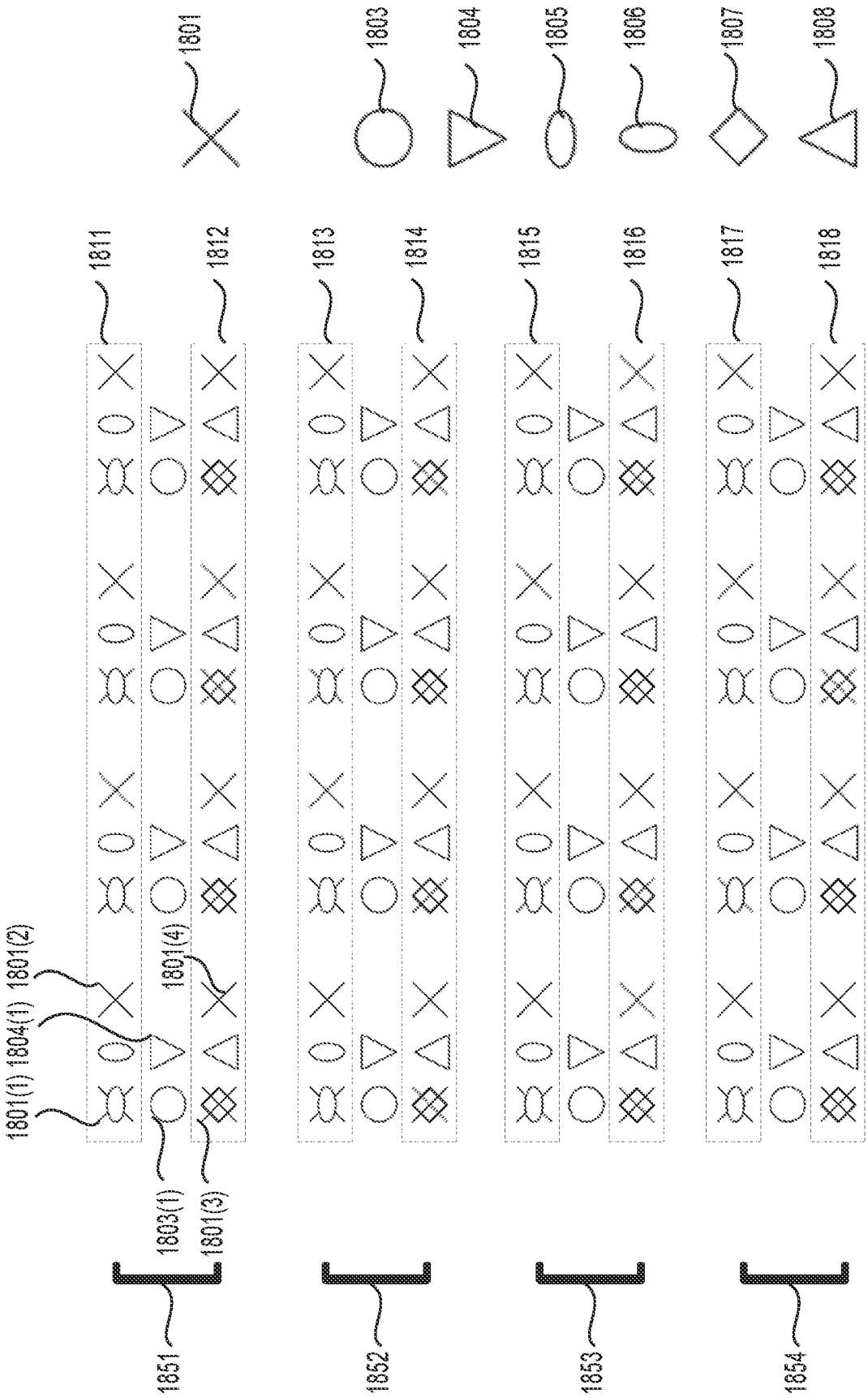
1801
1803
1804
1805
1806
1807
1808
1811
1812
1813
1814
1815
1816
1817
1818
1801(1) 1804(1) 1801(2)
1801(4)
1803(1)
1801(3)
1851
1852
1853
1854

FIG. 18A

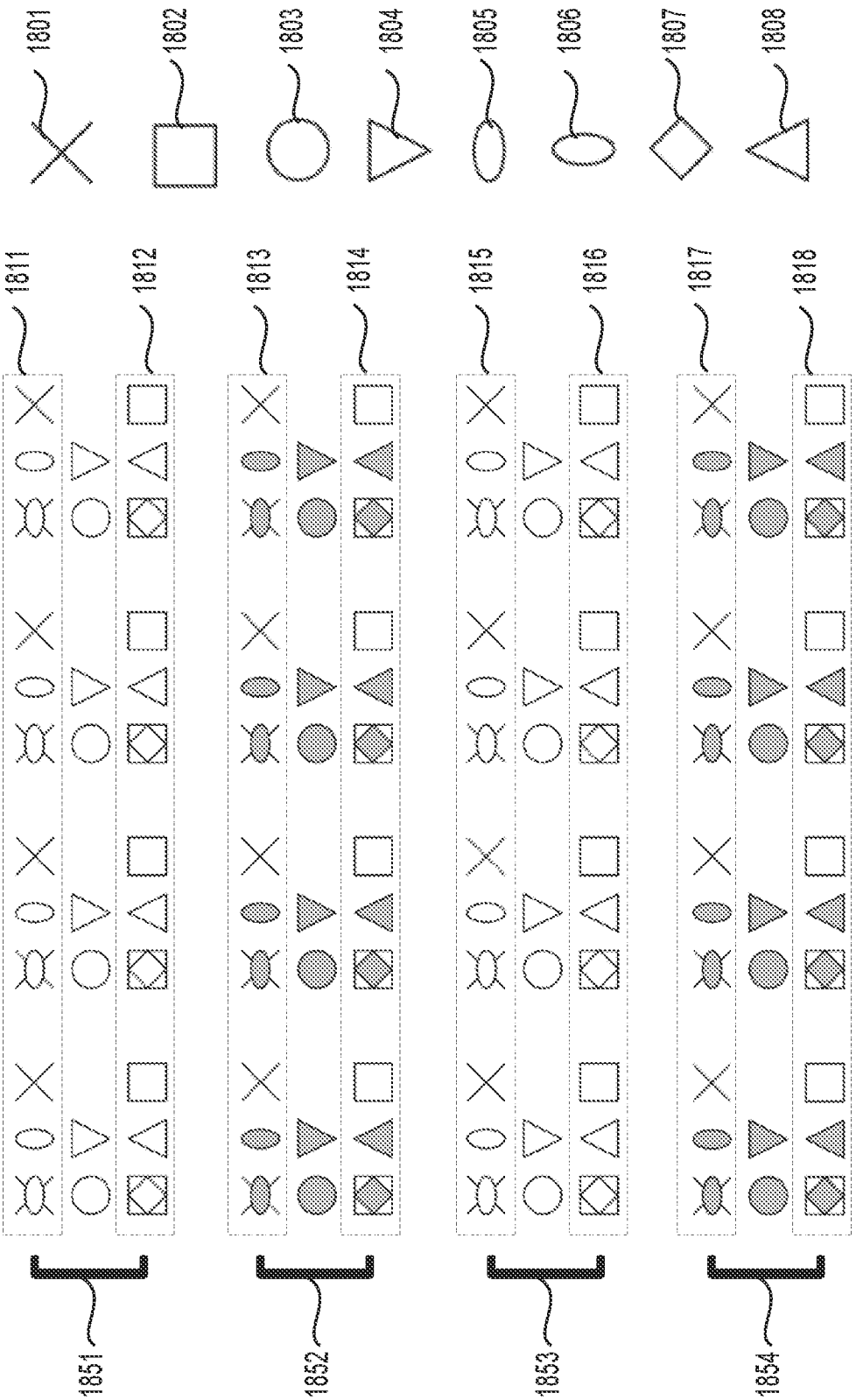
1801
1802
1803
1804
1805
1806
1807
1808
1811
1812
1813
1814
1815
1816
1817
1818
1851
1852
1853
1854

FIG. 18B

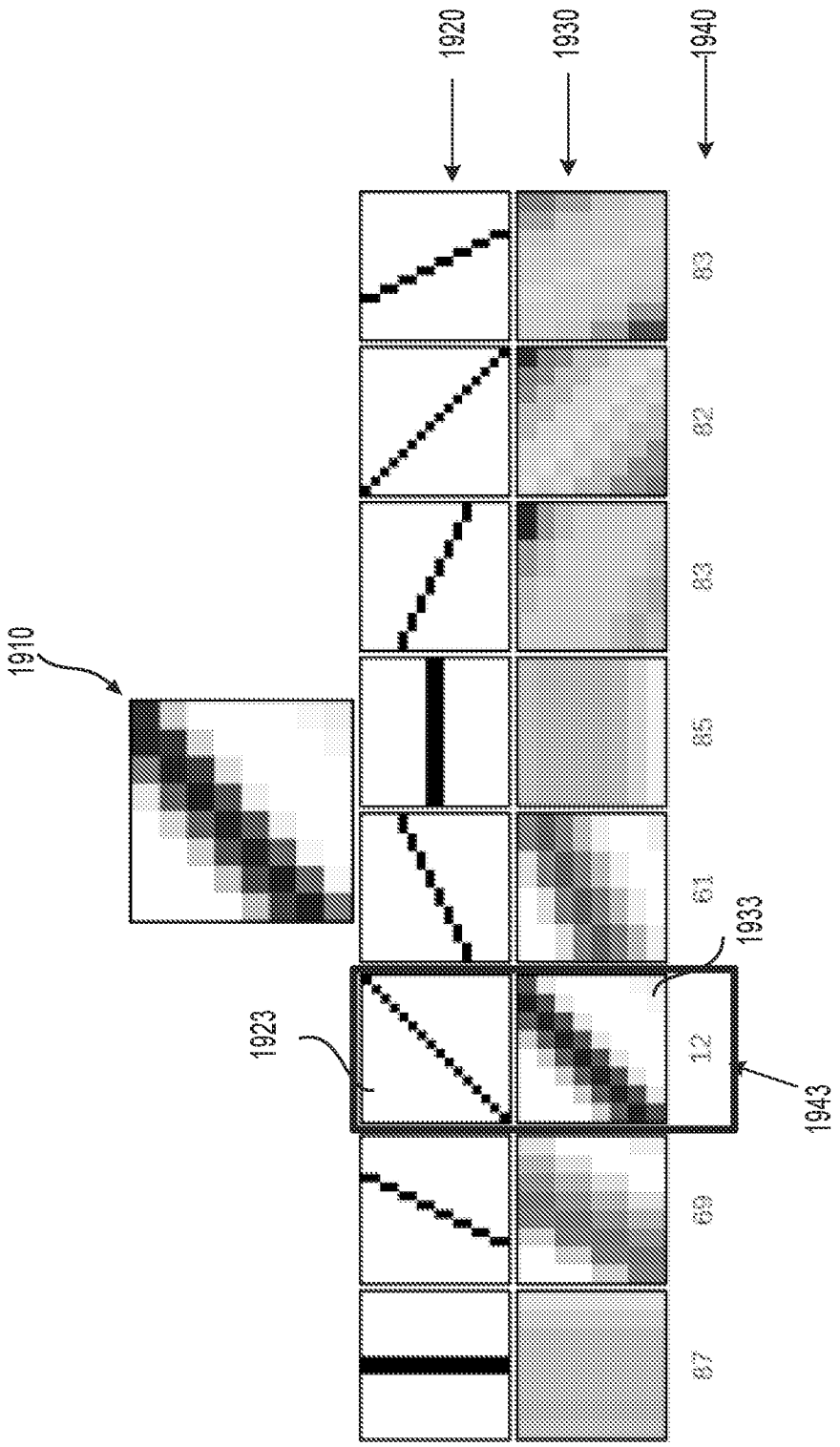

FIG. 19

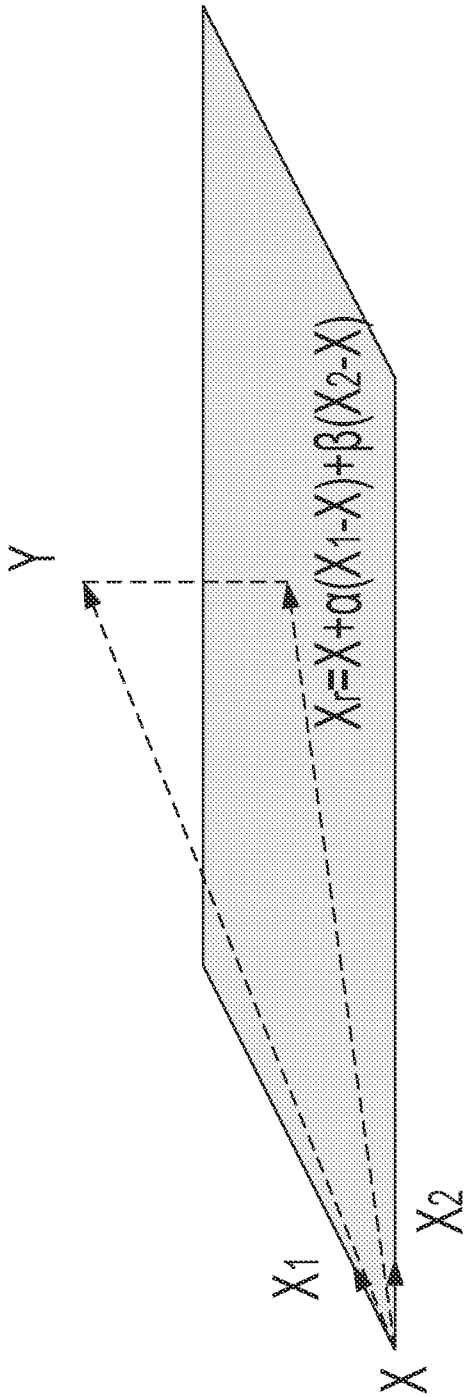

FIG. 20

2100

SAO Type	Sample Adaptive Offset Type to be used	Number of categories
0	None	0
1	1-D 0-degree pattern edge offset	4
2	1-D 90-degree pattern edge offset	4
3	1-D 135-degree pattern edge offset	4
4	1-D 45-degree pattern edge offset	4
5	central bands band offset	16
6	side bands band offset	16

FIG. 21

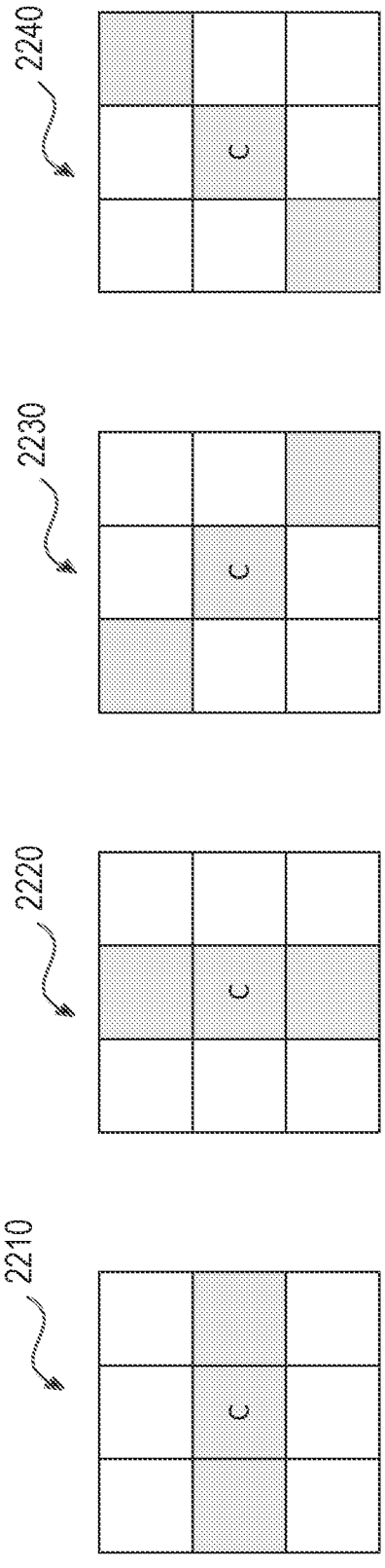

FIG. 22

2300

Category	Condition
1	c < 2 neighbors
2	c < 1 neighbor && c == 1 neighbor
3	c > 1 neighbor && c == 1 neighbor
4	c > 2 neighbors
0	None of the above

FIG. 23

2400

sao_offset_vlc(rx, ry, cIdx) {	**Descriptor**
sao_type_idx[cIdx][rx][ry]	ue(v)
if(sao_type_idx[cIdx][rx][ry] = =5) {	
sao_band_position[cIdx][rx][ry]	u(5)
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	se(v)
} else if(sao_type_idx[cIdx][rx][ry] != 0)	
for(i = 0; i < 4; i++)	
sao_offset[cIdx][rx][ry][i]	ue(v)
}	

FIG. 24

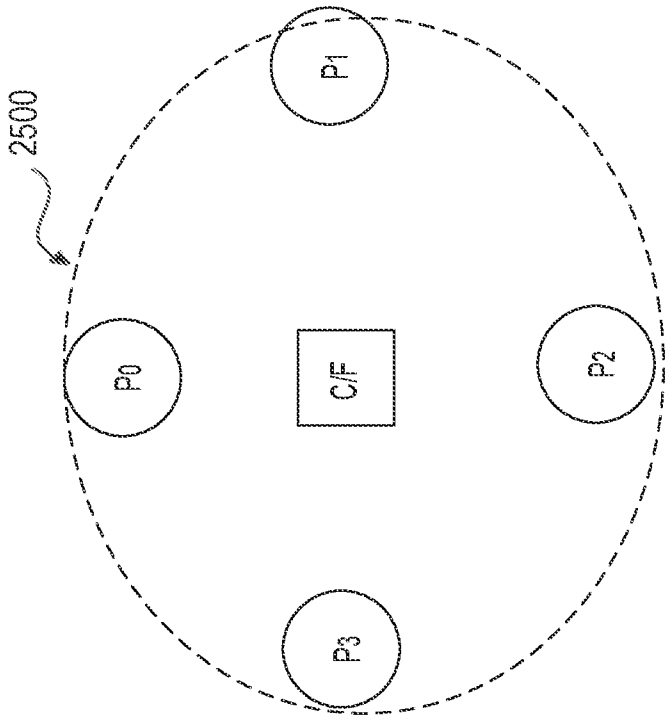

FIG. 25

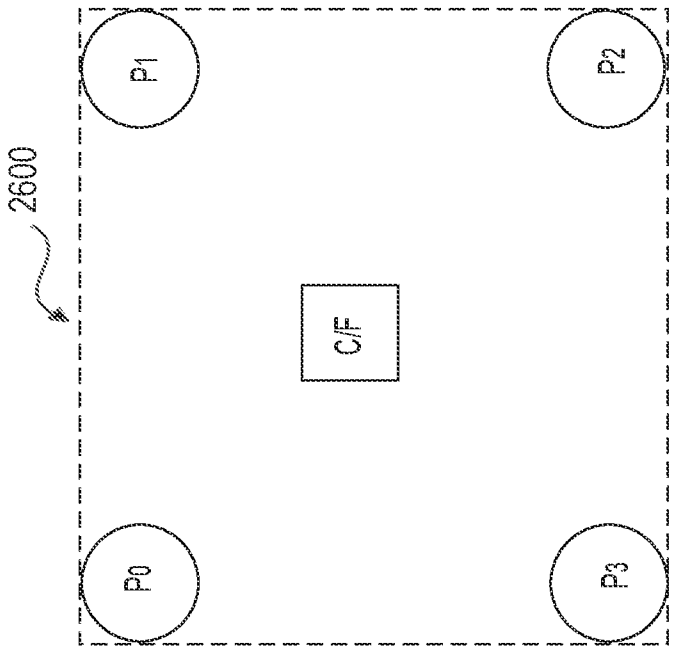

FIG. 26

2700

Combination	d0	d1	d2	d3	Offset
0	-1	-1	-1	-1	s0
1	-1	-1	-1	0	s1
2	-1	-1	-1	1	s2
3	-1	-1	0	-1	s3
4	-1	-1	0	0	s4
5	-1	-1	0	1	s5
6	-1	-1	1	-1	s6
7	-1	-1	1	0	s7
8	-1	-1	1	1	s8
9	-1	0	-1	-1	s9
10	-1	0	-1	0	s10
11	-1	0	-1	1	s11
12	-1	0	0	-1	s12
13	-1	0	0	0	s13
14	-1	0	0	1	s14
15	-1	0	1	-1	s15
16	-1	0	1	0	s16
17	-1	0	1	1	s17
18	-1	1	-1	-1	s18
19	-1	1	-1	0	s19
20	-1	1	-1	1	s20
21	-1	1	0	-1	s21
22	-1	1	0	0	s22
23	-1	1	0	1	s23
24	-1	1	1	-1	s24
25	-1	1	1	0	s25
26	-1	1	1	1	s26

FIG. 27A

2700

Combination	d0	d1	d2	d3	Offset
27	0	-1	-1	-1	s27
28	0	-1	-1	0	s28
29	0	-1	-1	1	s29
30	0	-1	0	-1	s30
31	0	-1	0	0	s31
32	0	-1	0	1	s32
33	0	-1	1	-1	s33
34	0	-1	1	0	s34
35	0	-1	1	1	s35
36	0	0	-1	-1	s36
37	0	0	-1	0	s37
38	0	0	-1	1	s38
39	0	0	0	-1	s39
40	0	0	0	0	s40
41	0	0	0	1	s41
42	0	0	1	-1	s42
43	0	0	1	0	s43
44	0	0	1	1	s44
45	0	1	-1	-1	s45
46	0	1	-1	0	s46
47	0	1	-1	1	s47
48	0	1	0	-1	s48
49	0	1	0	0	s49
50	0	1	0	1	s50
51	0	1	1	-1	s51
52	0	1	1	0	s52
53	0	1	1	1	s53

FIG. 27B

2700

Combination	d0	d1	d2	d3	Offset
54	1	-1	-1	-1	s54
55	1	-1	-1	0	s55
56	1	-1	-1	1	s56
57	1	-1	0	-1	s57
58	1	-1	0	0	s58
59	1	-1	0	1	s59
60	1	-1	1	-1	s60
61	1	-1	1	0	s61
62	1	-1	1	1	s62
63	1	0	-1	-1	s63
64	1	0	-1	0	s64
65	1	0	-1	1	s65
66	1	0	0	-1	s66
67	1	0	0	0	s67
68	1	0	0	1	s68
69	1	0	1	-1	s69
70	1	0	1	0	s70
71	1	0	1	1	s71
72	1	1	-1	-1	s72
73	1	1	-1	0	s73
74	1	1	-1	1	s74
75	1	1	0	-1	s75
76	1	1	0	0	s76
77	1	1	0	1	s77
78	1	1	1	-1	s78
79	1	1	1	0	s79
80	1	1	1	1	s80

FIG. 27C

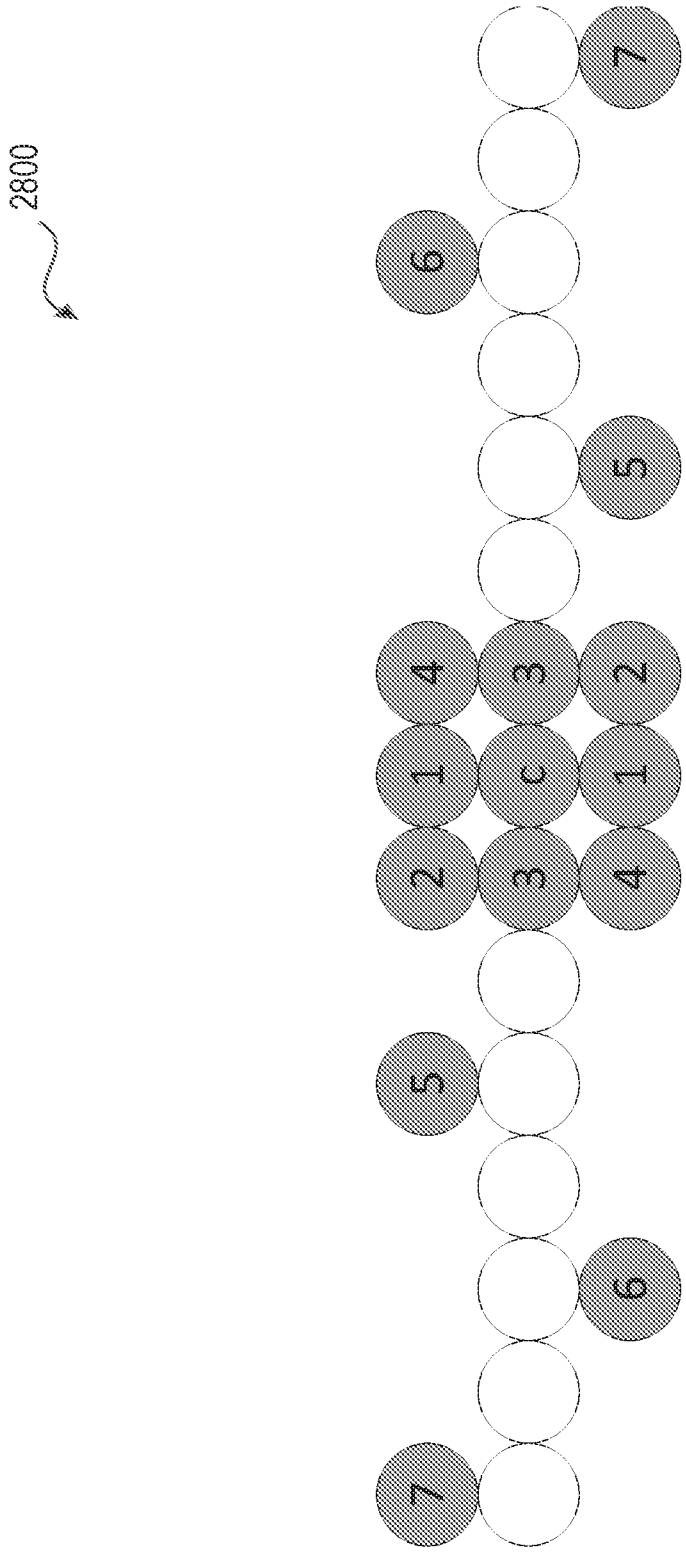

FIG. 28

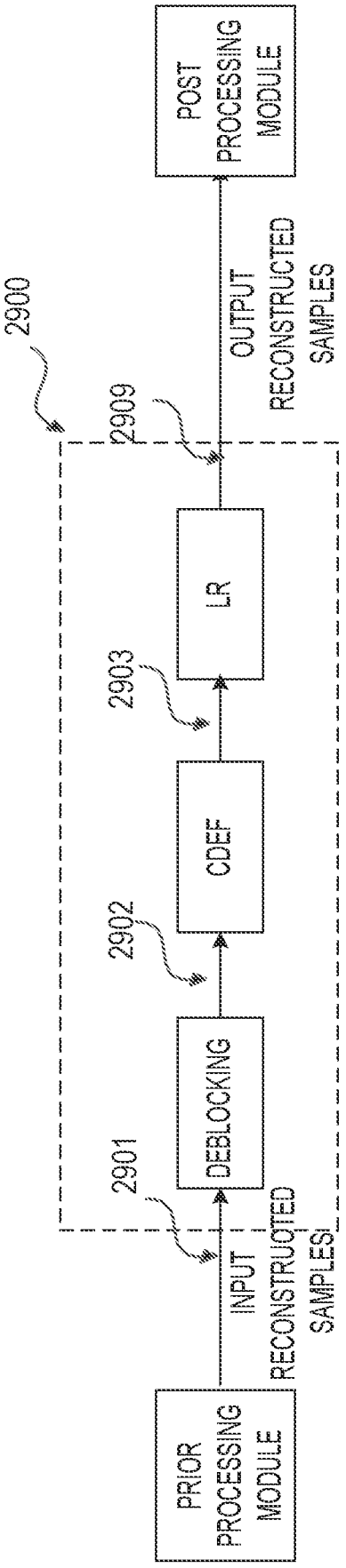
PRIOR PROCESSING MODULE
INPUT RECONSTRUCTED SAMPLES
2900
2901
DEBLOCKING
2902
CDEF
2903
LR
2909
OUTPUT RECONSTRUCTED SAMPLES
POST PROCESSING MODULE

FIG. 29

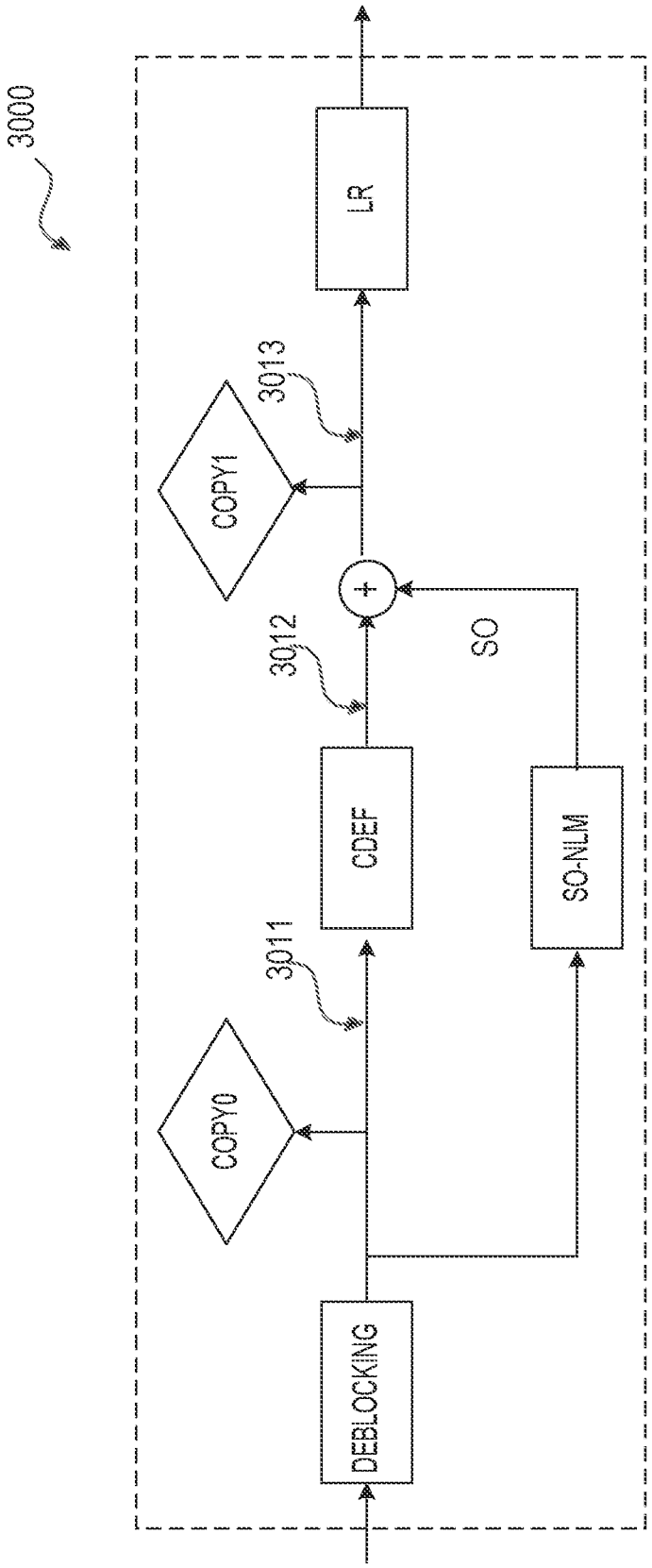

FIG. 30

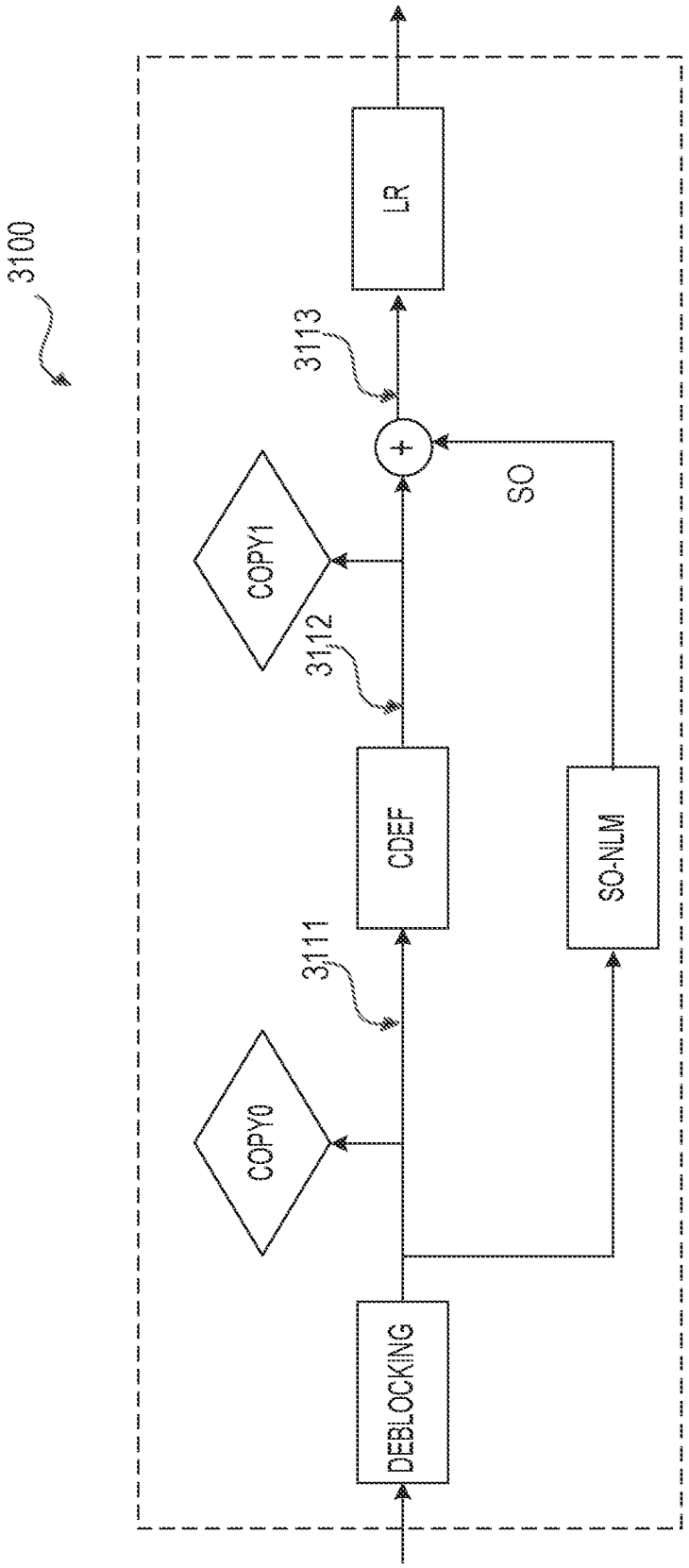

FIG. 31

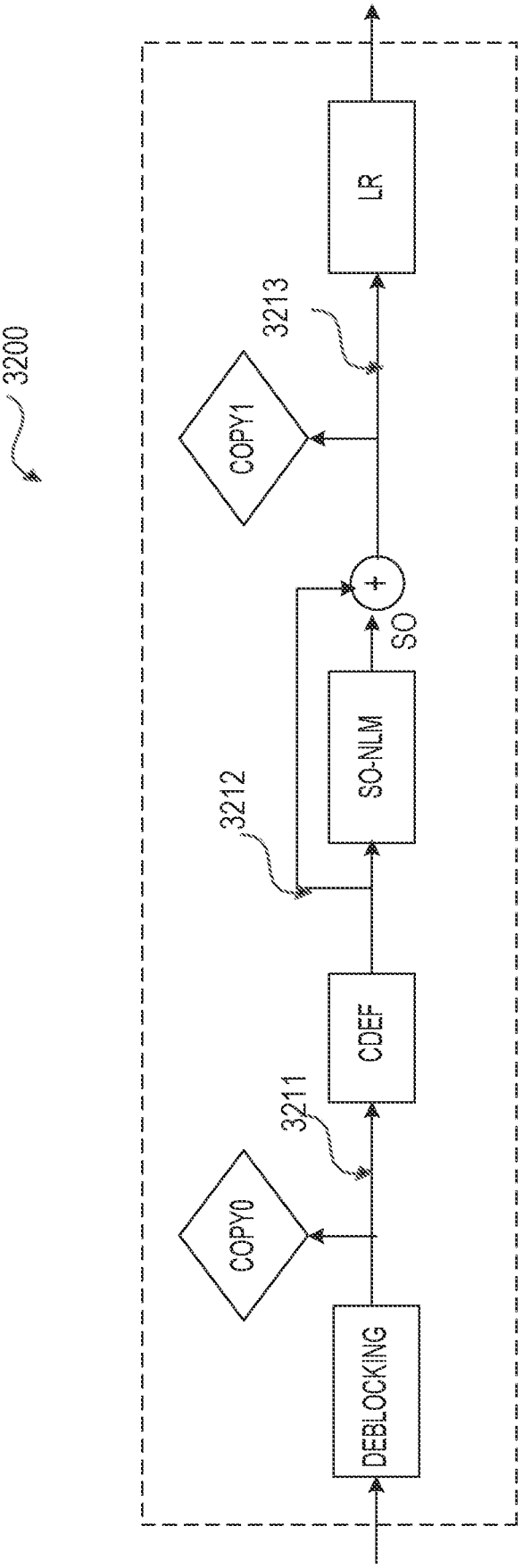

FIG. 32

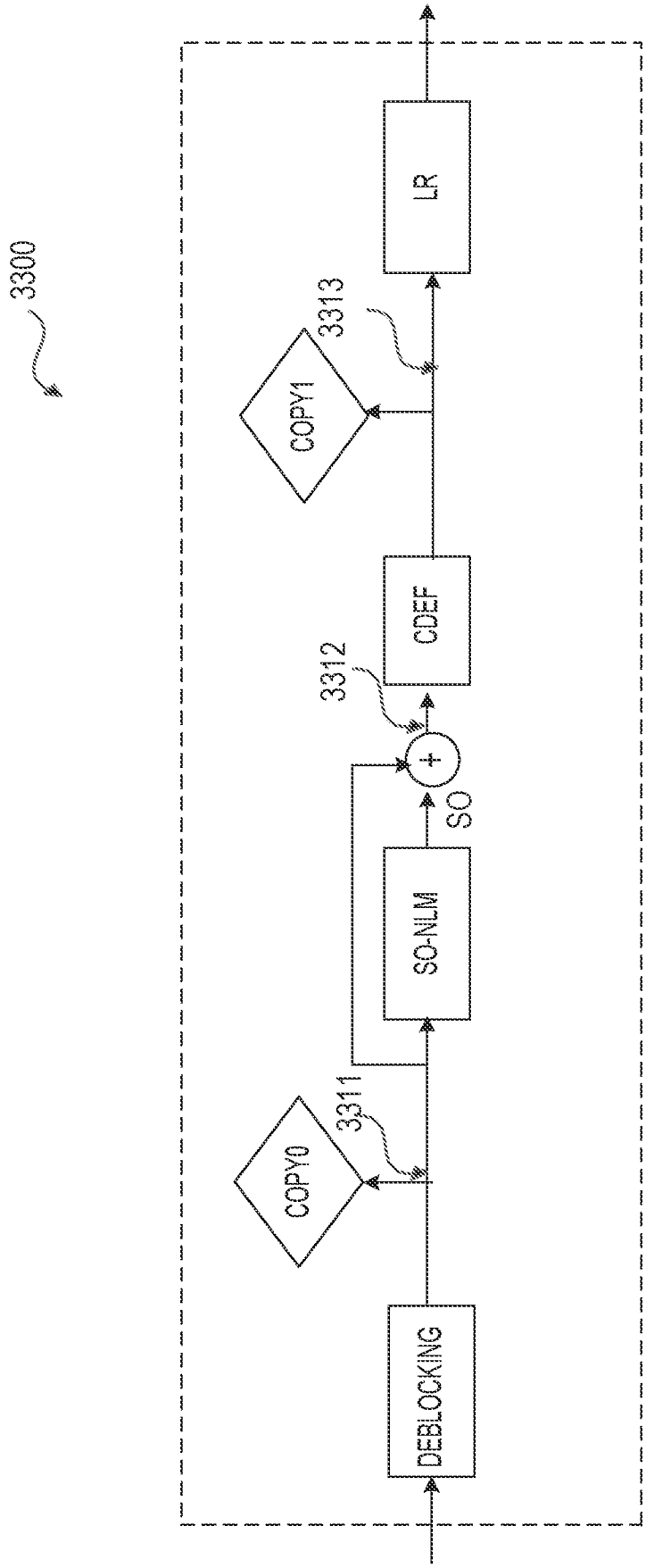

FIG. 33

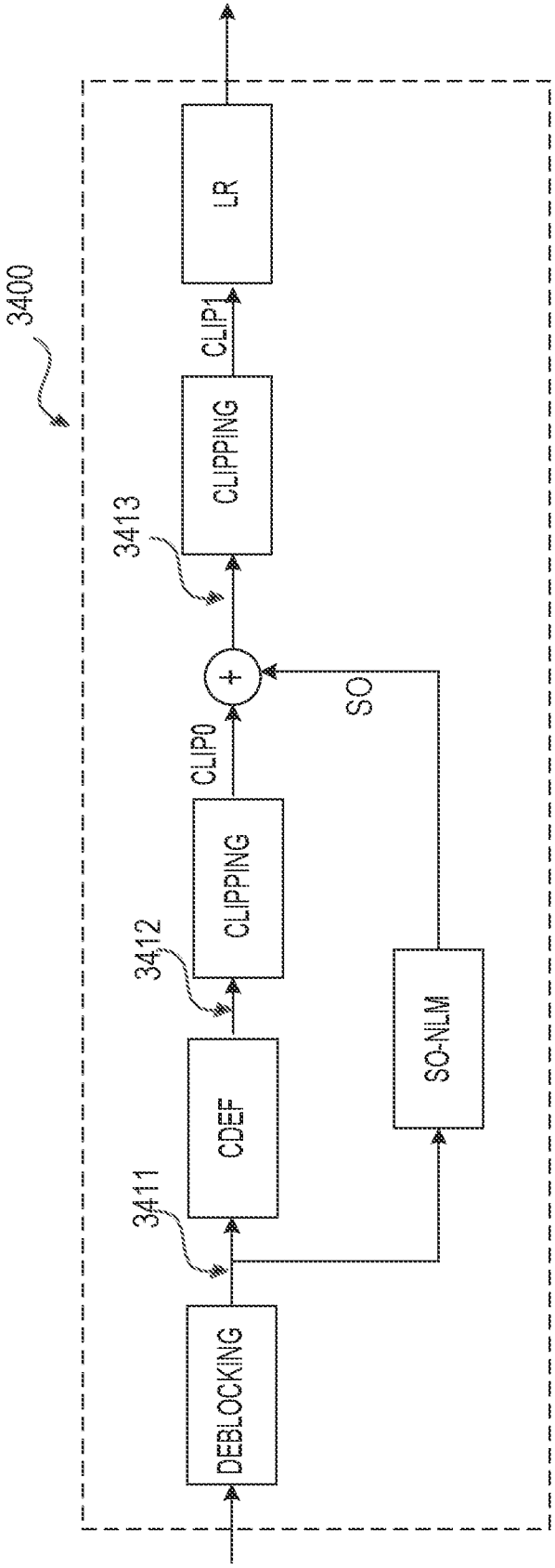
3400
DEBLOCKING
3411
CDEF
3412
CLIPPING
CLIP0
+
SO
SO-NLM
3413
CLIPPING
CLIP1
LR

FIG. 34

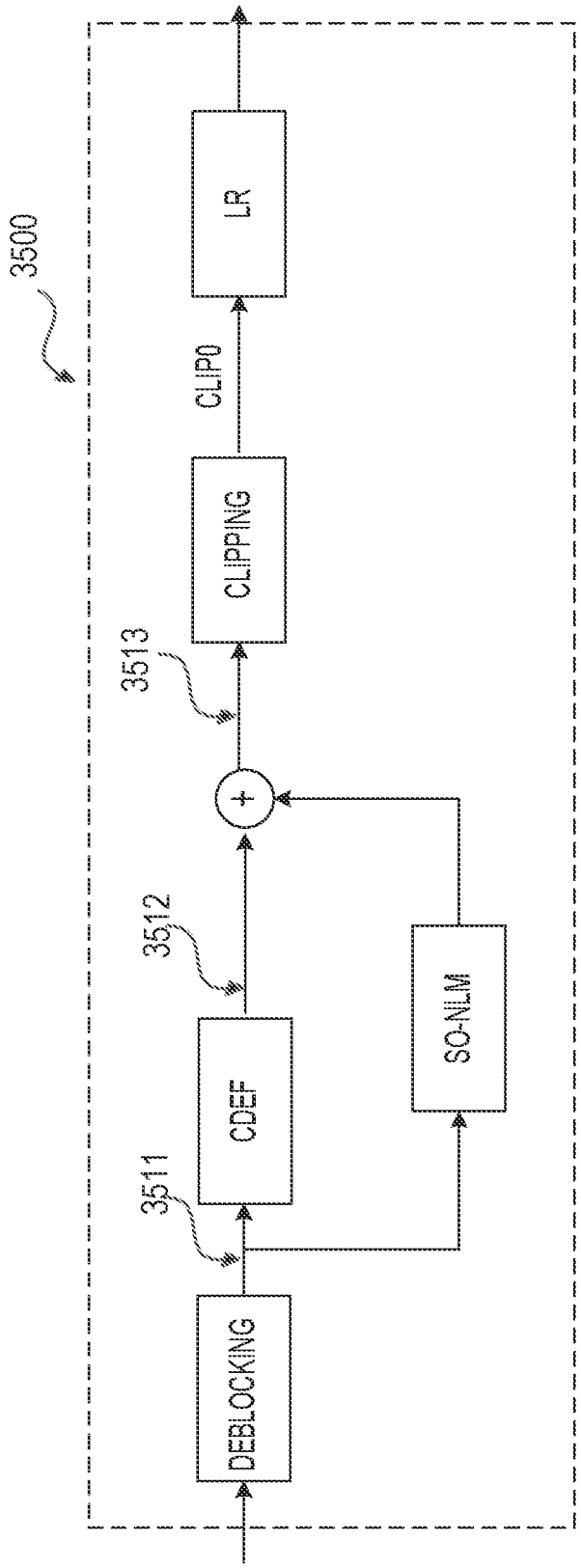
3500
DEBLOCKING
3511
CDEF
3512
SO-NLM
+
3513
CLIPPING
CLIP0
LR

FIG. 35

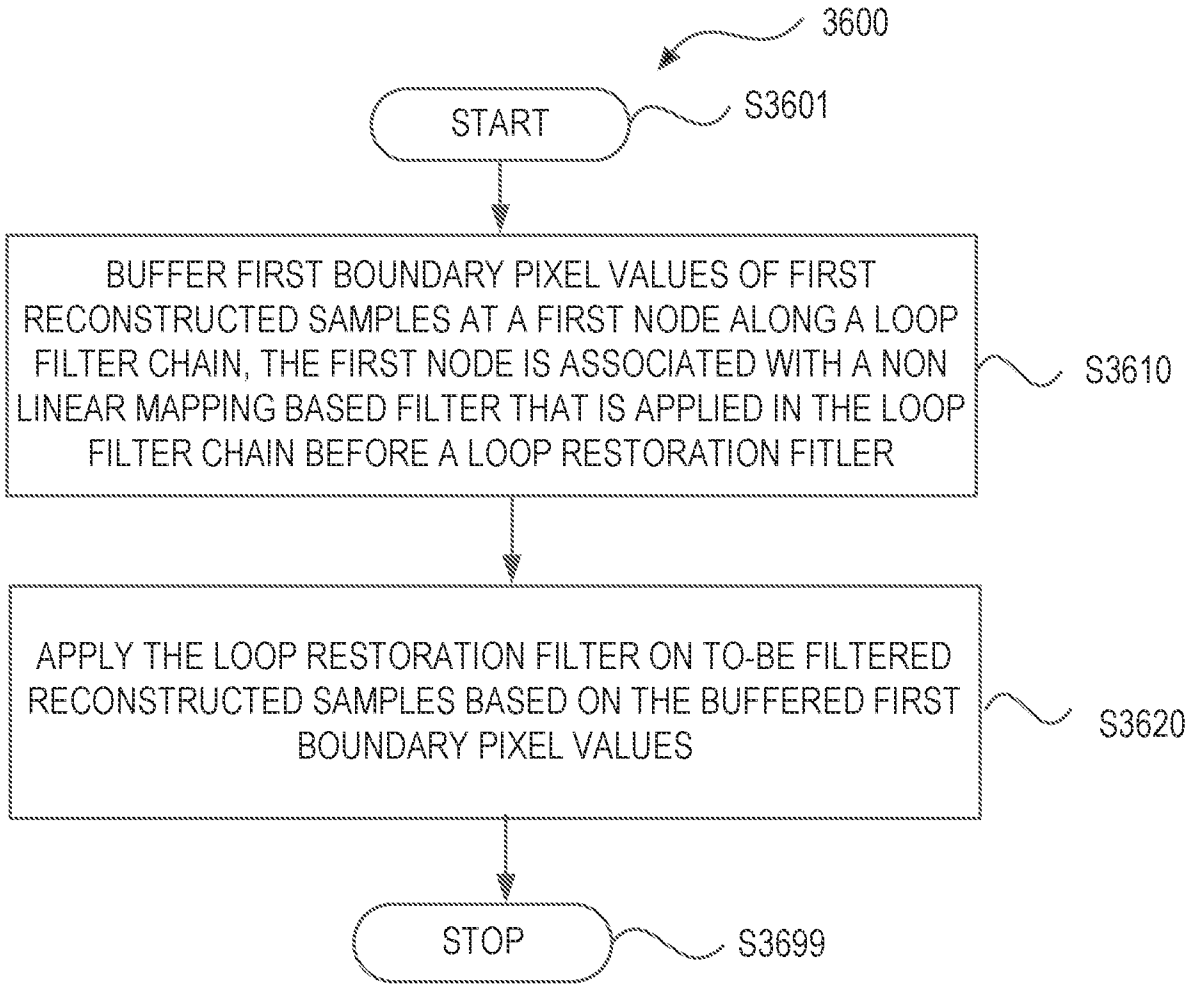
3600
START
S3601
BUFFER FIRST BOUNDARY PIXEL VALUES OF FIRST RECONSTRUCTED SAMPLES AT A FIRST NODE ALONG A LOOP FILTER CHAIN, THE FIRST NODE IS ASSOCIATED WITH A NON LINEAR MAPPING BASED FILTER THAT IS APPLIED IN THE LOOP FILTER CHAIN BEFORE A LOOP RESTORATION FITLER
S3610
APPLY THE LOOP RESTORATION FILTER ON TO-BE FILTERED RECONSTRUCTED SAMPLES BASED ON THE BUFFERED FIRST BOUNDARY PIXEL VALUES
S3620
STOP
S3699

FIG. 36

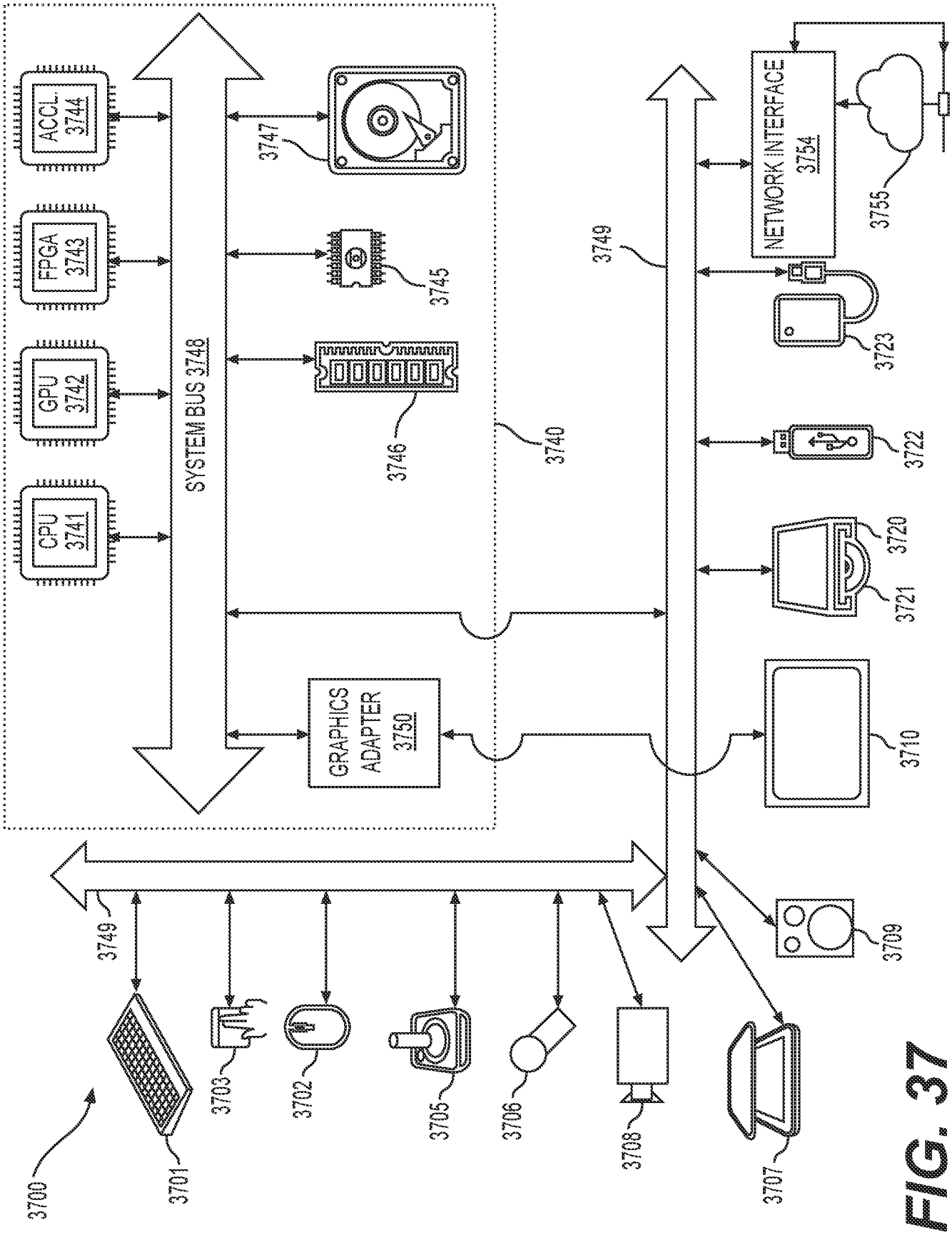
ACCL. 3744
FPGA 3743
GPU 3742
CPU 3741
SYSTEM BUS 3748
3747
3745
3746
3740
GRAPHICS ADAPTER 3750
3749
NETWORK INTERFACE 3754
3755
3723
3722
3720
3721
3710
3709
3700
3701
3703
3702
3705
3706
3708
3707

FIG. 37

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 21/52051

A. CLASSIFICATION OF SUBJECT MATTER

IPC - H04N 19/176 (2021.01)

CPC - H04N 19/176, H04N 19/61, H04N 19/119, H04N 19/172, H04N 19/82

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X ---- Y	US 2020/0404339 A1 (Huawei Technologies Co., Ltd.) 24 December 2020 (24.12.2020) entire document (especially para [0106]-[0110], [0154]-[0155], [0158]-[0159], [0194]).	1-8, 11-18 ---------- 9-10, 19-20
Y	US 2020/0344468 A1 (MEDIATEK INC) 29 October 2020 (29.10.2020) entire document (especially para [0040]-[0045])	9-10, 19-20

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

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 November 2021 (30.11.2021)	DEC 23 2021
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Kari Rodriquez Telephone No. PCT Helpdesk: 571-272-4300