



(51) International Patent Classification:

H04N 19/12 (2014.01) H04N 19/102 (2014.01)  
H04N 19/122 (2014.01) H04N 19/88 (2014.01)  
H04N 19/176 (2014.01) H04N 19/166 (2014.01)  
H04N 19/132 (2014.01) H04N 19/00 (2014.01)  
H04N 19/44 (2014.01)

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

(72) Inventors: **ZHAO, Xin**; c/o TENCENT AMERICA LLC, 2747 Park Boulevard, Palo Alto, California 94306 (US). **LI, Xiang**; 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).

(21) International Application Number:

PCT/US2020/021659

(22) International Filing Date:

09 March 2020 (09.03.2020)

(74) Agent: **MA, Johnny** et al.; Oblon, McClelland, Maier & Neustadt, L.L.P., 1940 Duke Street, Alexandria, Virginia 22314 (US).

(25) Filing Language:

English

(26) Publication Language:

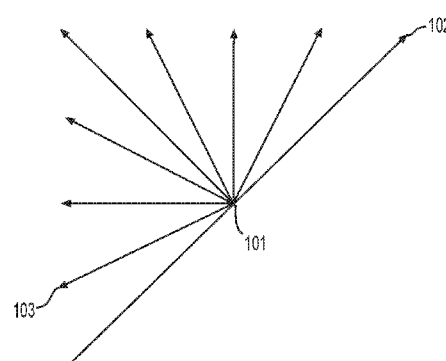
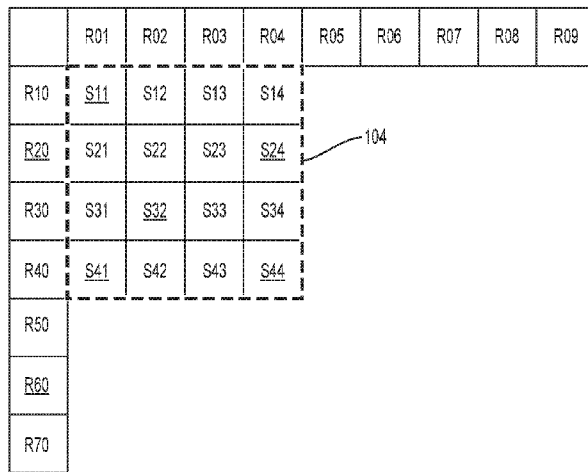
English

(30) Priority Data:

62/816,124 09 March 2019 (09.03.2019) US  
16/812,000 06 March 2020 (06.03.2020) US

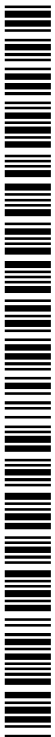
(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,

(54) Title: METHOD AND APPARATUS FOR VIDEO CODING



**FIG. 1A**

(57) Abstract: Aspects of the disclosure provide methods and apparatuses for video encoding/decoding. In some examples, an apparatus for video decoding includes processing circuitry. The processing circuitry can decode coding information of a coefficient block from a coded video bitstream. The coding information can indicate a size of the coefficient block. The processing circuitry can determine, based on the size of the coefficient block, an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed on transform coefficients of the coefficient block to obtain residual data of a residual block. When the size of the coefficient block satisfies a condition, the inverse vertical transform is performed after the inverse horizontal transform is performed on the transform coefficients of the coefficient block. The processing circuitry can reconstruct a sample in the residual block based on the residual data.



HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

**(84) Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, 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))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

## METHOD AND APPARATUS FOR VIDEO CODING

INCORPORATION BY REFERENCE

[0001] This present application claims the benefit of priority to U.S. Patent Application No. 16/812,000, "Method and Apparatus for Video Coding" filed on March 6, 2020, which claims the benefit of priority to U.S. Provisional Application No. 62/816,124, "Adaptive Transform Coefficient Zero-Out" filed on March 9, 2019. The entire disclosures of the prior applications are 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 significant 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 or storage space requirements, in some cases by two orders of magnitude or more. Both lossless 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 only using reference data 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]** Video coding and decoding can be performed using inter-picture prediction with motion compensation. 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 video encoding/decoding. In some examples, an apparatus for video decoding includes processing circuitry. The processing circuitry can decode coding information of a coefficient block from a coded video bitstream. The coding information can indicate a size of the coefficient block. The processing circuitry can determine, based on the size of the coefficient block, an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed on transform coefficients of the coefficient block to obtain residual data of a residual block. When the size of the coefficient block satisfies a condition, the inverse

vertical transform is performed after the inverse horizontal transform is performed on the transform coefficients of the coefficient block. The processing circuitry can reconstruct a sample in the residual block based on the residual data. In an example, the condition is that the size of the coefficient block is 32x64. In an example, the condition is that a height N of the coefficient block of MxN is larger than a width M of the coefficient block. In an example, the size of the coefficient block is MxN, where M and N are positive integers. First residual data in a mxn region in the residual block are to be calculated by the inverse primary transform and second residual data outside the mxn region in the residual block are not to be calculated by the inverse primary transform where m is less than or equal to M, and n is less than or equal to N. The condition is that a ratio m/M is larger than or equal to a ratio n/N.

**[0023]** Aspects of the disclosure provide methods and apparatuses for video encoding/decoding. In some examples, an apparatus for video decoding includes processing circuitry. The processing circuitry can decode coding information of a coefficient block from a coded video bitstream. The coding information can indicate a size of the coefficient block. The processing circuitry can determine, based on the size of the coefficient block, whether to reduce a number of calculations in one of inverse horizontal and inverse vertical transforms of an inverse primary transform. The inverse vertical transform can transform transform coefficients of the coefficient block to intermediate data of an intermediate block, and the inverse horizontal transform can transform the intermediate data to residual data in a residual block. The processing circuitry can perform the inverse primary transform. When the number of calculations in the inverse vertical transform is determined to be reduced, top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero. When the number of calculations in the inverse horizontal transform is determined to be reduced, left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero. The processing circuitry can reconstruct a sample in the residual block based on the residual data.

**[0024]** In an embodiment, the size of the coefficient block is 32x64. The one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform. The processing circuitry can determine that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is 32x64. The processing circuitry can perform the inverse primary transform where the top 16 rows of the intermediate data in the intermediate block can be calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero.

[0025] In an embodiment, the size of the coefficient block is  $32 \times 64$ . The one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform. The processing circuitry can determine that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 64$ . The processing circuitry can perform the inverse primary transform where the left 16 columns of the residual data in the residual block can be calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero.

[0026] In an embodiment, the size of the coefficient block is  $32 \times 32$ . The one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform. The processing circuitry can determine that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 32$ . The processing circuitry can perform the inverse primary transform where the left 16 columns of the residual data in the residual block can be calculated by the inverse horizontal transform, the remaining residual data in the residual block are zero, and the intermediate data in the intermediate block can be calculated by the inverse vertical transform.

[0027] In an embodiment, the size of the coefficient block is  $32 \times 32$ . The one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform. The processing circuitry can determine that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is  $32 \times 32$ . The processing circuitry can perform the inverse primary transform where the top 16 rows of the intermediate data in the intermediate block can be calculated by the inverse vertical transform, the remaining intermediate data in the intermediate block are zero, and the residual data in the residual block can be calculated by the inverse horizontal transform.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0028] 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:

[0029] FIG. 1A is a schematic illustration of an exemplary subset of intra prediction modes.

[0030] FIG. 1B is an illustration of exemplary intra prediction directions.

[0031] FIG. 2 is a schematic illustration of a current block and its surrounding spatial merge candidates in one example.

[0032] FIG. 3 is a schematic illustration of a simplified block diagram of a communication system (300) in accordance with an embodiment.

[0033] FIG. 4 is a schematic illustration of a simplified block diagram of a communication system (400) in accordance with an embodiment.

[0034] FIG. 5 is a schematic illustration of a simplified block diagram of a decoder in accordance with an embodiment.

[0035] FIG. 6 is a schematic illustration of a simplified block diagram of an encoder in accordance with an embodiment.

[0036] FIG. 7 shows a block diagram of an encoder in accordance with another embodiment.

[0037] FIG. 8 shows a block diagram of a decoder in accordance with another embodiment.

[0038] FIG. 9 shows an example of transform unit syntax in accordance with an embodiment.

[0039] FIGs. 10A-10C show an example of residual coding syntax in accordance with an embodiment.

[0040] FIGs. 11A-11B show examples of primary transforms in accordance with an embodiment.

[0041] FIGs. 12A-12E show an exemplary transformation process in accordance with an embodiment.

[0042] FIG. 13 shows an example of residual coding syntax in accordance with another embodiment.

[0043] FIG. 14 shows a flow chart outlining a process (1400) in accordance with an embodiment.

[0044] FIG. 15 shows a flow chart outlining a process (1500) in accordance with an embodiment.

[0045] FIG. 16 is a schematic illustration of a computer system in accordance with an embodiment.

#### DETAILED DESCRIPTION OF EMBODIMENTS

[0046] 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.

**[0047]** 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.

**[0048]** 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.

**[0049]** 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.

**[0050]** 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.

**[0051]** 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.

**[0052]** 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.

**[0053]** 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).

**[0054]** 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.

**[0055]** 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).

**[0056]** 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.

**[0057]** 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.

**[0058]** 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).

**[0059]** 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).

**[0060]** 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.

**[0061]** 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.

**[0062]** 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.

**[0063]** 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.

**[0064]** 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.

**[0065]** 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.

**[0066]** 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.

**[0067]** 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).

**[0068]** 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 colorspace (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.

**[0069]** 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.

**[0070]** 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.

**[0071]** 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).

**[0072]** 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.

**[0073]** 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.

**[0074]** 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).

**[0075]** 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).

**[0076]** 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.

**[0077]** 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.

**[0078]** 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).

**[0079]** 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:

**[0080]** 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.

**[0081]** 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.

**[0082]** 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.

**[0083]** 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.

**[0084]** 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.

**[0085]** 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.

**[0086]** 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.

[0087] 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.

[0088] Further, a merge mode technique can be used in the inter-picture prediction to improve coding efficiency.

[0089] 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.

[0090] 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.

[0091] 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.

**[0092]** 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.

**[0093]** 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.

**[0094]** 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.

**[0095]** 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.

**[0096]** 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.

**[0097]** 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.

**[0098]** 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.

**[0099]** 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.

**[0100]** 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).

**[0101]** The inter decoder (880) is configured to receive the inter prediction information, and generate inter prediction results based on the inter prediction information.

**[0102]** The intra decoder (872) is configured to receive the intra prediction information, and generate prediction results based on the intra prediction information.

**[0103]** 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).

**[0104]** 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.

**[0105]** 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.

**[0106]** In some embodiments, such as in HEVC, a primary transform may include 4-point, 8-point, 16-point and 32-point discrete cosine transform (DCT) type 2 (DCT-2), and the transform core matrices may be represented using 8-bit integers (i.e., 8-bit transform core). The transform core matrices of a smaller DCT-2 are part of transform core matrices of a larger DCT-2, as shown in APPENDIX I.

**[0107]** The DCT-2 core matrices show symmetry/anti-symmetry characteristics. Therefore, a “partial butterfly” implementation may be supported to reduce a number of operation counts (e.g., multiplications, additions, subtractions, shifts, and/or the like), and identical results of matrix multiplication can be obtained using the partial butterfly.

**[0108]** In some embodiments, such as in VVC, besides the 4-point, 8-point, 16-point, and 32-point DCT-2 transforms described above, additional 2-point and 64-point DCT-2 may also be included. An example of a 64-point DCT-2 core, such as used in VVC, is shown in APPENDIX II as a 64x64 matrix.

**[0109]** In addition to DCT-2 and  $4 \times 4$  DST-7, such as used in HEVC, an Adaptive Multiple Transform (AMT) (also known as Enhanced Multiple Transform (EMT) or Multiple Transform Selection (MTS)) scheme, can be used, such as in VVC, for residual coding for both inter and intra coded blocks. The AMT scheme may use multiple selected transforms from the DCT/DST families other than the current transforms in HEVC. The newly introduced transform matrices are DST-7, DCT-8. Table 1 shows examples of the basis functions of the selected DST/DCT for an N-point input.

Transform Type	Basis function $T_i(j), i, j=0, 1, \dots, N-1$
DCT-2	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j + 1)}{2N}\right)$ <p>where <math>\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} &amp; i = 0 \\ 1 &amp; i \neq 0 \end{cases}</math></p>
DCT-8	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \cos\left(\frac{\pi \cdot (2i + 1) \cdot (2j + 1)}{4N + 2}\right)$
DST-7	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (j + 1)}{2N + 1}\right)$

Table 1

[0110] The primary transform matrices, such as used in VVC, may be used with a 8-bit representation. In an embodiment, the AMT applies transform matrices to the CUs with both a width and a height smaller than or equal to 32. Whether AMT is applied may be controlled by a flag (e.g., an mts\_flag). When the mts\_flag is equal to 0, in some examples, only DCT-2 is applied for coding residue data. When the mts\_flag is equal to 1, an index (e.g., an mts\_idx) may be further signalled using 2 bins to identify the horizontal and vertical transform to be used according to Table 2 for example, where a type value of 1 means DST-7 is used, and type value of 2 means DCT-8 is used. In Table 2, the specification of trTypeHor and trTypeVer depend on mts\_idx[ x ][ y ][ cIdx ].

mts_idx[ xTbY ][ yTbY ][ cIdx ]	trTypeHor	trTypeVer
-1	0	0
0	1	1
1	2	1
2	1	2
3	2	2

Table 2

[0111] In some embodiments, an implicit MTS can be applied when the above signaling based MTS (i.e., explicit MTS) is not used. With the implicit MTS, the transform selection is made according to the block width and height instead of the signaling. For example, with an implicit MTS, DST-7 is selected for a shorter side (i.e., a minimum one of M and N) of the block of MxN and DCT-2 is selected for a longer side (i.e., a maximum one of M and N) of the block.

[0112] Exemplary transform cores, each of which is a matrix composed by the basis vectors, of DST-7 and DCT-8 are illustrated in APPENDIX III.

**[0113]** In some examples, such as in VVC, when both the height and width of the coding block is smaller than or equal to 64, the TB size is the same as the coding block size. When either the height or width of the coding block is larger than 64, when doing a transform (such as an inverse transform, an inverse primary transform, or the like) or intra prediction, the coding block is further split into multiple sub-blocks, where the width and height of each sub-block is smaller than or equal to 64. One transform can be performed on each sub-block.

**[0114]** Related syntax and semantics of MTS in some examples in VVC can be described below (highlighted using a label 910 and a label 1010) in FIGs. 9 and 10A-10C. An example of transform unit syntax is shown in FIG. 9. An example of a residual coding syntax is shown in FIGs. 10A-10C.

**[0115]** An example of the transform unit semantics is as follows. `cu_mts_flag[ x0 ][ y0 ]` equal to 1 specifies that multiple transform selection is applied to the residual samples of the associated luma transform block. `cu_mts_flag[ x0 ][ y0 ]` equal to 0 specifies that multiple transform selection is not applied to the residual samples of the associated luma transform block. The array indices `x0`, `y0` specify the location ( `x0`, `y0` ) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. When `cu_mts_flag[ x0 ][ y0 ]` is not present, it is inferred to be equal to 0.

**[0116]** An example of the residual coding semantics is as follows. `mts_idx[ x0 ][ y0 ]` specifies which transform kernels are applied to the luma residual samples along the horizontal and vertical direction of the current transform block. The array indices `x0`, `y0` specify the location ( `x0`, `y0` ) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. When `mts_idx[ x0 ][ y0 ]` is not present, it is inferred to be equal to -1.

**[0117]** FIG. 11A shows an exemplary forward transform (also referred to as a forward primary transform) performed by an encoder. The forward transform can include a forward horizontal transform and a forward vertical transform. The forward horizontal transform is applied first to a residual block (1110) having residual data to obtain an intermediate block. Subsequently, the forward vertical transform is applied to the intermediate block to obtain a coefficient block (1112) having transform coefficients.

**[0118]** FIG. 11B shows an exemplary backward transform (also referred to as an inverse primary transform or an inverse transform) performed by a decoder. Generally speaking, the inverse transform matches the forward transform. The inverse primary transform can include an inverse primary horizontal transform (also referred to as an inverse horizontal transform) and an inverse primary vertical transform (also referred to as an inverse

vertical transform). To match the forward transform, an order of applying the inverse horizontal and vertical transforms is switched in the inverse transform. Accordingly, the inverse vertical transform is applied first to a coefficient block (1122) to obtain an intermediate block. Subsequently, the inverse horizontal transform is applied to the intermediate block to obtain a residual block (1120).

**[0119]** A primary transform can refer to a forward primary transform or an inverse primary transform. A horizontal transform can refer to an inverse horizontal transform or a forward horizontal transform. Similarly, a vertical transform can refer to an inverse vertical transform or a forward vertical transform.

**[0120]** In an example, such as in VVC, at the decoder, the inverse vertical primary transform (also referred to as inverse vertical transform) is performed first, then the inverse horizontal primary transform (also referred to as inverse horizontal transform) is performed second after applying the inverse vertical transform, as shown in FIG. 11B and in texts highlighted using labels 1210-1211 in FIGs. 12A-12E. An example of a transformation process for scaled transform coefficients is shown in FIGs. 12A-12E.

**[0121]** A primary transform, such as a forward primary transform or an inverse primary transform can utilize a zero-out method or zero-out scheme as described below. In some examples, such as in VVC, for a 64-point (or 64-length) DCT-2, only the first 32 coefficients are calculated and the remaining coefficients are set as 0. Therefore, for an  $M \times N$  block which is coded using a DCT-2 transform, the top-left  $\min(M, 32) \times \min(N, 32)$  low frequency coefficients are calculated. The remaining coefficients are set as 0 and not signaled. In an example, the remaining coefficients are not calculated. The entropy coding of the coefficient block can be performed by setting the coefficient block size as  $\min(M, 32) \times \min(N, 32)$ , such that the coefficient coding of an  $M \times N$  block is regarded as a  $\min(M, 32) \times \min(N, 32)$  coefficient block.

**[0122]** In some examples in which MTS is used, for 32-point DST-7 or DCT-8, only the first 16 coefficients are calculated and the remaining coefficients are set as 0. Therefore, for an  $M \times N$  block which is coded using a DST-7 or DCT-8 transform, the top-left  $\min(M, 16) \times \min(N, 16)$  low frequency coefficients are kept. The remaining coefficients can be set as 0 and not signaled. However, different from the coefficient coding scheme which is used when 64-point zero-out DCT-2 is applied, for 32-point MTS, the coefficient coding is still performed on the whole  $M \times N$  block even when  $M$  or  $N$  is larger than 16. However, when a coefficient group (CG) is outside the top-left  $16 \times 16$  low-frequency region (i.e., the coefficient group is in a zero-out region) a flag indicating whether the coefficient group has a nonzero

coefficient (e.g., a coded\_sub\_block\_flag) is not signaled. The zero-out region refers to a region in the coefficient block where coefficients are zero, and thus the coefficients in the zero-out region are zero. An example of a residual coding syntax is described below, indicated by the highlighted texts using label 1310 in FIG. 13.

**[0123]** One major complexity aspect of a primary transform, such as an inverse primary transform or a forward primary transform, is the average number of multiplications per sample (MPS). For example, for an 8x8 primary transform, if DST-7 using a full matrix multiplication is applied for both the horizontal and vertical transform for the 8x8 block, the horizontal transform requires 8 multiplications to calculate each coefficient and the vertical transform requires 8 multiplications to calculate each coefficient, so together 16 multiplications per sample is used.

**[0124]** In some examples, a worst-case MPS is 32x32 DCT2 and 32x64 DCT-2, as described below. A MPS can also be referred to as multiplications per coefficient.

**[0125]** For a MxN TU, if non-zero transform coefficients are kept for the top-left mxn (i.e., m by n) region of the TU, the number of multiplications per coefficient for the inverse transform implemented in the direct matrix multiply structure can be computed by

$Muls\_per\_coeff_{directMatrixMultiply} = \frac{m}{M}n + m$ . When n is less than N or m is less than M, the zero-out method is applied and the number of multiplications per coefficient can be reduced.

**[0126]** Likewise, the number of multiplications per coefficient for the inverse transform implemented in the structure of half butterfly and half direct matrix multiply can be computed by

$$Muls\_per\_coeff_{halfButterFly} = \frac{m}{M} \left( \frac{n}{2} + \log_2 \frac{n}{2} \right) + \left( \frac{m}{2} + \log_2 \frac{m}{2} \right).$$

**[0127]** In some examples, such as in VVC, DST7/DCT8 transforms up to 32-point (e.g., 4-point DST-7, 8-point DST-7, 16-point DST-7, 32-point DST-7, 4-point DCT-8, 8-point DCT-8, 16-point DCT-8, and 32-point DCT-8), and DCT2 transforms up to 64-point are defined for primary transforms (e.g., 4-point DCT-2, 8-point DCT-2, 16-point DCT-2, 32-point DCT-2, and 64-point DCT-2). For the 64-point primary transform (e.g., 64-point DCT-2), a transform coefficient zero-out scheme (also referred to zero-out method, zero-out scheme) can be applied. With the zero-out scheme, for a 64-point primary transform, in some examples, only the first 32 coefficient are kept, and the remaining coefficients are set as 0. For example, for a 64x64 block, the top-left 32x32 coefficients are kept, and the remaining coefficients are set as 0. For a 64xN block, where N is less than or equal to 32, the left 32xN coefficients may be kept, and the remaining coefficients may be set as 0. For a Mx64 block,

where M is less than or equal to 32, the top Mx32 coefficients may be kept and the remaining coefficients are set as 0.

[0128] Table 3 shows numbers of multiplications per coefficient where M is the block width, N is the block height, m is the number of coefficients kept in the horizontal direction (i.e., in each row), and n is the number of coefficients kept in the vertical direction (i.e., in each column). As illustrated in Table 3, when the top-left 16x16 coefficients are kept for DST-7/DCT-8 (i.e., m=n=16), the worst case (e.g., the number of multiplications per coefficient is a maximum number) includes 32x64 and 32x32 DCT-2. For example, the number of multiplications per coefficient for 32x64 and 32x32 DCT-2 is 40 (using half butterfly), and corresponds to the worse case.

[0129] In some examples, 32x32 DCT2 without zero-out and using half butterfly, 32x32 DST7/DCT8 with zero-out (m=n=16), 32x16 DST7/DCT8 with zero-out (m=n=16), 16x32 DST7/DCT8 with zero-out (m=n=16) can be employed in video coding.

	M	N	m	n	number of multiplications per coefficient (direct matrix multiply)	number of multiplications per coefficient (half butterfly)
DCT2	TU size M*N, and keep the top-left m*n non-zero coefficients					
	64	64	32	32	48	30
	64	32	32	32	48	30
	32	64	32	32	64	40
	32	32	32	32	64	40
DST7/DCT8	TU size M*N, and keep the top-left m*n non-zero coefficients					
	32	32	32	32	64	
	32	16	32	16	48	
	16	32	16	32	48	
	32	32	16	16	24	
	32	16	16	16	24	
	16	32	16	16	32	

Table 3

[0130] In an embodiment, a mode-dependent non-separable secondary transform (NSST) can be used between a forward core transform and a quantization at an encoder side and between a de-quantization and an inverse core transform at a decoder side. For example, to keep a low complexity, a NSST is applied to low frequency coefficients after a primary

transform (or a core transform). When both a width (W) and a height (H) of a transform coefficient block are larger than or equal to 8, an 8×8 NSST is applied to a top-left 8×8 region of the transform coefficients block. Otherwise, when either the width W or the height H of the transform coefficient block is 4, a 4×4 NSST is applied, and the 4×4 NSST is performed on a top-left min(8,W)×min(8,H) region of the transform coefficient block. The above transform selection method is applied for both luma and chroma components.

**[0131]** A matrix multiplication implementation of a NSST is described as follows using a 4×4 input block as an example. The 4×4 input block X is written in Eq. (1) as

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix} \quad (1)$$

**[0132]** The input block X can be represented as a vector  $\vec{X}$  in Eq. (2) where

$$\vec{X} = [X_{00} X_{01} X_{02} X_{03} X_{10} X_{11} X_{12} X_{13} X_{20} X_{21} X_{22} X_{23} X_{30} X_{31} X_{32} X_{33}]^T \quad (2)$$

The non-separable transform is calculated as  $\vec{F} = T \cdot \vec{X}$ , where  $\vec{F}$  indicates a transform coefficient vector, and T is a 16×16 transform matrix. The 16×1 transform coefficient vector  $\vec{F}$  is subsequently reorganized as a 4×4 block using a scanning order (for example, a horizontal scanning order, a vertical scanning order or a diagonal scanning order) for the input block X. Coefficients with smaller indices can be placed with smaller scanning indices in the 4×4 coefficient block. In some embodiments, a Hypercube-Givens Transform (HyGT) with a butterfly implementation can be used instead of the matrix multiplication described above to reduce the complexity of the NSST.

**[0133]** In an example, 35×3 non-separable secondary transforms are available for both 4×4 and 8×8 block sizes, where 35 is a number of transform sets associated with the intra prediction modes, and 3 is a number of NSST candidates for each intra prediction mode. Table 4 shows an exemplary mapping from an intra prediction mode to a respective transform set. A transform set applied to luma/chroma transform coefficients is specified by a corresponding luma/chroma intra prediction mode, according to Table 4 that shows mapping from an intra prediction mode to a transform set index. For an intra prediction mode larger than 34, which corresponds to a diagonal prediction direction, a transform coefficient block is transposed before/after the NSST at the encoder/decoder, respectively.

**[0134]** For each transform set, a selected NSST candidate can be further specified by an explicitly signaled CU level NSST index. The CU level NSST index is signaled in a bitstream for each intra coded CU after transform coefficients and a truncated unary binarization is used for the CU level NSST index. For example, a truncated value is 2 for the

planar or the DC mode, and 3 for an angular intra prediction mode. In an example, the CU level NSST index is signaled only when there is more than one non-zero coefficient in the CU. The default value is zero and not signaled, indicating that a NSST is not applied to the CU. Each of values 1-3 indicates which NSST candidate is to be applied from the transform set.

Intra mode	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
set	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Intra mode	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
set	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Intra mode	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
set	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18
Intra mode	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67(LM)
set	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	NULL

Table 4

[0135] In some embodiments, a NSST is not applied for a block coded with a transform skip mode. When the CU level NSST index is signaled for a CU and not equal to zero, a NSST is not used for a block that is coded with the transform skip mode in the CU. When the CU with blocks of all components are coded in a transform skip mode or a number of non-zero coefficients of non-transform-skip mode CBs is less than 2, the CU level NSST index is not signaled for the CU.

[0136] Referring to Table 3, when DCT-2 is used in a primary transform, the worst case MPS used for the primary transform can correspond to a TB size of 32x32 or 32x64. In an example, the worst case MPS uses 1/3 more multiplications with respect to the second worst case (e.g., a TB size of 64x64 or 64x32 when using DCT2). Aspects of the disclosure include methods that reduce a MPS of a TB size of 32x32 or 32x64 when using DCT-2 to perform a primary transform, such as an inverse primary transform. Embodiments herein can also be applied to reduce a MPS of any suitable TB sizes and using any suitable transform that is not limited to DCT-2. The suitable transform can use a transform matrix in DCT and/or DST families including DCT-2, DST-7, DST-4, DCT-8, DCT-4, and the like.

[0137] Embodiments described herein may be used separately or combined in any order. Further, the embodiments may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits) in an encoder, a decoder, or the like. In one example, the one or more processors can execute a program that is stored in a non-transitory computer-readable medium.

[0138] In the disclosure, embodiments that are applicable for DST-7 of an MTS candidate can be applicable to DST-4 and vice versa. Similarly, embodiments that are applicable for DCT-8 of an MTS candidate are also applicable to DCT-4 and vice versa.

[0139] In the disclosure, embodiments that are applicable for NSST can be applicable for Reduced Secondary Transform (RST) that is an alternative design of NSST. A secondary transform can refer to NSST or RST.

[0140] As described above in FIG. 11B, a first order in which an inverse primary transform is applied on a TB is applying an inverse vertical transform first followed by an inverse horizontal transform. According to some embodiments, a second order can be used for applying the inverse primary transform in which the inverse horizontal transform is applied first, and the inverse vertical transform is applied after applying the inverse horizontal transform. Accordingly, the first order can be switched to the second order. In general, the inverse primary transform can use any suitable type of primary transforms, such as a transform matrix in DCT and/or DST families including DCT-2, DST-7, DST-4, DCT-8, DCT-4, and the like. In some embodiments, the inverse primary transform is implemented with a DCT-2 transform matrix. In an example, the inverse primary transform is implemented with a direct matrix multiplication structure. In another example, the inverse primary transform is implemented with a structure of half butterfly and half direct matrix multiplication.

[0141] In an embodiment, the second order is used (e.g., the first order is switched to the second order) for certain TB sizes. Accordingly, an order of applying the inverse horizontal and vertical transforms can depend on a TB size. The TB size can refer to an area of the TB, such as  $M \times N$ , a width of the TB, a height of the TB, or the like. In an example, when the TB size is  $32 \times 64$ , the second order is used (i.e., the inverse horizontal transform is applied before the inverse vertical transform). When the TB size is different from  $32 \times 64$ , the first order is used (i.e., the inverse vertical transform is applied before the inverse horizontal transform).

[0142] In an embodiment, when a height  $N$  of the TB of  $M \times N$  is greater than a width  $M$  of the TB, the second order can be used such that the inverse horizontal transform is applied before the inverse vertical transform. Otherwise, when the height  $N$  of the TB is smaller than or equal to the width  $M$  of the TB, the first order can be used such that the inverse vertical transform is applied before the inverse horizontal transform.

[0143] The inverse horizontal transform can be performed to transform first elements in a first block of  $M \times N$  into second elements in a second block of  $M \times N$ . The inverse

horizontal transform can be referred to as an one-dimensional transform, such as an  $M$ -length transform, where the inverse horizontal transform can be applied to the first block row-by-row where each row has  $M$  first elements. In an embodiment, the second elements in left  $m$  columns in the second block are calculated by the inverse horizontal transform while the second elements in right  $(M-m)$  columns in the second block are not calculated and are zero. When the inverse horizontal transform includes the zero-out method,  $m$  is a positive integer that is less than  $M$ , the inverse horizontal transform described above calculates first  $m$  second elements in each row of the second block while remaining  $(M-m)$  second elements in each row are not calculated and are zero. Thus, the first  $m$  second elements (in each row) are kept by the  $M$ -length transform using a zero-out method. Accordingly, a number of calculations (and an amount of data to be calculated and stored) can be reduced by including or using the zero-out method. When the inverse horizontal transform does not include the zero-out method,  $m$  can be equal to  $M$ .

**[0144]** The inverse vertical transform can be performed to transform third elements in a third block of  $M \times N$  into fourth elements in a fourth block of  $M \times N$ . The inverse vertical transform can be referred to as an one-dimensional transform, such as an  $N$ -length transform, where the inverse vertical transform can be applied to the third block column-by-column where each column has  $N$  third elements. In an embodiment, the fourth elements in top  $n$  rows in the fourth block are calculated by the inverse vertical transform while the fourth elements in bottom  $(N-n)$  rows in the fourth block are not calculated and are zero. When the inverse vertical transform includes the zero-out method,  $n$  is a positive integer that is less than  $N$ , the inverse vertical transform described above calculates first  $n$  fourth elements in each column of the fourth block while remaining  $(N-n)$  fourth elements in each column are not calculated and are zero. Thus, the first  $n$  fourth elements (in each column) are kept by the  $N$ -length transform using a zero-out method. Accordingly, a number of calculations (and an amount of data to be calculated and stored) can be reduced by including the zero-out method. When the inverse vertical transform does not include the zero-out method,  $n$  can be equal to  $N$ .

**[0145]** The inverse primary transform, e.g., DCT-2, can be applied on a coefficient block having the TB size of  $M \times N$  to obtain a residual block of  $M \times N$ . According to an embodiment of the disclosure, first residual data in a  $m \times n$  region in the residual block are to be determined by the inverse primary transform and second residual data outside the  $m \times n$  region in the residual block are not to be calculated by the inverse primary transform where  $m$  is less than or equal to  $M$ , and  $n$  is less than or equal to  $N$ . As described above, when the

inverse horizontal transform includes the zero-out method,  $m$  is less than  $M$ . When the inverse vertical transform includes the zero-out method,  $n$  is less than  $N$ . When a ratio  $m/M$  is larger than or equal to a ratio  $n/N$ , the inverse horizontal transform is applied before applying the inverse vertical transform. Otherwise, when the ratio  $m/M$  is smaller than the ratio  $n/N$ , the inverse vertical transform is applied before applying the inverse horizontal transform.

**[0146]** In general, the inverse primary transform is performed to transform the transform coefficients in the coefficient block into the residual data in the residual block. The TB can refer to the coefficient block and/or the residual block. The TB size, a size of the coefficient block, and a size of the residual block are  $M \times N$ . When the ratio  $m/M$  is larger than or equal to the ratio  $n/N$ , the inverse horizontal transform is applied before applying the inverse vertical transform. Accordingly, the first block is a coefficient block of  $M \times N$  and the first data are transform coefficients. The second block is an intermediate block of  $M \times N$  and the second data are intermediate data. The third block is identical to the second block, and the third data are identical to the second data. The fourth block is a residual block of  $M \times N$  and the fourth data are residual data.

**[0147]** According to some embodiments, an amount of data that is calculated or kept in a one-dimensional transform that is applied to a TB using the zero-out method can be determined based on the TB size. As described above, the one-dimensional transform can refer to an inverse horizontal transform or an inverse vertical transform. In an embodiment, the inverse primary transform is performed using the first order shown in FIG. 11B such that the inverse vertical transform is applied first followed by the inverse horizontal transform. When the TB size is  $32 \times 64$  with a width of 32 and a height of 64, a 64-length inverse vertical transform can be applied with the zero-out to transform a coefficient block of  $32 \times 64$  having transform coefficients into an intermediate block having intermediate data by calculating the top 16 rows of the intermediate block (i.e., first 16 pieces of intermediate data in each column). Remaining intermediate data in the intermediate block are not calculated and are zero (e.g., the remaining intermediate data can be set to zero). In some examples, when the TB size is different from  $32 \times 64$ , such as  $16 \times 64$ , the 64-length inverse vertical transform calculates the top 32 rows of the intermediate data in the intermediate block, and the remaining intermediate data in the intermediate block are not calculated and are zero.

**[0148]** When the TB size is  $32 \times 64$  and the first order is used, a 32-length inverse horizontal transform can be applied with the zero-out to transform an intermediate block of  $32 \times 64$  having intermediate data into a residual block having residual data by calculating the

left 16 columns of the residual block (i.e., first 16 pieces of residual data in each row). Remaining residual data in the residual block are not calculated and are zero (e.g., the remaining residual data can be set to zero).

**[0149]** In some examples, when the TB size is  $M \times 32$  where  $M$  is smaller than 32, the 32-length inverse vertical transform does not apply the zero-out scheme. In some examples, when the TB size is  $32 \times N$  where  $N$  is smaller than 32, the 32-length inverse horizontal transform does not apply the zero-out scheme.

**[0150]** According to some embodiments, an amount of data to be calculated for a row or a column using a one-dimensional transform can be determined based on whether the one-dimensional transform is an inverse horizontal transform or an inverse vertical transform. For example, a zero-out scheme can be used for only one of the one-dimensional transforms. In an embodiment, the inverse primary transform is performed using the first order such that the inverse vertical transform is applied first followed by the inverse horizontal transform. The TB size is  $32 \times 32$ . In an example, the 32-length vertical transform does not apply the zero-out method. Accordingly, the 32-length inverse vertical transform calculates or keeps all intermediate data in 32 rows of an intermediate block. The 32-length inverse horizontal transform calculates or keeps the left 16 columns of a residual block using the zero-out method. Remaining residual data in the residual block are not calculated and are zero (e.g., the remaining residual data can be set to zero). In an example, the 32-length inverse vertical transform calculates or keeps the top 16 rows of an intermediate block using the zero-out method. Remaining intermediate data in the intermediate block are not calculated and are zero (e.g., the remaining intermediate data can be set to zero). Meanwhile, the 32-length inverse horizontal transform does not apply the zero-out method. Accordingly, the 32-length inverse horizontal transform calculates or keeps all residual data in 32 columns of a residual block.

**[0151]** When a first inverse primary transform is applied to a TB without an inverse secondary transform, a first MPS is due to the first inverse primary transform. When a second inverse primary transform and an inverse secondary transform are applied to the TB, a second MPS (also referred to as a combined MPS) can include multiplications in both the second inverse primary transform and the inverse secondary transform. When the first inverse primary transform is identical to the second inverse primary transform, the second MPS can be larger than the first MPS because additional multiplications are used in the inverse secondary transform. Embodiments below can be implemented to constrain or reduce

the combined MPS to be within a MPS limit (e.g., the worse case MPS shown in Table 3) and thus to improve coding efficiency,

**[0152]** According to aspects of the disclosure, a number of non-zero transform coefficients that are kept using a zero-out scheme, and/or whether a zero-out method is used in an inverse primary transform, can depend on whether a secondary transform is applied. Thus, a combined MPS for the inverse primary and secondary transforms can be reduced. Referring back to the first and second inverse primary transforms, according to an aspect of the disclosure, the first inverse primary transform may calculate or keep a first number of residual data, and the second inverse primary transform may calculate or keep a second number of residual data where the second number of residual data is less than the first number of residual data so that a combined MPS (i.e., the second MPS) is similar to the 1<sup>st</sup> MPS. In an example, at least one one-dimensional transform (e.g., an inverse horizontal transform or an inverse vertical transform) of the second inverse primary transform includes the zero-out method. A one-dimensional transform of the first inverse primary transform can also include the zero-out method, however, the second inverse primary transform can calculate or keep less residual data than those of the first inverse primary transform.

**[0153]** As described above, an inverse primary transform and an inverse secondary transform can be applied to a TB and result in a combined MPS. In general, inverse primary transforms can include different types (e.g., DCT-2, DST-7, DST-4, DCT-8, DCT-4, or the like). According to aspects of the disclosure, whether to use the zero-out method and/or an amount of data to be calculated or kept in a coefficient block when using the zero-out method in the inverse secondary transform can be determined based on a type of the inverse primary transform, and thus the combined MPS can be reduced. For example, when the inverse primary transform is more complex (e.g., a MPS of the inverse primary transform is relatively large), an amount of data to be calculated or kept in the coefficient block when using the zero-out method in the inverse secondary transform can be further reduced.

**[0154]** According to aspects of the disclosure, the zero-out method can be performed depending on an intra prediction mode used for the TB. In some embodiments, whether to use the zero-out method and an amount of data to be calculated or kept when using the zero-out method can be determined based on the intra prediction mode.

**[0155]** In an embodiment, whether an inverse horizontal transform or an inverse vertical transform is performed with the zero-out method can depend on an intra prediction direction used for the TB. The intra prediction direction corresponds to the intra prediction mode. In an example, when the intra prediction direction is close to a horizontal prediction

direction, the zero-out method is applied for the inverse horizontal transform. In an example, when the intra prediction direction is close to a vertical prediction direction, the zero-out method is applied for the inverse vertical transform. For example, the horizontal prediction direction is indexed by a value  $Hor$ , and the vertical prediction direction is indexed by a value  $Ver$ . A threshold  $thres$  can be defined to determine a closeness of the intra prediction mode to a horizontal mode corresponding to the the horizontal prediction direction or a vertical mode corresponding to the the vertical prediction direction. The intra prediction mode (or the intra prediction direction) indexed within a range of  $[Hor - thres, Hor + thres]$  is determined to be close to the horizontal prediction direction. The intra prediction mode indexed within a range of  $[Ver - thres, Ver + thres]$  can be determined to be close to the vertical prediction direction. Example values of the threshold  $thres$  include, but are not limited to 1, 2, 3, 4, ..., and 16.

**[0156]** In an embodiment, an amount of data to be calculated or kept when using the zero-out method in the inverse primary transform and/or secondary transform can be determined based on the intra prediction direction for the TB.

**[0157]** FIG. 14 shows a flow chart outlining a process (1400) according to an embodiment of the disclosure. The process (1400) can be used in the reconstruction of a block coded in intra mode, so to generate a prediction block for the block under reconstruction. In some examples, the process (1400) can be used in the reconstruction of a block coded in inter mode. In various embodiments, the process (1400) 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 (1400) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (1400). The process starts at (S1401) and proceeds to (S1410).

**[0158]** At (S1410), coding information of a TB, such as a coefficient block, from a coded video bitstream can be decoded. The coding information can indicate a TB size (or referred to as a size of the coefficient block). As described above, the TB size can refer to an area, a width, a height, or the like of the TB.

**[0159]** At (S1420), an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed can be determined based on

the size of the coefficient block. Transform coefficients of the coefficient block can be transformed into residual data of a residual block by the inverse primary transform. As described above, in an example, the order is the first order. In an example, the order is the second order.

**[0160]** At (S1430), whether the TB size satisfies a condition can be determined, as described above. The condition can be whether the TB size is 32x64. The condition can be a height  $N$  of the coefficient block of  $M \times N$  is larger than a width  $M$  of the coefficient block. The condition can compare a first ratio and a second ratio where the first ratio (e.g.,  $m/M$ ) is an amount of data calculated for a row over a total amount of data in the row and the second ratio ( $n/N$ ) is an amount of data calculated for a column over a total amount of data in the column. When the TB size is determined to satisfy the condition, the process (1400) proceeds to (S1440). Otherwise, the process (1400) proceeds to (S1450).

**[0161]** At (S1440), the inverse vertical transform is performed after the inverse horizontal transform is performed on the transform coefficients of the coefficient block, and thus the second order is used.

**[0162]** At (S1450), the inverse horizontal transform is performed after the inverse vertical transform is performed on the transform coefficients of the coefficient block, and thus the first order is used.

**[0163]** At (S1460), a sample in the residual block can be reconstructed based on the residual data.

**[0164]** The process (1400) can be suitably adapted. For example, one or more steps can be modified, omitted, or combined. For example, (S1420) and (S1430) can be combined. Additional step(s) can also be added. An order that the process (1400) is executed can also be modified.

**[0165]** FIG. 15 shows a flow chart outlining a process (1500) according to an embodiment of the disclosure. The process (1500) can be used in the reconstruction of a block coded in intra mode, so to generate a prediction block for the block under reconstruction. In some examples, the process (1500) can be used in the reconstruction of a block coded in inter mode. In various embodiments, the process (1500) 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 (1500) is implemented in software instructions, thus when the processing circuitry executes the software instructions, the processing circuitry performs the process (1500). The process starts at (S1501) and proceeds to (S1510).

**[0166]** At (S1510), coding information of a TB, such as a coefficient block, from a coded video bitstream can be decoded. The coding information can indicate a TB size (or referred to as a size of the coefficient block).

**[0167]** At (S1520), whether to reduce a number of calculations (or an amount of data to be calculated) in one of inverse horizontal and inverse vertical transforms of an inverse primary transform can be determined based on the size of the coefficient block. Transform coefficients of the coefficient block can be transformed into residual data of a residual block by the inverse primary transform. In an embodiment, the first order is used. The inverse vertical transform can transform the transform coefficients of the coefficient block to intermediate data of an intermediate block, and the inverse horizontal transform can transform the intermediate data to the residual data in the residual block. In an example, reducing the number of calculations in the one-dimensional transform (e.g., the inverse horizontal transform or the inverse vertical transform) means the one-dimensional transform includes the zero-out method where first 16 rows (for the inverse horizontal transform) or columns (for the inverse vertical transform) of data are calculated and remaining data are not calculated. For example, when the TB size is one of 32x64 and 32x32, the number of calculations in the one-dimensional transform is determined to be reduced. When the number of calculations in the one-dimensional transform is determined to be reduced, the process (1500) proceeds to (S1530).

**[0168]** At (S1530), whether the number of calculations in the inverse horizontal transform is to be reduced can be determined. When the number of calculations in the inverse horizontal transform is to be reduced, the process (1500) proceeds to (S1540). Otherwise, the process (1500) proceeds to (S1550).

**[0169]** (S1530) can be suitably modified. For example, instead of determining whether the number of calculations in the inverse horizontal transform is to be reduced, at (S1530), whether the number of calculations in the inverse vertical transform is to be reduced can be determined.

**[0170]** At (S1540), the inverse primary transform is performed where the inverse horizontal transform includes the zero-out method, and left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform and the remaining residual data in the residual block are not calculated and are zero, as described above.

[0171] In an example, at (S1540), the inverse vertical transform also includes the zero-out method.

[0172] At (S1550), the inverse primary transform is performed where the inverse vertical transform includes the zero-out method, and thus top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are not calculated and are zero, as described above.

[0173] At (S1560), a sample in the residual block can be reconstructed based on the residual data.

[0174] The process (1500) can be suitably adapted. For example, one or more steps can be modified, omitted, or combined. For example, steps (S1520) and (S1530) can be combined into a single step. Additional step(s) can also be added. An order that the process (1500) is executed can also be modified.

[0175] In an example, step(s) in the process (1400) and step(s) in the process (1500) can be modified and combined.

[0176] 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. 16 shows a computer system (1600) suitable for implementing certain embodiments of the disclosed subject matter.

[0177] 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.

[0178] 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.

[0179] The components shown in FIG. 16 for computer system (1600) 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 (1600).

**[0180]** Computer system (1600) 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).

**[0181]** Input human interface devices may include one or more of (only one of each depicted): keyboard (1601), mouse (1602), trackpad (1603), touch screen (1610), data-glove (not shown), joystick (1605), microphone (1606), scanner (1607), camera (1608).

**[0182]** Computer system (1600) 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 (1610), data-glove (not shown), or joystick (1605), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (1609), headphones (not depicted)), visual output devices (such as screens (1610) 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).

**[0183]** Computer system (1600) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (1620) with CD/DVD or the like media (1621), thumb-drive (1622), removable hard drive or solid state drive (1623), 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.

**[0184]** 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.

**[0185]** Computer system (1600) can also include an interface to one or more communication networks. 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 (1649) (such as, for example USB ports of the computer system (1600)); others are commonly integrated into the core of the computer system (1600) 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 (1600) 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.

**[0186]** Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (1640) of the computer system (1600).

**[0187]** The core (1640) can include one or more Central Processing Units (CPU) (1641), Graphics Processing Units (GPU) (1642), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (1643), hardware accelerators for certain tasks (1644), and so forth. These devices, along with Read-only memory (ROM) (1645), Random-access memory (1646), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (1647), may be connected through a system bus (1648). In some computer systems, the system bus (1648) 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 (1648), or through a peripheral bus (1649). Architectures for a peripheral bus include PCI, USB, and the like.

**[0188]** CPUs (1641), GPUs (1642), FPGAs (1643), and accelerators (1644) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (1645) or RAM (1646). Transitional data can be also be stored in RAM (1646), whereas permanent data can be stored for example, in the internal mass storage (1647). 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 (1641), GPU (1642), mass storage (1647), ROM (1645), RAM (1646), and the like.

**[0189]** 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.

**[0190]** As an example and not by way of limitation, the computer system having architecture (1600), and specifically the core (1640) 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 (1640) that are of non-transitory nature, such as core-internal mass storage (1647) or ROM (1645). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (1640). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (1640) 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 (1646) 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 (1644)), 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

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  
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

[0191] 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.

APPENDIX I**4x4 transform**

```
{ 64, 64, 64, 64 }
{ 83, 36, -36, -83 }
{ 64, -64, -64, 64 }
{ 36, -83, 83, -36 }
```

**8x8 transform**

```
{ 64, 64, 64, 64, 64, 64, 64, 64 }
{ 89, 75, 50, 18, -18, -50, -75, -89 }
{ 83, 36, -36, -83, -83, -36, 36, 83 }
{ 75, -18, -89, -50, 50, 89, 18, -75 }
{ 64, -64, -64, 64, 64, -64, -64, 64 }
{ 50, -89, 18, 75, -75, -18, 89, -50 }
{ 36, -83, 83, -36, -36, 83, -83, 36 }
{ 18, -50, 75, -89, 89, -75, 50, -18 }
```

**16x16 transform**

```
{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ 90 87 80 70 57 43 25 9 -9 -25 -43 -57 -70 -80 -87 -90 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ 87 57 9 -43 -80 -90 -70 -25 25 70 90 80 43 -9 -57 -87 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ 80 9 -70 -87 -25 57 90 43 -43 -90 -57 25 87 70 -9 -80 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ 70 -43 -87 9 90 25 -80 -57 57 80 -25 -90 -9 87 43 -70 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 }
{ 57 -80 -25 90 -9 -87 43 70 -70 -43 87 9 -90 25 80 -57 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ 43 -90 57 25 -87 70 9 -80 80 -9 -70 87 -25 -57 90 -43 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36 }
{ 25 -70 90 -80 43 9 -57 87 -87 57 -9 -43 80 -90 70 -25 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ 9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9 }
```





{ ad, ae, af, ag, -ag, -af, -ae, -ad, -ad, -ae, -af, -ag, ag, af, ae, ad, ad, ae, af, ag, -ag, -af, -ae, -ad, -ad, -ae, -af, -ag, ag, af, ae, ad, ad, ae, af, ag, -ag, -af, -ae, -ad, -ad, -ae, -af, -ag, ag, af, ae, ad, ad, ae, af, ag, -ag, -af, -ae, -ad, -ad, -ae, -af, -ag, ag, af, ae, ad }

{ bj, bs, cb, ck, -cc, -bt, -bk, -bi, -br, -ca, -cj, cd, bu, bl, bh, bq, bz, ci, -ce, -bv, -bm, -bg, -bp, -by, -ch, cf, bw, bn, bf, bo, bx, cg, -cg, -bx, -bo, -bf, -bn, -bw, -cf, ch, by, bp, bg, bm, bv, ce, -ci, -bz, -bq, -bh, -bl, -bu, -cd, cj, ca, br, bi, bk, bt, ce, -ck, -cb, -bs, -bj }

{ ar, aw, bb, -bd, -ay, -at, -ap, -au, -az, -be, ba, av, aq, as, ax, bc, -bc, -ax, -as, -aq, -av, -ba, be, az, au, ap, at, ay, bd, -bb, -aw, -ar, -ar, -aw, -bb, bd, ay, at, ap, au, az, be, -ba, -av, -aq, -as, -ax, -bc, bc, ax, as, aq, av, ba, -be, -az, -au, -ap, -at, -ay, -bd, bb, aw, ar }

{ bk, bv, cg, -ce, -bt, -bi, -bm, -bx, -ci, cc, br, bg, bo, bz, ck, -ca, -bp, -bf, -bq, -cb, cj, by, bn, bh, bs, cd, -ch, -bw, -bl, -bj, -bu, -cf, cf, bu, bj, bl, bw, ch, -cd, -bs, -bh, -bn, -by, -cj, cb, bq, bf, bp, ca, -ck, -bz, -bo, -bg, -br, -cc, ci, bx, bm, bi, bt, ce, -cg, -bv, -bk }

{ ai, al, ao, -am, -aj, -ah, -ak, -an, an, ak, ah, aj, am, -ao, -al, -ai, -ai, -al, -ao, am, aj, ah, ak, an, -an, -ak, -ah, -aj, -am, ao, al, ai, ai, al, ao, -am, -aj, -ah, -ak, -an, an, ak, ah, aj, am, -ao, -al, -ai, -ai, -al, -ao, am, aj, ah, ak, an, -an, -ak, -ah, -aj, -am, ao, al, ai }

{ bl, by, -ck, -bx, -bk, -bm, -bz, cj, bw, bj, bn, ca, -ci, -bv, -bi, -bo, -cb, ch, bu, bh, bp, cc, -cg, -bt, -bg, -bq, -cd, cf, bs, bf, br, ce, -ce, -br, -bf, -bs, -cf, cd, bq, bg, bt, cg, -cc, -bp, -bh, -bu, -ch, cb, bo, bi, bv, ci, -ca, -bn, -bj, -bw, -cj, bz, bm, bk, bx, ck, -by, -bl }

{ as, az, -bd, -aw, -ap, -av, -bc, ba, at, ar, ay, -be, -ax, -aq, -au, -bb, bb, au, aq, ax, be, -ay, -ar, -at, -ba, bc, av, ap, aw, bd, -az, -as, -as, -az, bd, aw, ap, av, bc, -ba, -at, -ar, -ay, be, ax, aq, au, bb, -bb, -au, -aq, -ax, -be, ay, ar, at, ba, -bc, -av, -ap, -aw, -bd, az, as }

{ bm, cb, -cf, -bq, -bi, -bx, cj, bu, bf, bt, ci, -by, -bj, -bp, -ce, cc, bn, bl, ca, -cg, -br, -bh, -bw, ck, bv, bg, bs, ch, -bz, -bk, -bo, -cd, cd, bo, bk, bz, -ch, -bs, -bg, -bv, -ck, bw, bh, br, cg, -ca, -bl, -bn, -cc, ce, bp, bj, by, -ci, -bt, -bf, -bu, -cj, bx, bi, bq, cf, -cb, -bm }

{ ab, ac, -ac, -ab, -ab, -ac, ac, ab, ab, ac, -ac, -ab, -ab, -ac, ac, ab, ab, ac, -ac, -ab, -ab, -ac, ac, ab, ab, ac, -ac, -ab, -ab, -ac, ac, ab, ab, ac, -ac, -ab, -ab, -ac, ac, ab, ab, ac, -ac, -ab, -ab, -ac, ac, ab }

{ bn, ce, -ca, -bj, -br, -ci, bw, bf, bv, -cj, -bs, -bi, -bz, cf, bo, bm, cd, -cb, -bk, -bq, -ch, bx, bg, bu, -ck, -bt, -bh, -by, cg, bp, bl, cc, -cc, -bl, -bp, -cg, by, bh, bt, ck, -bu, -bg, -bx, ch, bq, bk, cb, -cd, -bm, -bo, -cf, bz, bi, bs, cj, -bv, -bf, -bw, ci, br, bj, ca, -ce, -bn }

{ at, bc, -ay, -ap, -ax, bd, au, as, bb, -az, -aq, -aw, be, av, ar, ba, -ba, -ar, -av, -be, aw, aq, az, -bb, -as, -au, -bd, ax, ap, ay, -bc, -at, -at, -bc, ay, ap, ax, -bd, -au, -as, -bb, az, aq, aw, -be, -av, -ar, -ba, ba, ar, av, be, -aw, -aq, -az, bb, as, au, bd, -ax, -ap, -ay, bc, at }

{ bo, ch, -bv, -bh, -ca, cc, bj, bt, -cj, -bq, -bm, -cf, bx, bf, by, -ce, -bl, -br, -ck, bs, bk, cd, -bz, -bg, -bw, cg, bn, bp, ci, -bu, -bi, -cb, cb, bi, bu, -ci, -bp, -bn, -cg, bw, bg, bz, -cd, -bk, -bs, ck, br, bl, ce, -by, -bf, -bx, cf, bm, bq, cj, -bt, -bj, -cc, ca, bh, bv, -ch, -bo }

{ aj, ao, -ak, -ai, -an, al, ah, am, -am, -ah, -al, an, ai, ak, -ao, -aj, -aj, -ao, ak, ai, an, -al, -ah, -am, am, ah, al, -an, -ai, -ak, ao, aj, aj, ao, -ak, -ai, -an, al, ah, am, -am, -ah, -al, an, ai, ak, -ao, -aj, -aj, -ao, ak, ai, an, -al, -ah, -am, am, ah, al, -an, -ai, -ak, ao, aj }

{ bp, ck, -bq, -bo, -cj, br, bn, ci, -bs, -bm, -ch, bt, bl, cg, -bu, -bk, -cf, bv, bj, ce, -bw, -bi, -cd, bx, bh, cc, -by, -bg, -cb, bz, bf, ca, -ca, -bf, -bz, cb, bg, by, -cc, -bh, -bx, cd, bi, bw, -ce, -bj, -bv, cf, bk, bu, -cg, -bl, -bt, ch, bm, bs, -ci, -bn, -br, cj, bo, bq, -ck, -bp }

{ au, -be, -at, -av, bd, as, aw, -bc, -ar, -ax, bb, aq, ay, -ba, -ap, -az, az, ap, ba, -ay, -aq, -bb, ax, ar, bc, -aw, -as, -bd, av, at, be, -au, -au, be, at, av, -bd, -as, -aw, bc, ar, ax, -bb, -aq, -ay, ba, ap, az, -az, -ap, -ba, ay, aq, bb, -ax, -ar, -bc, aw, as, bd, -av, -at, -be, au }

{ bq, -ci, -bl, -bv, cd, bg, ca, -by, -bi, -cf, bt, bn, ck, -bo, -bs, cg, bj, bx, -cb, -bf, -cc, bw, bk, ch, -br, -bp, cj, bm, bu, -ce, -bh, -bz, bz, bh, ce, -bu, -bm, -cj, bp, br, -ch, -bk, -bw, cc, bf, cb, -bx, -bj, -cg, bs, bo, -ck, -bn, -bt, cf, bi, by, -ca, -bg, -cd, bv, bl, ci, -bq }

{ ae, -ag, -ad, -af, af, ad, ag, -ae, -ae, ag, ad, af, -af, -ad, -ag, ae, ae, -ag, -ad, -af, af, ad, ag, -ae, -ae, ag, ad, af, -af, -ad, -ag, ae, ae, -ag, -ad, -af, af, ad, ag, -ae, -ae, ag, ad, af, -af, -ad, -ag, ae, ae, -ag, -ad, -af, af, ad, ag, -ae, -ae, ag, ad, af, -af, -ad, -ag, ae }

{ br, -cf, -bg, -cc, bu, bo, -ci, -bj, -bz, bx, bl, ck, -bm, -bw, ca, bi, ch, -bp, -bt, cd, bf, ce, -bs, -bq, cg, bh, cb, -bv, -bn, cj, bk, by, -by, -bk, -cj, bn, bv, -cb, -bh, -cg, bq, bs, -ce, -bf, -cd, bt, bp, -ch, -bi, -ca, bw, bm, -ck, -bl, -bx, bz, bj, ci, -bo, -bu, cc, bg, cf, -br }

{ av, -bb, -ap, -bc, au, aw, -ba, -aq, -bd, at, ax, -az, -ar, -be, as, ay, -ay, -as, be, ar, az, -ax, -at, bd, aq, ba, -aw, -au, bc, ap, bb, -av, -av, bb, ap, bc, -au, -aw, ba, aq, bd, -at, -ax, az, ar, be, -as, -ay, ay, as, -be, -ar, -az, ax, at, -bd, -aq, -ba, aw, au, -bc, -ap, -bb, av }

{ bs, -cc, -bi, -cj, bl, bz, -bv, -bp, cf, bf, cg, -bo, -bw, by, bm, -ci, -bh, -cd, br, bt, -cb, -bj, -ck, bk, ca, -bu, -bq, ce, bg, ch, -bn, -bx, bx, bn, -ch, -bg, -ce, bq, bu, -ca, -bk, ck, bj, cb, -bt, -br, cd, bh, ci, -bm, -by, bw, bo, -cg, -bf, -cf, bp, bv, -bz, -bl, cj, bi, cc, -bs }

{ ak, -am, -ai, ao, ah, an, -aj, -al, al, aj, -an, -ah, -ao, ai, am, -ak, -ak, am, ai, -ao, -ah, -an, aj, al, -al, -aj, an, ah, ao, -ai, -am, ak, ak, -am, -ai, ao, ah, an, -aj, -al, al, aj, -an, -ah, -ao, ai, am, -ak, -ak, am, ai, -ao, -ah, -an, aj, al, -al, -aj, an, ah, ao, -ai, -am, ak }

{ bt, -bz, -bn, cf, bh, ck, -bi, -ce, bo, by, -bu, -bs, ca, bm, -cg, -bg, -cj, bj, cd, -bp, -bx, bv, br, -cb, -bl, ch, bf, ci, -bk, -cc, bq, bw, -bw, -bq, cc, bk, -ci, -bf, -ch, bl, cb, -br, -bv, bx, bp, -cd, -bj, cj, bg, cg, -bm, -ca, bs, bu, -by, -bo, ce, bi, -ck, -bh, -cf, bn, bz, -bt }

{ aw, -ay, -au, ba, as, -bc, -aq, be, ap, bd, -ar, -bb, at, az, -av, -ax, ax, av, -az, -at, bb, ar, -bd, -ap, -be, aq, bc, -as, -ba, au, ay, -aw, -aw, ay, au, -ba, -as, bc, aq, -be, -ap, -bd, ar, bb, -at, -az, av, ax, -ax, -av, az, at, -bb, -ar, bd, ap, be, -aq, -bc, as, ba, -au, -ay, aw }

{ bu, -bw, -bs, by, bq, -ca, -bo, cc, bm, -ce, -bk, cg, bi, -ci, -bg, ck, bf, cj, -bh, -ch, bj, cf, -bl, -cd, bn, cb, -bp, -bz, br, bx, -bt, -bv, bv, bt, -bx, -br, bz, bp, -cb, -bn, cd, bl, -cf, -bj, ch, bh, -cj, -bf, -ck, bg, ci, -bi, -cg, bk, ce, -bm, -cc, bo, ca, -bq, -by, bs, bw, -bu }

{ aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa, aa, -aa, -aa, aa }

{ bv, -bt, -bx, br, bz, -bp, -cb, bn, cd, -bl, -cf, bj, ch, -bh, -cj, bf, -ck, -bg, ci, bi, -cg, -bk, ce, bm, -cc, -bo, ca, bq, -by, -bs, bw, bu, -bu, -bw, bs, by, -bq, -ca, bo, cc, -bm, -ce, bk, cg, -bi, -ci, bg, ck, -bf, cj, bh, -ch, -bj, cf, bl, -cd, -bn, cb, bp, -bz, -br, bx, bt, -bv }

{ ax, -av, -az, at, bb, -ar, -bd, ap, -be, -aq, bc, as, -ba, -au, ay, aw, -aw, -ay, au, ba, -as, -bc, aq, be, -ap, bd, ar, -bb, -at, az, av, -ax, -ax, av, az, -at, -bb, ar, bd, -ap, be, aq, -bc, -as, ba, au, -ay, -aw, aw, ay, -au, -ba, as, bc, -aq, -be, ap, -bd, -ar, bb, at, -az, -av, ax }

{ bw, -bq, -cc, bk, ci, -bf, ch, bl, -cb, -br, bv, bx, -bp, -cd, bj, cj, -bg, cg, bm, -ca, -bs, bu, by, -bo, -ce, bi, ck, -bh, cf, bn, -bz, -bt, bt, bz, -bn, -cf, bh, -ck, -bi, ce, bo, -by, -bu, bs, ca, -bm, -cg, bg, -cj, -bj, cd, bp, -bx, -bv, br, cb, -bl, -ch, bf, -ci, -bk, cc, bq, -bw }

{ al, -aj, -an, ah, -ao, -ai, am, ak, -ak, -am, ai, ao, -ah, an, aj, -al, -al, aj, an, -ah, ao, ai, -am, -ak, ak, am, -ai, -ao, ah, -an, -aj, al, al, -aj, -an, ah, -ao, -ai, am, ak, -ak, -am, ai, ao, -ah, an, aj, -al, -al, aj, an, -ah, ao, ai, -am, -ak, ak, am, -ai, -ao, ah, -an, -aj, al }

{ bx, -bn, -ch, bg, -ce, -bq, bu, ca, -bk, -ck, bj, -cb, -bt, br, cd, -bh, ci, bm, -by, -bw, bo, cg, -bf, cf, bp, -bv, -bz, bl, cj, -bi, cc, bs, -bs, -cc, bi, -cj, -bl, bz, bv, -bp, -cf, bf, -cg, -bo, bw, by, -bm, -ci, bh, -cd, -br, bt, cb, -bj, ck, bk, -ca, -bu, bq, ce, -bg, ch, bn, -bx }

{ ay, -as, -be, ar, -az, -ax, at, bd, -aq, ba, aw, -au, -bc, ap, -bb, -av, av, bb, -ap, bc, au, -aw, -ba, aq, -bd, -at, ax, az, -ar, be, as, -ay, -ay, as, be, -ar, az, ax, -at, -bd, aq, -ba, -aw, au, bc, -ap, bb, av, -av, -bb, ap, -bc, -au, aw, ba, -aq, bd, at, -ax, -az, ar, -be, -as, ay }

{ by, -bk, cj, bn, -bv, -cb, bh, -cg, -bq, bs, ce, -bf, cd, bt, -bp, -ch, bi, -ca, -bw, bm, ck, -bl, bx, bz, -bj, ci, bo, -bu, -cc, bg, -cf, -br, br, cf, -bg, cc, bu, -bo, -ci, bj, -bz, -bx, bl, -ck, -bm, bw, ca, -bi, ch, bp, -bt, -cd, bf, -ce, -bs, bq, cg, -bh, cb, bv, -bn, -cj, bk, -by }

{ af, -ad, ag, ae, -ae, -ag, ad, -af, -af, ad, -ag, -ae, ae, ag, -ad, af, af, -ad, ag, ae, -ae, -ag, ad, -af, -af, ad, -ag, -ae, ae, ag, -ad, af, af, -ad, ag, ae, -ae, -ag, ad, -af, -af, ad, -ag, -ae, ae, ag, -ad, af, af, -ad, ag, ae, -ae, -ag, ad, -af, -af, ad, -ag, -ae, ae, ag, -ad, af }

{ bz, -bh, ce, bu, -bm, cj, bp, -br, -ch, bk, -bw, -cc, bf, -cb, -bx, bj, -cg, -bs, bo, ck, -bn, bt, cf, -bi, by, ca, -bg, cd, bv, -bl, ci, bq, -bq, -ci, bl, -bv, -cd, bg, -ca, -by, bi, -cf, -bt, bn, -ck, -bo, bs, cg, -bj, bx, cb, -bf, cc, bw, -bk, ch, br, -bp, -cj, bm, -bu, -ce, bh, -bz }

{ az, -ap, ba, ay, -aq, bb, ax, -ar, bc, aw, -as, bd, av, -at, be, au, -au, -be, at, -av, -bd, as, -aw, -bc, ar, -ax, -bb, aq, -ay, -ba, ap, -az, -az, ap, -ba, -ay, aq, -bb, -ax, ar, -bc, -aw, as, -bd, -av, at, -be, -au, au, be, -at, av, bd, -as, aw, bc, -ar, ax, bb, -aq, ay, ba, -ap, az }

{ ca, -bf, bz, cb, -bg, by, cc, -bh, bx, cd, -bi, bw, ce, -bj, bv, cf, -bk, bu, cg, -bl, bt, ch, -bm, bs, ci, -bn, br, cj, -bo, bq, ck, -bp, bp, -ck, -bq, bo, -cj, -br, bn, -ci, -bs, bm, -ch, -bt, bl, -cg, -bu, bk, -cf, -bv, bj, -ce, -bw, bi, -cd, -bx, bh, -cc, -by, bg, -cb, -bz, bf, -ca }

{ am, -ah, al, an, -ai, ak, ao, -aj, aj, -ao, -ak, ai, -an, -al, ah, -am, -am, ah, -al, -an, ai, -ak, -ao, aj, -aj, ao, ak, -ai, an, al, -ah, am, am, -ah, al, an, -ai, ak, ao, -aj, aj, -ao, -ak, ai, -an, -al, ah, -am, -am, ah, -al, -an, ai, -ak, -ao, aj, -aj, ao, ak, -ai, an, al, -ah, am }

{ cb, -bi, bu, ci, -bp, bn, -cg, -bw, bg, -bz, -cd, bk, -bs, -ck, br, -bl, ce, by, -bf, bx, cf, -bm, bq, -cj, -bt, bj, -cc, -ca, bh, -bv, -ch, bo, -bo, ch, bv, -bh, ca, cc, -bj, bt, cj, -bq, bm, -cf, -bx, bf, -by, -ce, bl, -br, ck, bs, -bk, cd, bz, -bg, bw, cg, -bn, bp, -ci, -bu, bi, -cb }

{ ba, -ar, av, -be, -aw, aq, -az, -bb, as, -au, bd, ax, -ap, ay, bc, -at, at, -bc, -ay, ap, -ax, -bd, au, -as, bb, az, -aq, aw, be, -av, ar, -ba, -ba, ar, -av, be, aw, -aq, az, bb, -as, au, -bd, -ax, ap, -ay, -bc, at, -at, bc, ay, -ap, ax, bd, -au, as, -bb, -az, aq, -aw, -be, av, -ar, ba }

{ cc, -bl, bp, -cg, -by, bh, -bt, ck, bu, -bg, bx, ch, -bq, bk, -cb, -cd, bm, -bo, cf, bz, -bi, bs, -cj, -bv, bf, -bw, -ci, br, -bj, ca, ce, -bn, bn, -ce, -ca, bj, -br, ci, bw, -bf, bv, cj, -bs, bi, -bz, -cf, bo, -bm, cd, cb, -bk, bq, -ch, -bx, bg, -bu, -ck, bt, -bh, by, cg, -bp, bl, -cc }

{ ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac, ac, -ab, ab, -ac, -ac, ab, -ab, ac }

{ cd, -bo, bk, -bz, -ch, bs, -bg, bv, -ck, -bw, bh, -br, cg, ca, -bl, bn, -cc, -ce, bp, -bj, by, ci, -bt, bf, -bu, cj, bx, -bi, bq, -cf, -cb, bm, -bm, cb, cf, -bq, bi, -bx, -cj, bu, -bf, bt, -ci, -by, bj, -bp, ce, cc, -bn, bl, -ca, -cg, br, -bh, bw, ck, -bv, bg, -bs, ch, bz, -bk, bo, -cd }

{ bb, -au, aq, -ax, be, ay, -ar, at, -ba, -bc, av, -ap, aw, -bd, -az, as, -as, az, bd, -aw, ap, -av, bc, ba, -at, ar, -ay, -be, ax, -aq, au, -bb, -bb, au, -aq, ax, -be, -ay, ar, -at, ba, bc, -av, ap, -aw, bd, az, -as, as, -az, -bd, aw, -ap, av, -bc, -ba, at, -ar, ay, be, -ax, aq, -au, bb }

{ ce, -br, bf, -bs, cf, cd, -bq, bg, -bt, cg, cc, -bp, bh, -bu, ch, cb, -bo, bi, -bv, ci, ca, -bn, bj, -bw, cj, bz, -bm, bk, -bx, ck, by, -bl, bl, -by, -ck, bx, -bk, bm, -bz, -cj, bw, -bj, bn, -ca, -ci, bv, -bi, bo, -cb, -ch, bu, -bh, bp, -cc, -cg, bt, -bg, bq, -cd, -cf, bs, -bf, br, -ce }

{ an, -ak, ah, -aj, am, ao, -al, ai, -ai, al, -ao, -am, aj, -ah, ak, -an, -an, ak, -ah, aj, -am, -ao, al, -ai, ai, -al, ao, am, -aj, ah, -ak, an, an, -ak, ah, -aj, am, ao, -al, ai, -ai, al, -ao, -am, aj, -ah, ak, -an, -an, ak, -ah, aj, -am, -ao, al, -ai, ai, -al, ao, am, -aj, ah, -ak, an }

{ cf, -bu, bj, -bl, bw, -ch, -cd, bs, -bh, bn, -by, cj, cb, -bq, bf, -bp, ca, ck, -bz, bo, -bg, br, -cc, -ci, bx, -bm, bi, -bt, ce, cg, -bv, bk, -bk, bv, -cg, -ce, bt, -bi, bm, -bx, ci, cc, -br, bg, -bo, bz, -ck, -ca, bp, -bf, bq, -cb, -cj, by, -bn, bh, -bs, cd, ch, -bw, bl, -bj, bu, -cf }

{ bc, -ax, as, -aq, av, -ba, -be, az, -au, ap, -at, ay, -bd, -bb, aw, -ar, ar, -aw, bb, bd, -ay, at, -ap, au, -az, be, ba, -av, aq, -as, ax, -bc, -bc, ax, -as, aq, -av, ba, be, -az, au, -ap, at, -ay, bd, bb, -aw, ar, -ar, aw, -bb, -bd, ay, -at, ap, -au, az, -be, -ba, av, -aq, as, -ax, bc }

{ cg, -bx, bo, -bf, bn, -bw, cf, ch, -by, bp, -bg, bm, -bv, ce, ci, -bz, bq, -bh, bl, -bu, cd, cj, -ca, br, -bi, bk, -bt, cc, ck, -cb, bs, -bj, bj, -bs, cb, -ck, -cc, bt, -bk, bi, -br, ca, -cj, -cd, bu, -bl, bh, -bq, bz, -ci, -ce, bv, -bm, bg, -bp, by, -ch, -cf, bw, -bn, bf, -bo, bx, -cg }

{ ag, -af, ae, -ad, ad, -ae, af, -ag, -ag, af, -ae, ad, -ad, ae, -af, ag, ag, -af, ae, -ad, ad, -ae, af, -ag, -ag, af, -ae, ad, -ad, ae, -af, ag, ag, -af, ae, -ad, ad, -ae, af, -ag, -ag, af, -ae, ad, -ad, ae, -af, ag, ag, -af, ae, -ad, ad, -ae, af, -ag, -ag, af, -ae, ad, -ad, ae, -af, ag }

{ ch, -ca, bt, -bm, bf, -bl, bs, -bz, cg, ci, -cb, bu, -bn, bg, -bk, br, -by, cf, cj, -cc, bv, -bo, bh, -bj, bq, -bx, ce, ck, -cd, bw, -bp, bi, -bi, bp, -bw, cd, -ck, -ce, bx, -bq, bj, -bh, bo, -bv, cc, -cj, -cf, by, -br, bk, -bg, bn, -bu, cb, -ci, -cg, bz, -bs, bl, -bf, bm, -bt, ca, -ch }

{ bd, -ba, ax, -au, ar, -ap, as, -av, ay, -bb, be, bc, -az, aw, -at, aq, -aq, at, -aw, az, -bc, -be, bb, -ay, av, -as, ap, -ar, au, -ax, ba, -bd, -bd, ba, -ax, au, -ar, ap, -as, av, -ay, bb, -be, -bc, az, -aw, at, -aq, aq, -at, aw, -az, bc, be, -bb, ay, -av, as, -ap, ar, -au, ax, -ba, bd }

{ ci, -cd, by, -bt, bo, -bj, bf, -bk, bp, -bu, bz, -ce, cj, ch, -cc, bx, -bs, bn, -bi, bg, -bl, bq, -bv, ca, -cf, ck, cg, -cb, bw, -br, bm, -bh, bh, -bm, br, -bw, cb, -cg, -ck, cf, -ca, bv, -bq, bl, -bg, bi, -bn, bs, -bx, cc, -ch, -cj, ce, -bz, bu, -bp, bk, -bf, bj, -bo, bt, -by, cd, -ci }

{ ao, -an, am, -al, ak, -aj, ai, -ah, ah, -ai, aj, -ak, al, -am, an, -ao, -ao, an, -am, al, -ak, aj, -ai, ah, -ah, ai, -aj, ak, -al, am, -an, ao, ao, -an, am, -al, ak, -aj, ai, -ah, ah, -ai, aj, -ak, al, -am, an, -ao, -ao, an, -am, al, -ak, aj, -ai, ah, -ah, ai, -aj, ak, -al, am, -an, ao }

{ cj, -cg, cd, -ca, bx, -bu, br, -bo, bl, -bi, bf, -bh, bk, -bn, bq, -bt, bw, -bz, cc, -cf, ci, ck, -ch, ce, -cb, by, -bv, bs, -bp, bm, -bj, bg, -bg, bj, -bm, bp, -bs, bv, -by, cb, -ce, ch, -ck, -ci, cf, -cc, bz, -bw, bt, -bq, bn, -bk, bh, -bf, bi, -bl, bo, -br, bu, -bx, ca, -cd, cg, -cj }

{ be, -bd, bc, -bb, ba, -az, ay, -ax, aw, -av, au, -at, as, -ar, aq, -ap, ap, -aq, ar, -as, at, -au, av, -aw, ax, -ay, az, -ba, bb, -bc, bd, -be, -be, bd, -bc, bb, -ba, az, -ay, ax, -aw, av, -au, at, -as, ar, -aq, ap, -ap, aq, -ar, as, -at, au, -av, aw, -ax, ay, -az, ba, -bb, bc, -bd, be }

{ ck, -cj, ci, -ch, cg, -cf, ce, -cd, cc, -cb, ca, -bz, by, -bx, bw, -bv, bu, -bt, bs, -br, bq, -bp, bo, -bn, bm, -bl, bk, -bj, bi, -bh, bg, -bf, bf, -bg, bh, -bi, bj, -bk, bl, -bm, bn, -bo, bp, -bq, br, -bs, bt, -bu, bv, -bw, bx, -by, bz, -ca, cb, -cc, cd, -ce, cf, -cg, ch, -ci, cj, -ck }

}

where

{ aa, ab, ac, ad, ae, af, ag, ah, ai, aj, ak, al, am, an, ao, ap, aq, ar, as, at, au, av, aw, ax, ay, az, ba, bb, bc, bd, be, bf, bg, bh, bi, bj, bk, bl, bm, bn, bo, bp, bq, br, bs, bt, bu, bv, bw, bx, by, bz, ca, cb, cc, cd, ce, cf, cg, ch, ci, cj, ck} =

{64,83,36,89,75,50,18,90,87,80,70,57,43,25,9,90,90,88,85,82,78,73,67,61,54,46,38,31,22,13, 4,91,90,90,90,88,87,86,84,83,81,79,77,73,71,69,65,62,59,56,52,48,44,41,37,33,28,24,20,15,1 1,7,2}

### APPENDIX III

#### **4-point DST-7**

{ a, b, c, d }

{ c, c, 0, -c }

{ d, -a, -c, b }

{ b, -d, c, -a }

where {a, b, c, d} = { 29, 55, 74, 84}

#### **8-point DST-7:**

{ a, b, c, d, e, f, g, h, }

{ c, f, h, e, b, -a, -d, -g, }

{ e, g, b, -c, -h, -d, a, f, }

{ g, c, -d, -f, a, h, b, -e, }

{ h, -a, -g, b, f, -c, -e, d, }

{ f, -e, -a, g, -d, -b, h, -c, }

{ d, -h, e, -a, -c, g, -f, b, }

{ b, -d, f, -h, g, -e, c, -a, }

where {a, b, c, d, e, f, g, h} = { 17, 32, 46, 60, 71, 78, 85, 86}

#### **16-point DST-7**

{ a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, }

{ c, f, i, l, o, o, l, i, f, c, 0, -c, -f, -i, -l, -o, }

{ e, j, o, m, h, c, -b, -g, -l, -p, -k, -f, -a, d, i, n, }

{ g, n, l, e, -b, -i, -p, -j, -c, d, k, o, h, a, -f, -m, }

- { i, o, f, -c, -l, -l, -c, f, o, i, 0, -i, -o, -f, c, l, }
- { k, k, 0, -k, -k, 0, k, k, 0, -k, -k, 0, k, k, 0, -k, }
- { m, g, -f, -n, -a, l, h, -e, -o, -b, k, i, -d, -p, -c, j, }
- { o, c, -l, -f, i, i, -f, -l, c, o, 0, -o, -c, l, f, -i, }
- { p, -a, -o, b, n, -c, -m, d, l, -e, -k, f, j, -g, -i, h, }
- { n, -e, -i, j, d, -o, a, m, -f, -h, k, c, -p, b, l, -g, }
- { l, -i, -c, o, -f, -f, o, -c, -i, l, 0, -l, i, c, -o, f, }
- { j, -m, c, g, -p, f, d, -n, i, a, -k, l, -b, -h, o, -e, }
- { h, -p, i, -a, -g, o, -j, b, f, -n, k, -c, -e, m, -l, d, }
- { f, -l, o, -i, c, c, -i, o, -l, f, 0, -f, l, -o, i, -c, }
- { d, -h, l, -p, m, -i, e, -a, -c, g, -k, o, -n, j, -f, b, }
- { b, -d, f, -h, j, -l, n, -p, o, -m, k, -i, g, -e, c, -a, }

where {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p} = { 9, 17, 25, 33, 41, 49, 56, 62, 66, 72, 77, 81, 83, 87, 89, 90}

**32-point DST-7**

- { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, }
- { c, f, i, l, o, r, u, x, A, D, F, C, z, w, t, q, n, k, h, e, b, -a, -d, -g, -j, -m, -p, -s, -v, -y, -B, -E, }
- { e, j, o, t, y, D, D, y, t, o, j, e, 0, -e, -j, -o, -t, -y, -D, -D, -y, -t, -o, -j, -e, 0, e, j, o, t, y, D, }
- { g, n, u, B, D, w, p, i, b, -e, -l, -s, -z, -F, -y, -r, -k, -d, c, j, q, x, E, A, t, m, f, -a, -h, -o, -v, -C, }
- { i, r, A, C, t, k, b, -g, -p, -y, -E, -v, -m, -d, e, n, w, F, x, o, f, -c, -l, -u, -D, -z, -q, -h, a, j, s, B, }
- { k, v, F, u, j, -a, -l, -w, -E, -t, -i, b, m, x, D, s, h, -c, -n, -y, -C, -r, -g, d, o, z, B, q, f, -e, -p, -A, }
- { m, z, z, m, 0, -m, -z, -z, -m, 0, m, z, z, m, 0, -m, -z, -z, -m, 0, m, z, z, m, 0, -m, -z, -z, -m, 0, m, z, }
- { o, D, t, e, -j, -y, -y, -j, e, t, D, o, 0, -o, -D, -t, -e, j, y, y, j, -e, -t, -D, -o, 0, o, D, t, e, -j, -y, }
- { q, E, n, -c, -t, -B, -k, f, w, y, h, -i, -z, -v, -e, l, C, s, b, -o, -F, -p, a, r, D, m, -d, -u, -A, -j, g, x, }

{ s, A, h, -k, -D, -p, c, v, x, e, -n, -F, -m, f, y, u, b, -q, -C, -j, i, B, r, -a, -t, -z, -g, l, E, o, -d, -w, }

{ u, w, b, -s, -y, -d, q, A, f, -o, -C, -h, m, E, j, -k, -F, -l, i, D, n, -g, -B, -p, e, z, r, -c, -x, -t, a, v, }

{ w, s, -d, -A, -o, h, E, k, -l, -D, -g, p, z, c, -t, -v, a, x, r, -e, -B, -n, i, F, j, -m, -C, -f, q, y, b, -u, }

{ y, o, -j, -D, -e, t, t, -e, -D, -j, o, y, 0, -y, -o, j, D, e, -t, -t, e, D, j, -o, -y, 0, y, o, -j, -D, -e, t, }

{ A, k, -p, -v, e, F, f, -u, -q, j, B, a, -z, -l, o, w, -d, -E, -g, t, r, -i, -C, -b, y, m, -n, -x, c, D, h, -s, }

{ C, g, -v, -n, o, u, -h, -B, a, D, f, -w, -m, p, t, -i, -A, b, E, e, -x, -l, q, s, -j, -z, c, F, d, -y, -k, r, }

{ E, c, -B, -f, y, i, -v, -l, s, o, -p, -r, m, u, -j, -x, g, A, -d, -D, a, F, b, -C, -e, z, h, -w, -k, t, n, -q, }

{ F, -a, -E, b, D, -c, -C, d, B, -e, -A, f, z, -g, -y, h, x, -i, -w, j, v, -k, -u, l, t, -m, -s, n, r, -o, -q, p, }

{ D, -e, -y, j, t, -o, -o, t, j, -y, -e, D, 0, -D, e, y, -j, -t, o, o, -t, -j, y, e, -D, 0, D, -e, -y, j, t, -o, }

{ B, -i, -s, r, j, -A, -a, C, -h, -t, q, k, -z, -b, D, -g, -u, p, l, -y, -c, E, -f, -v, o, m, -x, -d, F, -e, -w, n, }

{ z, -m, -m, z, 0, -z, m, m, -z, 0, z, -m, -m, z, 0, -z, m, m, -z, 0, z, -m, -m, z, 0, -z, m, m, -z, 0, z, -m, }

{ x, -q, -g, E, -j, -n, A, -c, -u, t, d, -B, m, k, -D, f, r, -w, -a, y, -p, -h, F, -i, -o, z, -b, -v, s, e, -C, l, }

{ v, -u, -a, w, -t, -b, x, -s, -c, y, -r, -d, z, -q, -e, A, -p, -f, B, -o, -g, C, -n, -h, D, -m, -i, E, -l, -j, F, -k, }

{ t, -y, e, o, -D, j, j, -D, o, e, -y, t, 0, -t, y, -e, -o, D, -j, -j, D, -o, -e, y, -t, 0, t, -y, e, o, -D, j, }

{ r, -C, k, g, -y, v, -d, -n, F, -o, -c, u, -z, h, j, -B, s, -a, -q, D, -l, -f, x, -w, e, m, -E, p, b, -t, A, -i, }

{ p, -F, q, -a, -o, E, -r, b, n, -D, s, -c, -m, C, -t, d, l, -B, u, -e, -k, A, -v, f, j, -z, w, -g, -i, y, -x, h, }

{ n, -B, w, -i, -e, s, -F, r, -d, -j, x, -A, m, a, -o, C, -v, h, f, -t, E, -q, c, k, -y, z, -l, -b, p, -D, u, -g, }

{ l, -x, C, -q, e, g, -s, E, -v, j, b, -n, z, -A, o, -c, -i, u, -F, t, -h, -d, p, -B, y, -m, a, k, -w, D, -r, f, }

{ j, -t, D, -y, o, -e, -e, o, -y, D, -t, j, 0, -j, t, -D, y, -o, e, e, -o, y, -D, t, -j, 0, j, -t, D, -y, o, -e, }

{ h, -p, x, -F, y, -q, i, -a, -g, o, -w, E, -z, r, -j, b, f, -n, v, -D, A, -s, k, -c, -e, m, -u, C, -B, t, -l, d, }

{ f, -l, r, -x, D, -C, w, -q, k, -e, -a, g, -m, s, -y, E, -B, v, -p, j, -d, -b, h, -n, t, -z, F, -A, u, -o, i, -c, }

{ d, -h, l, -p, t, -x, B, -F, C, -y, u, -q, m, -i, e, -a, -c, g, -k, o, -s, w, -A, E, -D, z, -v, r, -n, j, -f, b, }

{ b, -d, f, -h, j, -l, n, -p, r, -t, v, -x, z, -B, D, -F, E, -C, A, -y, w, -u, s, -q, o, -m, k, -i, g, -e, c, -a, }

where {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F} = { 4, 9, 13, 17, 21, 26, 30, 34, 38, 42, 45, 50, 53, 56, 60, 63, 66, 68, 72, 74, 77, 78, 80, 82, 84, 85, 86, 88, 88, 89, 90, 90 }

#### 4-point DCT-8

{ a, b, c, d, }

{ b, 0, -b, -b, }

{ c, -b, -d, a, }

{ d, -b, a, -c, }

where {a, b, c, d} = { 84, 74, 55, 29 }

#### 8-point DCT-8:

{ a, b, c, d, e, f, g, h, }

{ b, e, h, -g, -d, -a, -c, -f, }

{ c, h, -e, -a, -f, g, b, d, }

{ d, -g, -a, -h, c, e, -f, -b, }

{ e, -d, -f, c, g, -b, -h, a, }

{ f, -a, g, e, -b, h, d, -c, }

{ g, -c, b, -f, -h, d, -a, e, }

{ h, -f, d, -b, a, -c, e, -g, }

where {a, b, c, d, e, f, g, h} = { 86, 85, 78, 71, 60, 46, 32, 17 }

**16-point DCT-8**

- { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p,}
- { b, e, h, k, n, 0, -n, -k, -h, -e, -b, -b, -e, -h, -k, -n,}
- { c, h, m, -p, -k, -f, -a, -e, -j, -o, n, i, d, b, g, l,}
- { d, k, -p, -i, -b, -f, -m, n, g, a, h, o, -l, -e, -c, -j,}
- { e, n, -k, -b, -h, 0, h, b, k, -n, -e, -e, -n, k, b, h,}
- { f, 0, -f, -f, 0, f, f, 0, -f, -f, 0, f, f, 0, -f, -f,}
- { g, -n, -a, -m, h, f, -o, -b, -l, i, e, -p, -c, -k, j, d,}
- { h, -k, -e, n, b, 0, -b, -n, e, k, -h, -h, k, e, -n, -b,}
- { i, -h, -j, g, k, -f, -l, e, m, -d, -n, c, o, -b, -p, a,}
- { j, -e, -o, a, -n, -f, i, k, -d, -p, b, -m, -g, h, l, -c,}
- { k, -b, n, h, -e, 0, e, -h, -n, b, -k, -k, b, -n, -h, e,}
- { l, -b, i, o, -e, f, -p, -h, c, -m, -k, a, -j, -n, d, -g,}
- { m, -e, d, -l, -n, f, -c, k, o, -g, b, -j, -p, h, -a, i,}
- { n, -h, b, -e, k, 0, -k, e, -b, h, -n, -n, h, -b, e, -k,}
- { o, -k, g, -c, b, -f, j, -n, -p, l, -h, d, -a, e, -i, m,}
- { p, -n, l, -j, h, -f, d, -b, a, -c, e, -g, i, -k, m, -o,}

where {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p} = { 90,89,87,83,81,77,72,66,62,56,49,41,33,25,17, 9}

**32-point DCT-8**

- { a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F,}
- { b, e, h, k, n, q, t, w, z, C, F, -E, -B, -y, -v, -s, -p, -m, -j, -g, -d, -a, -c, -f, -i, -l, -o, -r, -u, -x, -A, -D,}
- { c, h, m, r, w, B, 0, -B, -w, -r, -m, -h, -c, -c, -h, -m, -r, -w, -B, 0, B, w, r, m, h, c, c, h, m, r, w, B,}
- { d, k, r, y, F, -A, -t, -m, -f, -b, -i, -p, -w, -D, C, v, o, h, a, g, n, u, B, -E, -x, -q, -j, -c, -e, -l, -s, -z,}
- { e, n, w, F, -y, -p, -g, -c, -l, -u, -D, A, r, i, a, j, s, B, -C, -t, -k, -b, -h, -q, -z, E, v, m, d, f, o, x,}
- { f, q, B, -A, -p, -e, -g, -r, -C, z, o, d, h, s, D, -y, -n, -c, -i, -t, -E, x, m, b, j, u, F, -w, -l, -a, -k, -v,}

- { g, t, 0, -t, -g, -g, -t, 0, t, g, g, t, 0, -t, -g, -g, -t, 0, t, g, g, t, 0, -t, -g, -g, -t, 0, t, g, g, t,}
- { h, w, -B, -m, -c, -r, 0, r, c, m, B, -w, -h, -h, -w, B, m, c, r, 0, -r, -c, -m, -B, w, h, h, w, -B, -m, -c, -r,}
- { i, z, -w, -f, -l, -C, t, c, o, F, -q, -a, -r, E, n, d, u, -B, -k, -g, -x, y, h, j, A, -v, -e, -m, -D, s, b, p,}
- { j, C, -r, -b, -u, z, g, m, F, -o, -e, -x, w, d, p, -E, -l, -h, -A, t, a, s, -B, -i, -k, -D, q, c, v, -y, -f, -n,}
- { k, F, -m, -i, -D, o, g, B, -q, -e, -z, s, c, x, -u, -a, -v, w, b, t, -y, -d, -r, A, f, p, -C, -h, -n, E, j, l,}
- { l, -E, -h, -p, A, d, t, -w, -a, -x, s, e, B, -o, -i, -F, k, m, -D, -g, -q, z, c, u, -v, -b, -y, r, f, C, -n, -j,}
- { m, -B, -c, -w, r, h, 0, -h, -r, w, c, B, -m, -m, B, c, w, -r, -h, 0, h, r, -w, -c, -B, m, m, -B, -c, -w, r, h,}
- { n, -y, -c, -D, i, s, -t, -h, E, d, x, -o, -m, z, b, C, -j, -r, u, g, -F, -e, -w, p, l, -A, -a, -B, k, q, -v, -f,}
- { o, -v, -h, C, a, D, -g, -w, n, p, -u, -i, B, b, E, -f, -x, m, q, -t, -j, A, c, F, -e, -y, l, r, -s, -k, z, d,}
- { p, -s, -m, v, j, -y, -g, B, d, -E, -a, -F, c, C, -f, -z, i, w, -l, -t, o, q, -r, -n, u, k, -x, -h, A, e, -D, -b,}
- { q, -p, -r, o, s, -n, -t, m, u, -l, -v, k, w, -j, -x, i, y, -h, -z, g, A, -f, -B, e, C, -d, -D, c, E, -b, -F, a,}
- { r, -m, -w, h, B, -c, 0, c, -B, -h, w, m, -r, -r, m, w, -h, -B, c, 0, -c, B, h, -w, -m, r, r, -m, -w, h, B, -c,}
- { s, -j, -B, a, -C, -i, t, r, -k, -A, b, -D, -h, u, q, -l, -z, c, -E, -g, v, p, -m, -y, d, -F, -f, w, o, -n, -x, e,}
- { t, -g, 0, g, -t, -t, g, 0, -g, t, t, -g, 0, g, -t, -t, g, 0, -g, t, t, -g, 0, g, -t, -t, g, 0, -g, t, t, -g,}
- { u, -d, B, n, -k, -E, g, -r, -x, a, -y, -q, h, -F, -j, o, A, -c, v, t, -e, C, m, -l, -D, f, -s, -w, b, -z, -p, i,}
- { v, -a, w, u, -b, x, t, -c, y, s, -d, z, r, -e, A, q, -f, B, p, -g, C, o, -h, D, n, -i, E, m, -j, F, l, -k,}
- { w, -c, r, B, -h, m, 0, -m, h, -B, -r, c, -w, -w, c, -r, -B, h, -m, 0, m, -h, B, r, -c, w, w, -c, r, B, -h, m,}

{ x, -f, m, -E, -q, b, -t, -B, j, -i, A, u, -c, p, F, -n, e, -w, -y, g, -l, D, r, -a, s, C, -k, h, -z, -v, d, -o, }

{ y, -i, h, -x, -z, j, -g, w, A, -k, f, -v, -B, l, -e, u, C, -m, d, -t, -D, n, -c, s, E, -o, b, -r, -F, p, -a, q, }

{ z, -l, c, -q, E, u, -g, h, -v, -D, p, -b, m, -A, -y, k, -d, r, -F, -t, f, -i, w, C, -o, a, -n, B, x, -j, e, -s, }

{ A, -o, c, -j, v, F, -t, h, -e, q, -C, -y, m, -a, l, -x, -D, r, -f, g, -s, E, w, -k, b, -n, z, B, -p, d, -i, u, }

{ B, -r, h, -c, m, -w, 0, w, -m, c, -h, r, -B, -B, r, -h, c, -m, w, 0, -w, m, -c, h, -r, B, B, -r, h, -c, m, -w, }

{ C, -u, m, -e, d, -l, t, -B, -D, v, -n, f, -c, k, -s, A, E, -w, o, -g, b, -j, r, -z, -F, x, -p, h, -a, i, -q, y, }

{ D, -x, r, -l, f, -a, g, -m, s, -y, E, C, -w, q, -k, e, -b, h, -n, t, -z, F, B, -v, p, -j, d, -c, i, -o, u, -A, }

{ E, -A, w, -s, o, -k, g, -c, b, -f, j, -n, r, -v, z, -D, -F, B, -x, t, -p, l, -h, d, -a, e, -i, m, -q, u, -y, C, }

{ F, -D, B, -z, x, -v, t, -r, p, -n, l, -j, h, -f, d, -b, a, -c, e, -g, i, -k, m, -o, q, -s, u, -w, y, -A, C, -E, }

where {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F } = {90,90,89,88,88,86,85,84,82,80,78,77,74,72,68,66,63,60,56,53,50,45,42,38,34,30,26,21,17,13, 9, 4}

## WHAT IS CLAIMED IS:

1. A method for video decoding in a decoder, comprising:
  - decoding coding information of a coefficient block from a coded video bitstream, the coding information indicating a size of the coefficient block;
  - determining, based on the size of the coefficient block, an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed on transform coefficients of the coefficient block to obtain residual data of a residual block, wherein
    - when the size of the coefficient block satisfies a condition, the inverse vertical transform is performed after the inverse horizontal transform is performed on the transform coefficients of the coefficient block; and
    - reconstructing a sample in the residual block based on the residual data.
2. The method of claim 1, wherein the condition is that the size of the coefficient block is 32x64.
3. The method of claim 1, wherein the condition is that a height N of the coefficient block of MxN is larger than a width M of the coefficient block.
4. The method of claim 1, wherein
  - the size of the coefficient block is MxN, M and N being positive integers;
  - first residual data in a mxn region in the residual block are to be calculated by the inverse primary transform and second residual data outside the mxn region in the residual block are not to be calculated by the inverse primary transform, m being less than or equal to M, and n being less than or equal to N; and
  - the condition is that a ratio  $m/M$  is larger than or equal to a ratio  $n/N$ .
5. A method for video decoding in a decoder, comprising:
  - decoding coding information of a coefficient block from a coded video bitstream, the coding information indicating a size of the coefficient block;
  - determining, based on the size of the coefficient block, whether to reduce a number of calculations in one of inverse horizontal and inverse vertical transforms of an inverse primary

transform, the inverse vertical transform transforming transform coefficients of the coefficient block to intermediate data of an intermediate block, and the inverse horizontal transform transforming the intermediate data to residual data in a residual block;

performing the inverse primary transform, wherein:

when the number of calculations in the inverse vertical transform is determined to be reduced, top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero; and

when the number of calculations in the inverse horizontal transform is determined to be reduced, left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero; and

reconstructing a sample in the residual block based on the residual data.

6. The method of claim 5, wherein

the size of the coefficient block is  $32 \times 64$ ;

the one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform;

the determining includes determining that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is  $32 \times 64$ ; and

the performing the inverse primary transform includes performing the inverse primary transform, the top 16 rows of the intermediate data in the intermediate block being calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block being zero.

7. The method of claim 5, wherein

the size of the coefficient block is  $32 \times 64$ ;

the one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform;

the determining includes determining that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 64$ ; and

the performing the inverse primary transform includes performing the inverse primary transform, the left 16 columns of the residual data in the residual block being calculated by

the inverse horizontal transform and the remaining residual data in the residual block being zero.

8. The method of claim 5, wherein:

the size of the coefficient block is  $32 \times 32$ ;

the one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform;

the determining includes determining that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 32$ ; and

the performing the inverse primary transform includes performing the inverse primary transform, the left 16 columns of the residual data in the residual block being calculated by the inverse horizontal transform, the remaining residual data in the residual block being zero, and the intermediate data in the intermediate block being calculated by the inverse vertical transform.

9. The method of claim 5, wherein

the size of the coefficient block is  $32 \times 32$ ;

the one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform;

the determining includes determining that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is  $32 \times 32$ ; and

the performing the inverse primary transform includes performing the inverse primary transform, the top 16 rows of the intermediate data in the intermediate block being calculated by the inverse vertical transform, the remaining intermediate data in the intermediate block being zero, and the residual data in the residual block being calculated by the inverse horizontal transform.

10. An apparatus for video decoding, comprising processing circuitry configured to:

decode coding information of a coefficient block from a coded video bitstream, the coding information indicating a size of the coefficient block;

determine, based on the size of the coefficient block, an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed on transform coefficients of the coefficient block to obtain residual data of a residual block, wherein

when the size of the coefficient block satisfies a condition, perform the inverse vertical transform after the inverse horizontal transform is performed on the transform coefficients of the coefficient block; and

reconstruct a sample in the residual block based on the residual data.

11. The apparatus of claim 10, wherein the condition is that the size of the coefficient block is  $32 \times 64$ .

12. The apparatus of claim 10, wherein the condition is that a height  $N$  of the coefficient block of  $M \times N$  is larger than a width  $M$  of the coefficient block.

13. The apparatus of claim 10, wherein the size of the coefficient block is  $M \times N$ ,  $M$  and  $N$  being positive integers; first residual data in a  $m \times n$  region in the residual block are to be calculated by the inverse primary transform and second residual data outside the  $m \times n$  region in the residual block are not to be calculated by the inverse primary transform,  $m$  being less than or equal to  $M$ , and  $n$  being less than or equal to  $N$ ; and

the condition is that a ratio  $m/M$  is larger than or equal to a ratio  $n/N$ .

14. An apparatus for video decoding, comprising processing circuitry configured to: decode coding information of a coefficient block from a coded video bitstream, the coding information indicating a size of the coefficient block;

determine, based on the size of the coefficient block, whether to reduce a number of calculations in one of inverse horizontal and inverse vertical transforms of an inverse primary transform, the inverse vertical transform transforming transform coefficients of the coefficient block to intermediate data of an intermediate block, and the inverse horizontal transform transforming the intermediate data to residual data in a residual block;

perform the inverse primary transform, wherein:

when the number of calculations in the inverse vertical transform is determined to be reduced, top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero; and

when the number of calculations in the inverse horizontal transform is determined to be reduced, left 16 columns of the residual data in the residual block are

calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero; and

reconstruct a sample in the residual block based on the residual data.

15. The apparatus of claim 14, wherein  
the size of the coefficient block is  $32 \times 64$ ;  
the one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform; and

the processing circuitry is further configured to:

determine that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is  $32 \times 64$ ; and

perform the inverse primary transform, wherein

the top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero.

16. The apparatus of claim 14, wherein  
the size of the coefficient block is  $32 \times 64$ ;  
the one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform; and

the processing circuitry is further configured to:

determine that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 64$ ; and

perform the inverse primary transform, wherein

the left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero.

17. The apparatus of claim 14, wherein:  
the size of the coefficient block is  $32 \times 32$ ;  
the one of the inverse horizontal and inverse vertical transforms is the inverse horizontal transform; and

the processing circuitry is further configured to:

determine that the number of calculations in the inverse horizontal transform is to be reduced when the size of the coefficient block is  $32 \times 32$ ; and

perform the inverse primary transform, wherein

the left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform, the remaining residual data in the residual block are zero, and the intermediate data in the intermediate block are calculated by the inverse vertical transform.

18. The apparatus of claim 14, wherein

the size of the coefficient block is  $32 \times 32$ ;

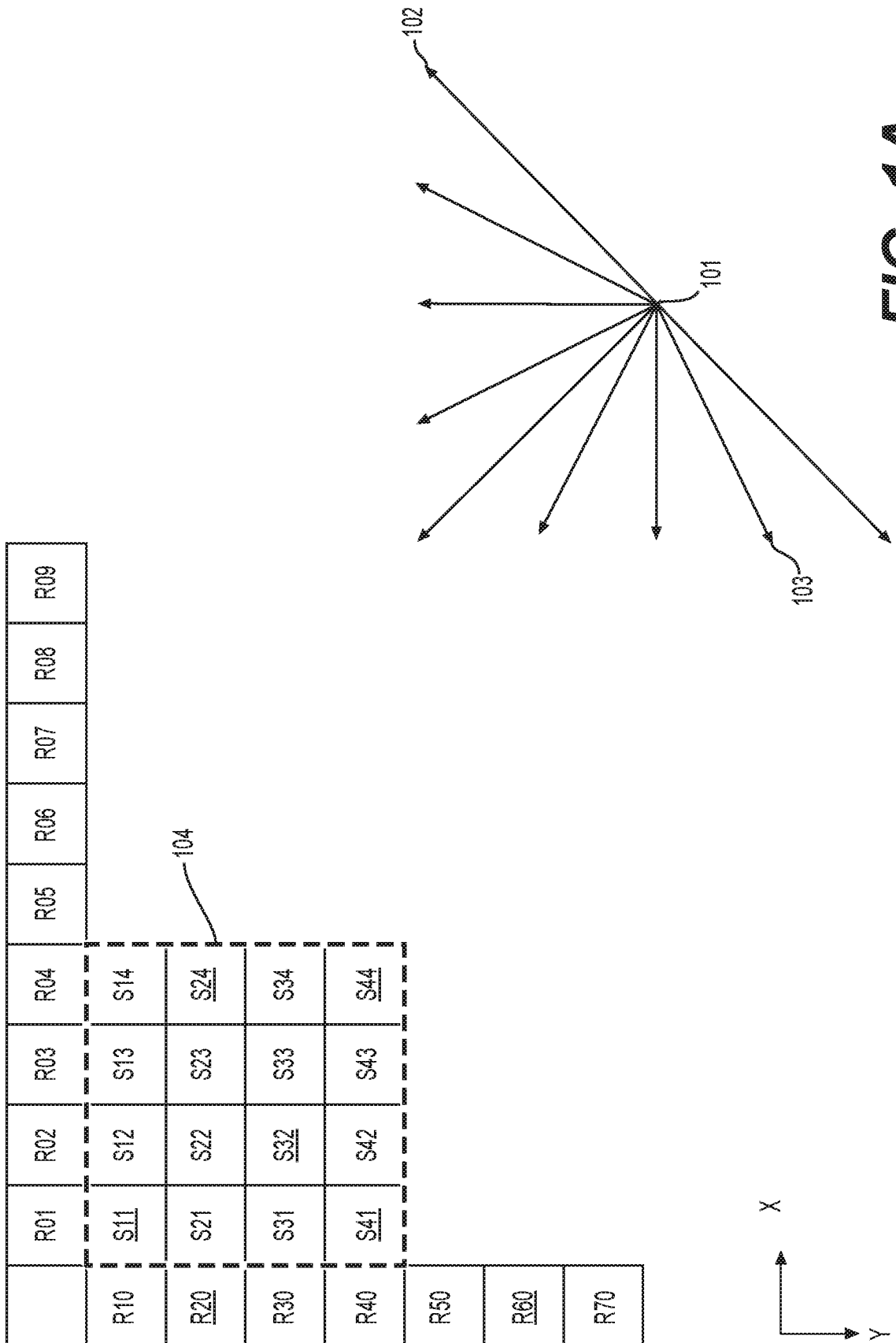
the one of the inverse horizontal and inverse vertical transforms is the inverse vertical transform; and

the processing circuitry is further configured to:

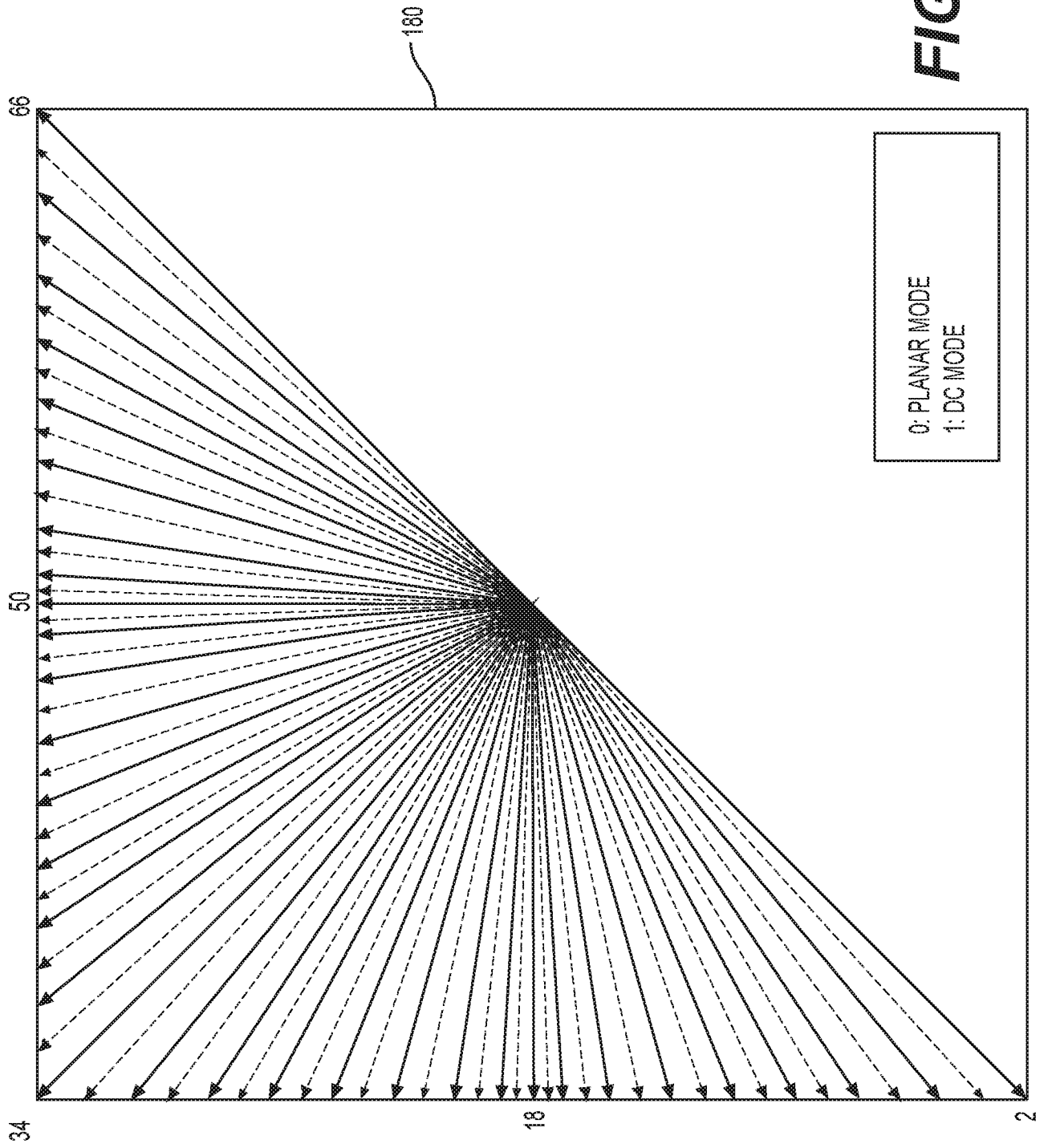
determine that the number of calculations in the inverse vertical transform is to be reduced when the size of the coefficient block is  $32 \times 32$ ; and

perform the inverse primary transform, wherein

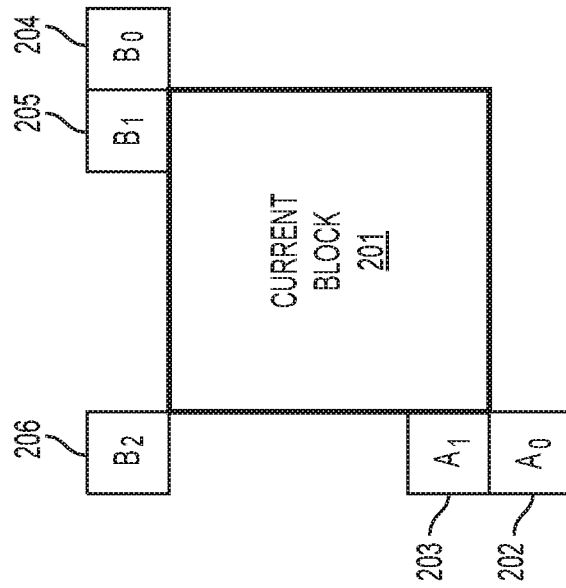
the top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform, the remaining intermediate data in the intermediate block are zero, and the residual data in the residual block are calculated by the inverse horizontal transform.



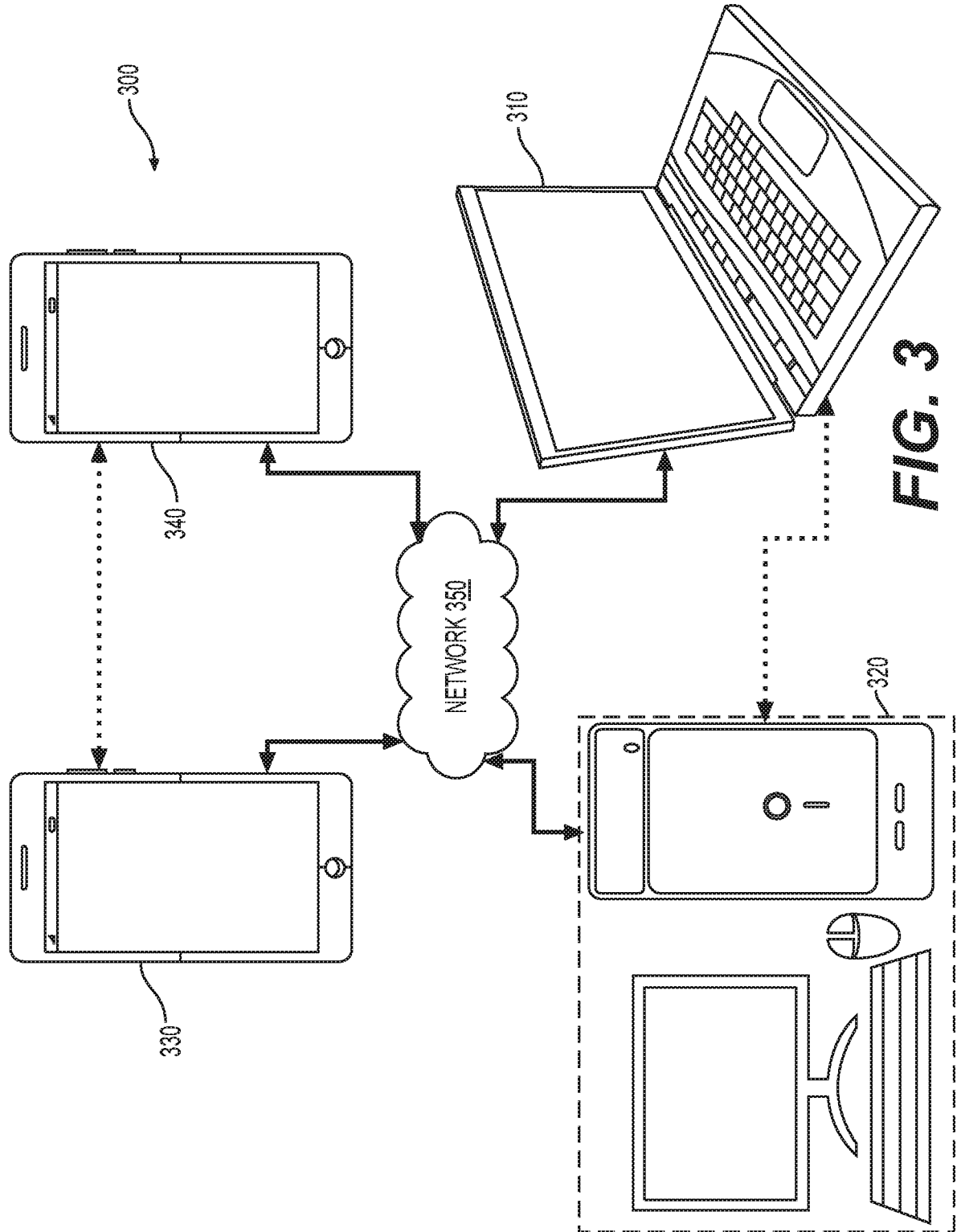
**FIG. 1A**



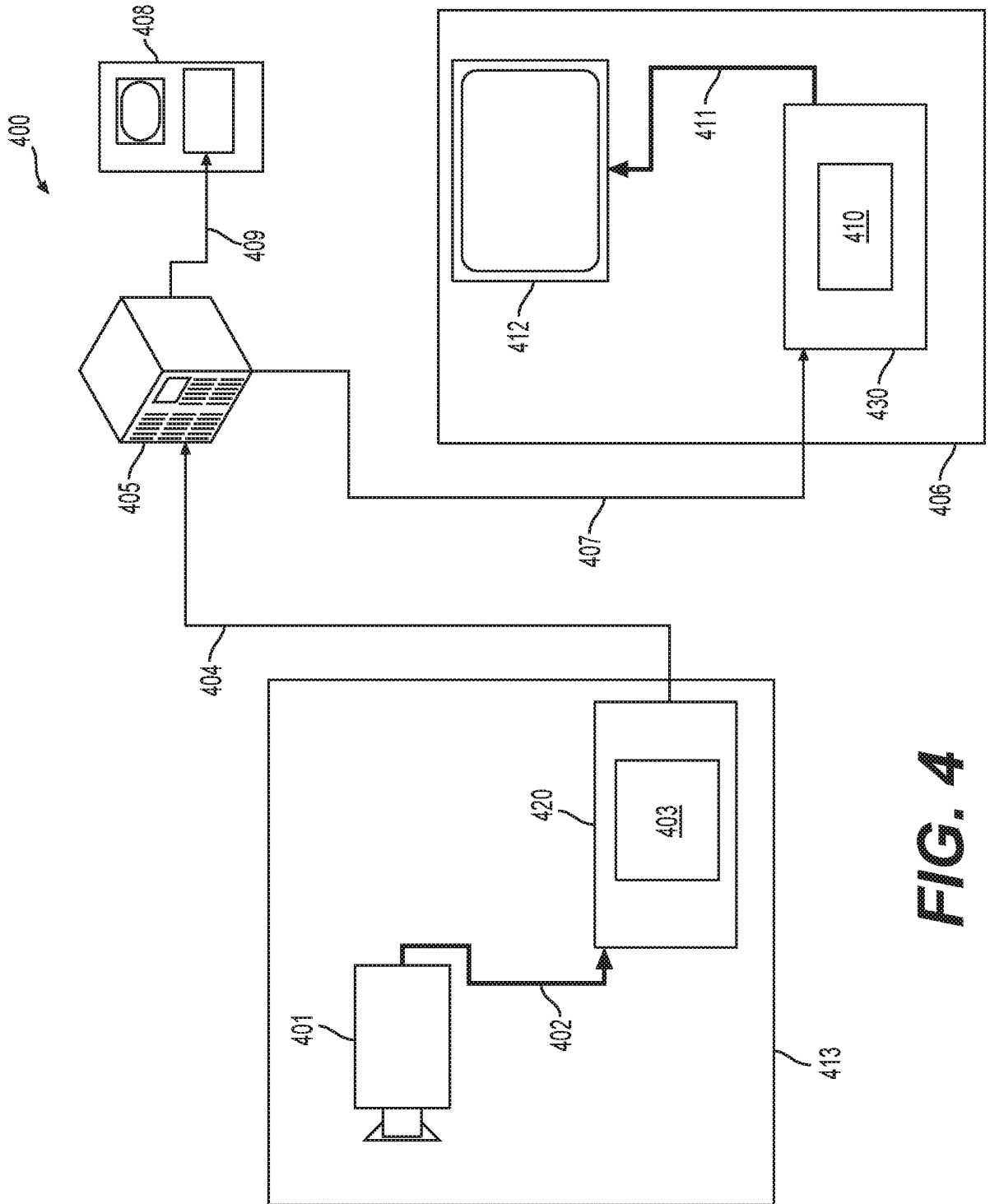
**FIG. 1B**



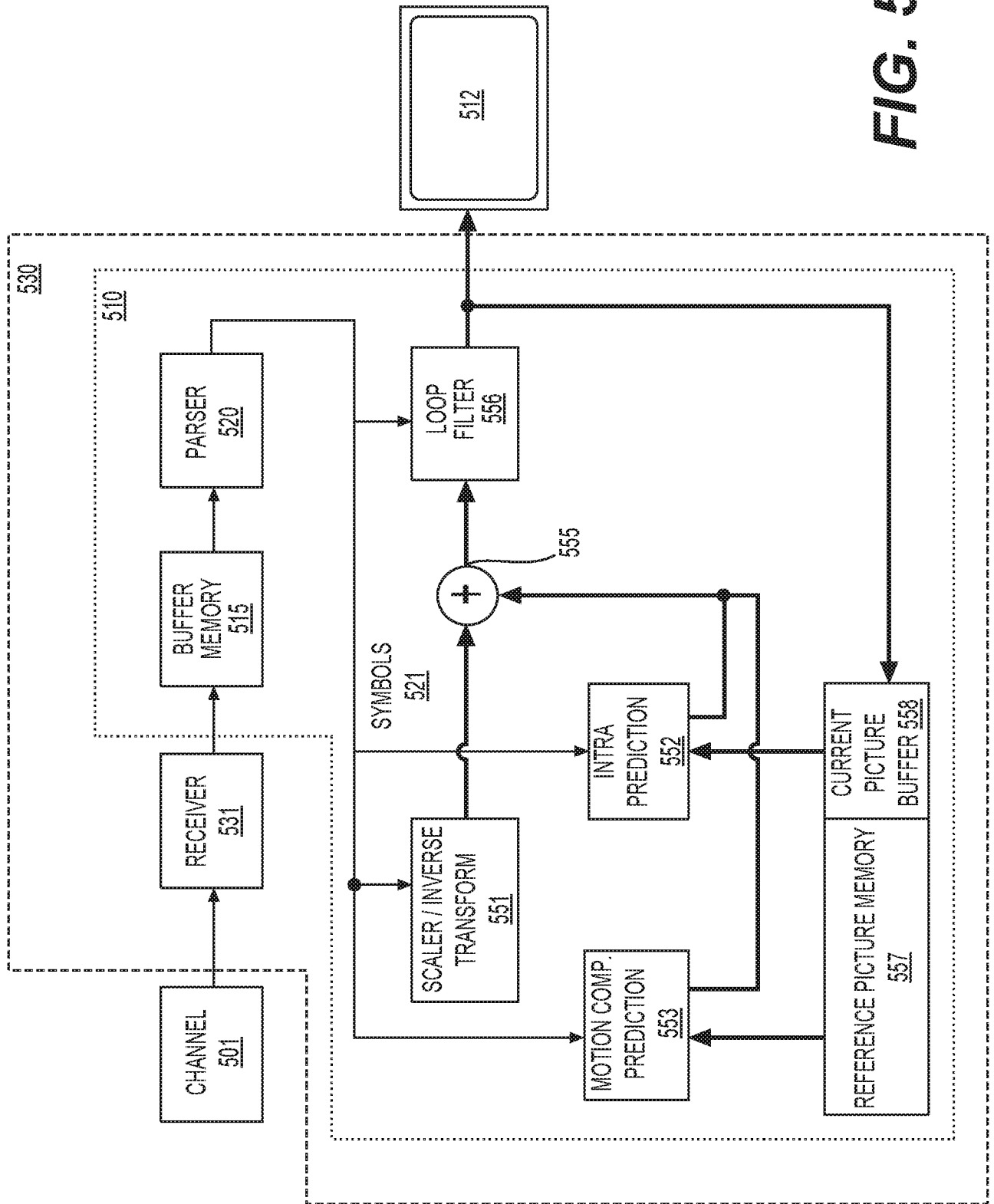
**FIG. 2**  
*(Related Art)*



**FIG. 3**



**FIG. 4**



**FIG. 5**

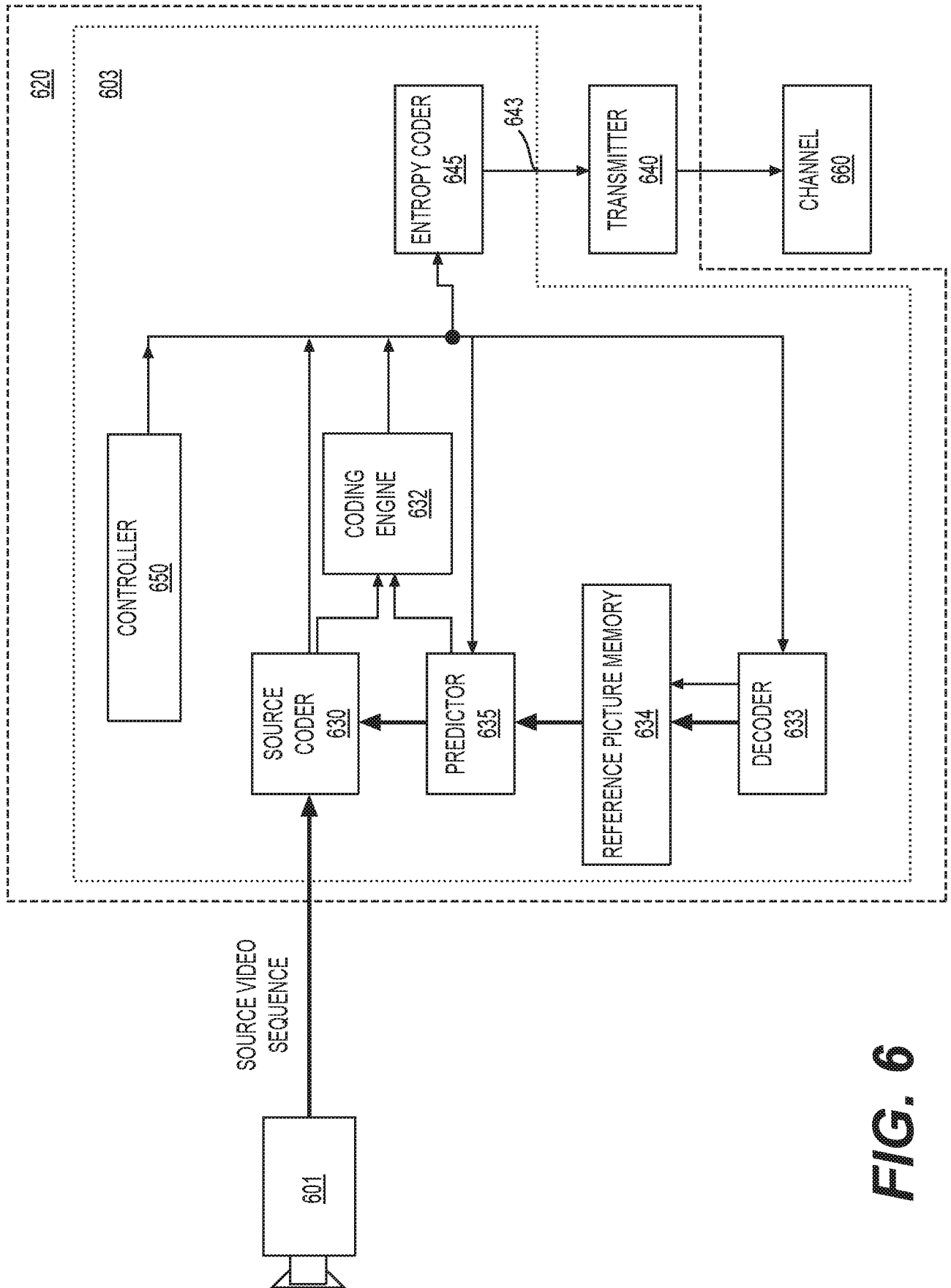
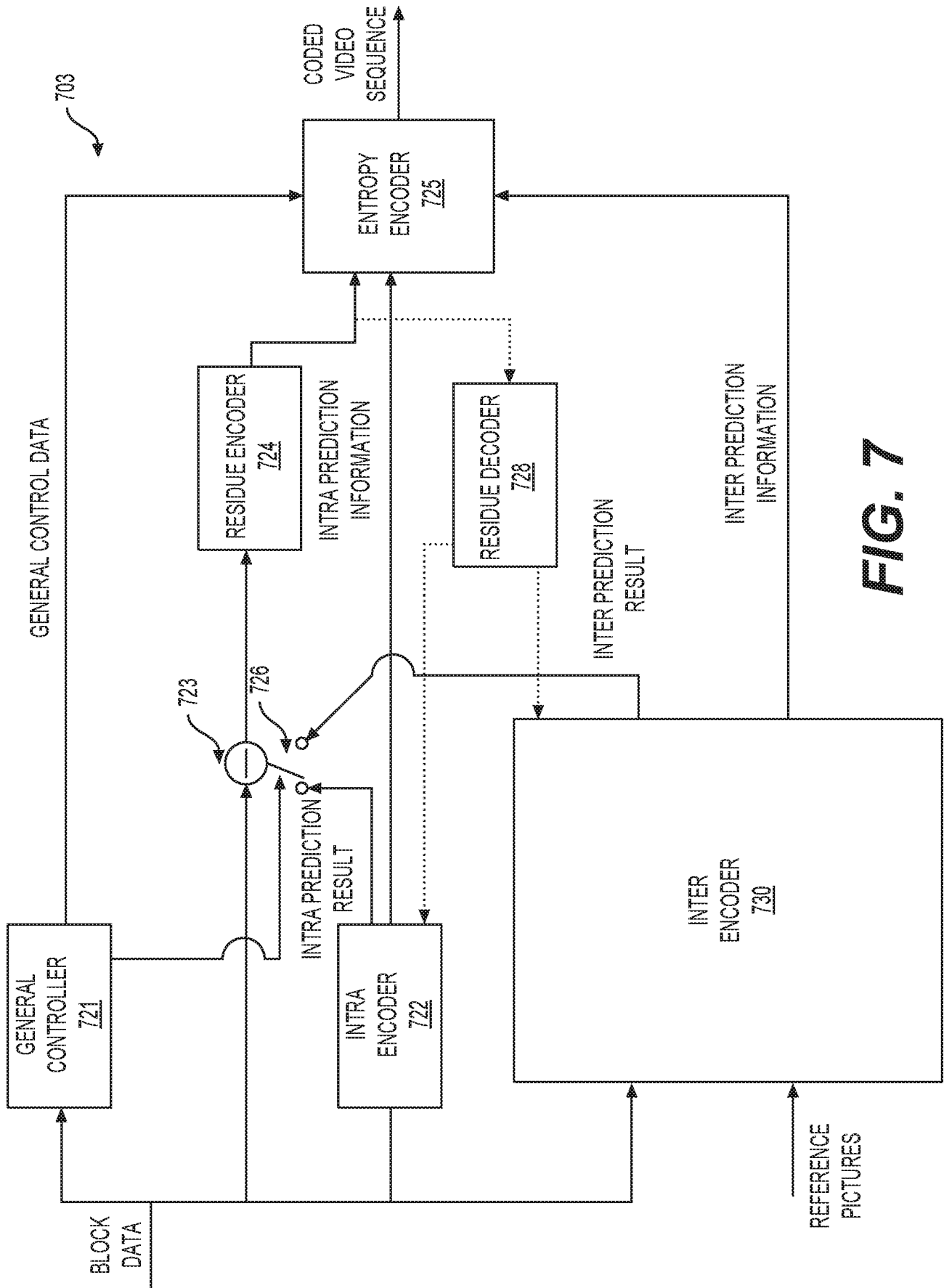
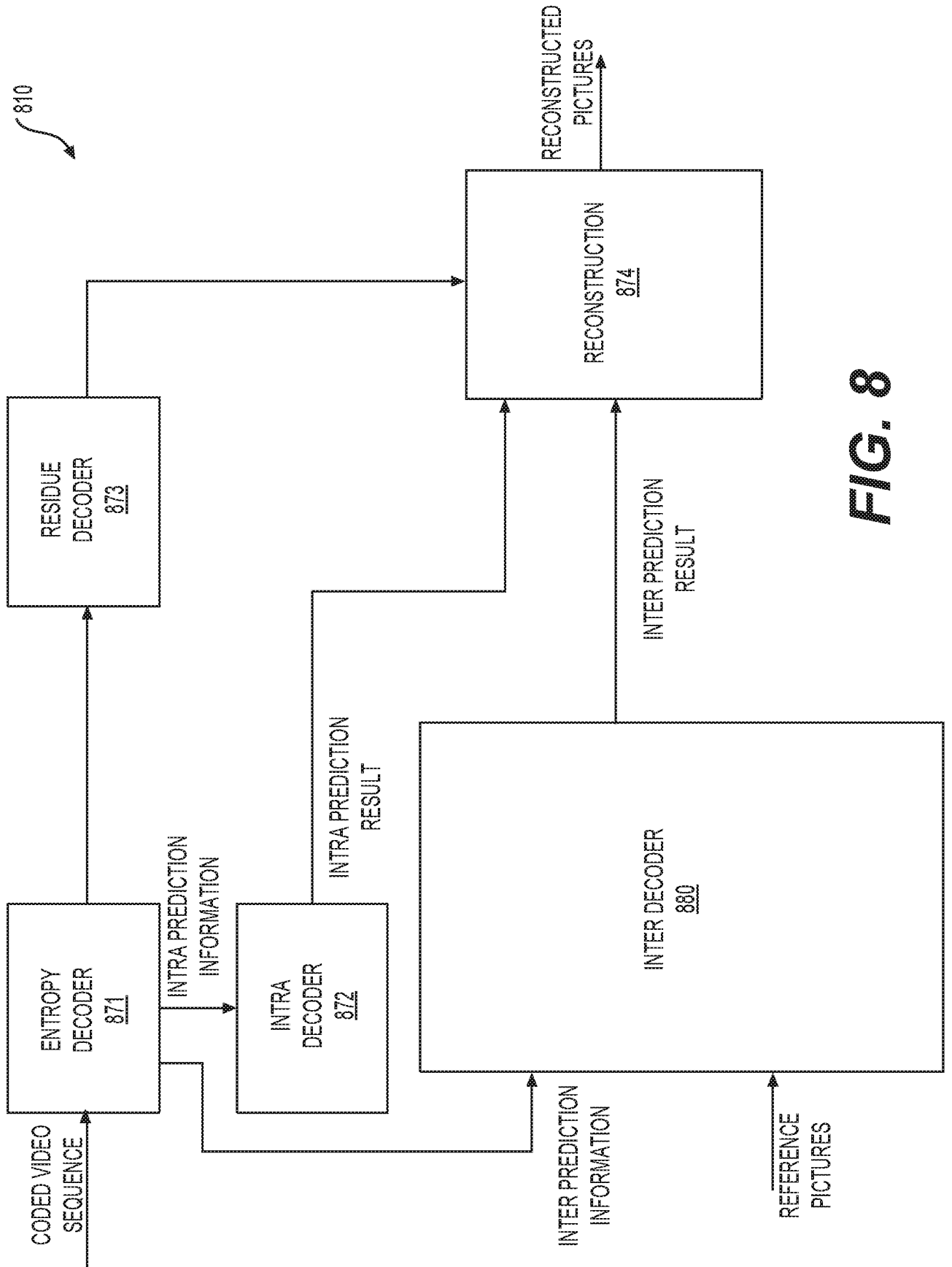


FIG. 6



**FIG. 7**



**FIG. 8**

	<b>Descriptor</b>
transform_unit(x0, y0, tbWidth, tbHeight, treeType) {	
if( treeType == SINGLE_TREE    treeType == DUAL_TREE_LUMA )	
tu_cbf_luma[x0][y0]	ae(v)
if( treeType == SINGLE_TREE    treeType == DUAL_TREE_CHROMA ) {	
tu_cbf_cb[x0][y0]	ae(v)
tu_cbf_cr[x0][y0]	ae(v)
}	
if(	
(( CuPredMode[x0][y0] == MODE_INTRA) && sps_mts_intra_enabled_flag)	
(( CuPredMode[x0][y0] == MODE_INTER) && sps_mts_inter_enabled_flag) )	
&& tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA	
&& ( tbWidth <= 32 ) && ( tbHeight <= 32 ) )	
cu_mts_flag[x0][y0]	ae(v)
if( tu_cbf_luma[x0][y0] )	
residual_coding(x0, y0, log2( tbWidth ), log2( tbHeight ), 0 )	
if( tu_cbf_cb[x0][y0] )	
residual_coding(x0, y0, log2( tbWidth/2 ), log2( tbHeight/2 ), 1 )	
if( tu_cbf_cr[x0][y0] )	
residual_coding(x0, y0, log2( tbWidth/2 ), log2( tbHeight/2 ), 2 )	
}	

910

**FIG. 9**

	<b>Descriptor</b>
<pre>residual_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) {   if( transform_skip_enabled_flag &amp;&amp; ( cIdx != 0    cu_mts_flag[ x0 ][ y0 ] == 0 ) &amp;&amp;       ( log2TbWidth &lt;= 2 ) &amp;&amp; ( log2TbHeight &lt;= 2 ) )     transform_skip_flag[ x0 ][ y0 ][ cIdx ]   last_sig_coeff_x_prefix   last_sig_coeff_y_prefix   if( last_sig_coeff_x_prefix &gt; 3 )     last_sig_coeff_x_suffix   if( last_sig_coeff_y_prefix &gt; 3 )     last_sig_coeff_y_suffix   .....</pre>	
	ae(v)
	ae(v)
	ae(v)
	ae(v)
	ae(v)

**FIG. 10A**

Continue from FIG. 10A

```

if( dep_quant_enabled_flag ) {
    QState = startQStateSb
    for( n = numSbCoeff - 1; n >= 0; n-- ) {
        xC = ( xS << log2SbSize ) +
            DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]
        yC = ( yS << log2SbSize ) +
            DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]
        if( sig_coeff_flag[ xC ][ yC ] )
            TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =
                ( 2 * AbsLevel[ xC ][ yC ] - ( QState > 1 ? 1 : 0 ) ) *
                ( 1 - 2 * coeff_sign_flag[ n ] )
        QState = QStateTransTable[ QState ][ par_level_flag[ n ] ]
    } else {
        sumAbsLevel = 0
        for( n = numSbCoeff - 1; n >= 0; n-- ) {
            xC = ( xS << log2SbSize ) +
                DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 0 ]
            yC = ( yS << log2SbSize ) +
                DiagScanOrder[ log2SbSize ][ log2SbSize ][ n ][ 1 ]
            if( sig_coeff_flag[ xC ][ yC ] ) {

```

**FIG. 10B**

Continue from FIG. 10B

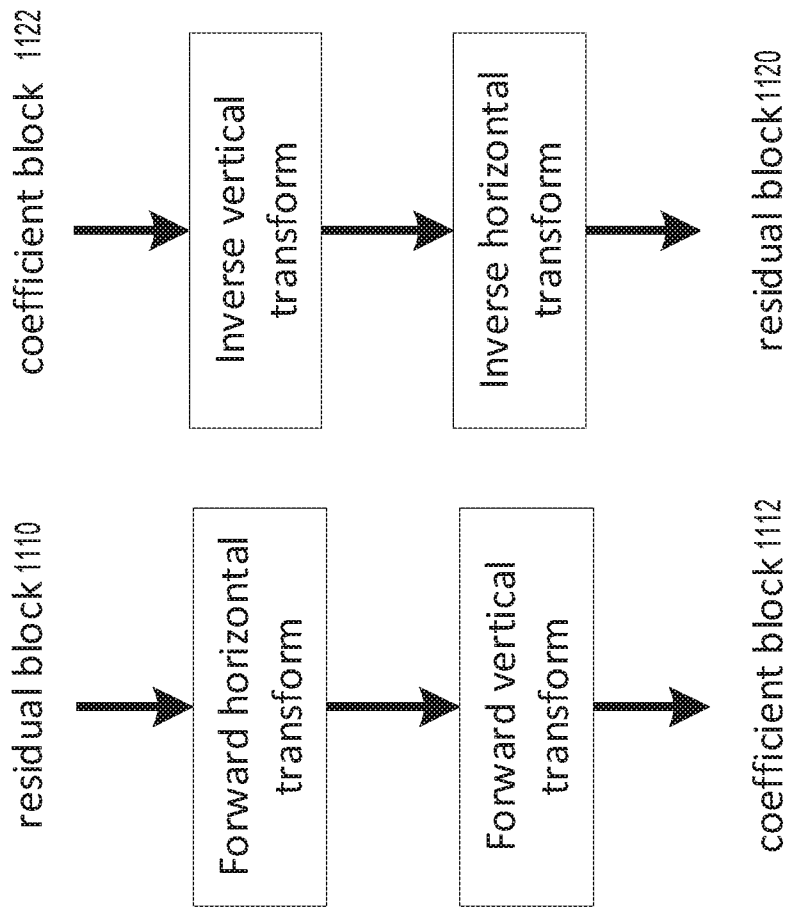
```

TransCoeffLevel[x0][y0][cIdx][xC][yC] =
  AbsLevel[xC][yC] * (1 - 2 * coeff_sign_flag[n])
if( signHidden ) {
  sumAbsLevel += AbsLevel[kC][yC]
  if( n == firstSigScanPosSb ) && ( sumAbsLevel%2 == 1 )
    TransCoeffLevel[x0][y0][cIdx][xC][yC] =
      - TransCoeffLevel[x0][y0][cIdx][xC][yC]
}
}
}
}
}
if( cu_mts_flag[x0][y0] && ( cIdx == 0 ) &&
! transform_skip_flag[x0][y0][cIdx] &&
( ( CuPredMode[x0][y0] == MODE_INTRA && numSigCoeff > 2 ) ||
( CuPredMode[x0][y0] == MODE_INTER ) ) {
  mts_idx[x0][y0]
}
ac(v)
}

```

1010

FIG. 10C



**FIG. 11A** **FIG. 11B**

Inputs to this process are:

- a luma location (  $xTbY, yTbY$  ) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable  $nTbW$  specifying the width of the current transform block,
- a variable  $nTbH$  specifying the height of the current transform block,
- a variable  $cIdx$  specifying the colour component of the current block,
- an  $(nTbW) \times (nTbH)$  array  $d[x][y]$  of scaled transform coefficients with  $x = 0..nTbW - 1, y = 0..nTbH - 1$ .

Output of this process is the  $(nTbW) \times (nTbH)$  array  $r[x][y]$  of residual samples with  $x = 0..nTbW - 1, y = 0..nTbH - 1$ .

**FIG. 12A**

*Continue from FIG. 12A*

The variable `implicitMtsEnabled` is derived as follows:

- If `sps_mts_enabled_flag` is equal to 1 and one of the following conditions is true, `implicitMtsEnabled` is set equal to 1:
  - `IntraSubPartitionsSplitType` is not equal to `ISP_NO_SPLIT`
  - `cu_sbt_flag` is equal to 1 and `Max( nTbW, nTbH )` is less than or equal to 32
  - `sps_explicit_mts_intra_enabled_flag` and `sps_explicit_mts_inter_enabled_flag` are both equal to 0 and `CuPredMode[ xTbY ][ yTbY ]` is equal to `MODE_INTRA`

Otherwise, `implicitMtsEnabled` is set equal to 0.

**FIG. 12B**

**Continue from FIG. 12B**

The variable trTypeHor specifying the horizontal transform kernel and the variable trTypeVer specifying the vertical transform kernel are derived as follows:

- If cIdx is greater than 0, trTypeHor and trTypeVer are set equal to 0.
- Otherwise, if implicitMisEnabled is equal to 1, the following applies:
  - If IntraSubPartitionsSplitType is not equal to ISP\_NO\_SPLIT, trTypeHor and trTypeVer are specified depending on intraPredMode.
  - Otherwise, if cu\_sbt\_flag is equal to 1, trTypeHor and trTypeVer are specified depending on cu\_sbt\_horizontal\_flag and cu\_sbt\_pos\_flag.
  - Otherwise (sps\_explicit\_mts\_intra\_enabled\_flag and sps\_explicit\_mts\_inter\_enabled\_flag are equal to 0), trTypeHor and trTypeVer are derived as follows:

$$\text{trTypeHor} = ( \text{nTbW} >= 4 \ \&\& \ \text{nTbW} <= 16 \ \&\& \ \text{nTbW} <= \text{nTbH} ) ? 1 : 0$$

$$\text{trTypeVer} = ( \text{nTbH} >= 4 \ \&\& \ \text{nTbH} <= 16 \ \&\& \ \text{nTbH} <= \text{nTbW} ) ? 1 : 0$$
**FIG. 12C**

*Continue from FIG. 12C*

- Otherwise, trTypeHor and trTypeVer are specified depending on tu\_mts\_idx[ xTbY ][ yTbY ].

The variables nonZeroW and nonZeroH are derived as follows:

$$\text{nonZeroW} = \text{Min}( \text{nTbW}, ( \text{trTypeHor} > 0 ) ? 16 : 32 )$$

$$\text{nonZeroH} = \text{Min}( \text{nTbH}, ( \text{trTypeVer} > 0 ) ? 16 : 32 )$$

**FIG. 12D**

Continue from FIG. 12D

1210

The (nTbW)x(nTbH) array r of residual samples is derived as follows:

1. When nTbH is greater than 1, each (vertical) column of scaled transform coefficients  $d[x][y]$  with  $x = 0..nonZeroW-1, y = 0..nonZeroH-1$  is transformed to  $e[x][y]$  with  $x = 0..nonZeroW-1, y = 0..nTbH-1$  by invoking the one dimensional transformation process for each column  $x.nonZeroW-1$  with the height of the transform block nTbH, the non-zero height of the scaled transform coefficients nonZeroH, the list  $d[x][y]$  with  $y = 0..nonZeroH-1$  and the transform type variable trType set equal to trTypeVer as inputs, and the output is the list  $e[x][y]$  with  $y = 0..nTbH-1$ .

2. When nTbH and nTbW are both greater than 1, the intermediate sample values  $g[x][y]$  with  $x = 0..nonZeroW-1, y = 0..nTbH-1$  are derived as follows:

$$g[x][y] = Clip3(CoeffMin, CoeffMax, (e[x][y] + 64) >> 7) \quad 1211$$

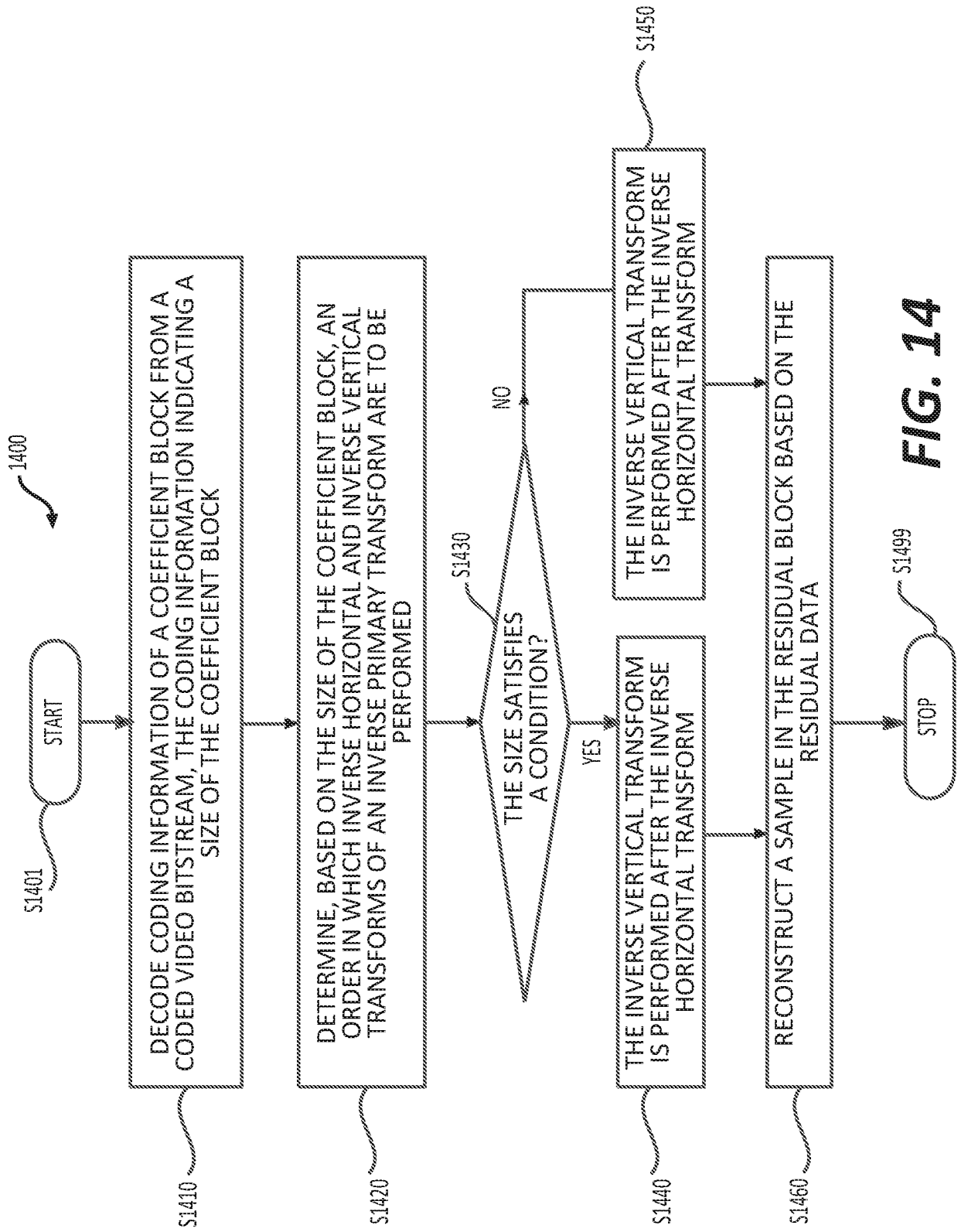
3. When nTbW is greater than 1, each (horizontal) row of the resulting array  $g[x][y]$  with  $x = 0..nonZeroW-1, y = 0..nTbH-1$  is transformed to  $r[x][y]$  with  $x = 0..nTbW-1, y = 0..nTbH-1$  by invoking the one-dimensional transformation process for each row  $y = 0..nTbH-1$  with the width of the transform block nTbW, the non-zero width of the resulting array  $g[x][y]$  nonZeroW, the list  $g[x][y]$  with  $x = 0..nonZeroW-1$  and the transform type variable trType set equal to trTypeHor as inputs, and the output is the list  $r[x][y]$  with  $x = 0..nTbW-1$ .

FIG. 12E

Descriptor	
	<pre> residual_coding( x0, y0, log2TbWidth, log2TbHeight, cIdx ) {   if( transform_skip_enabled_flag &amp;&amp; ( cIdx != 0    tu_mts_flag[x0][y0] == 0 ) &amp;&amp;     ( log2TbWidth &lt;= 2 ) &amp;&amp; ( log2TbHeight &lt;= 2 ) )     transform_skip_flag[x0][y0][ cIdx ]   last_sig_coeff_x_prefix   last_sig_coeff_y_prefix   if( last_sig_coeff_x_prefix &gt; 3 )     last_sig_coeff_x_suffix   if( last_sig_coeff_y_prefix &gt; 3 )     last_sig_coeff_y_suffix   .....   for( i = lastSubBlock; i &gt;= 0; i-- ) {     .....     if( ( i &lt; lastSubBlock ) &amp;&amp; ( i &gt; 0 ) ) {       if( ( ( tu_mts_flag[x0][y0] == 0    cIdx != 0 ) &amp;&amp; ( xS &lt;&lt; log2SbSize ) &lt; 32 &amp;&amp;         ( yS &lt;&lt; log2SbSize ) &lt; 32 )    ( tu_mts_flag[x0][y0] == 1 &amp;&amp;         ( xS &lt;&lt; log2SbSize ) &lt; 16 &amp;&amp; ( yS &lt;&lt; log2SbSize ) &lt; 16 ) )         coded_sub_block_flag[xS][yS]         inferSbDcSigCoeffFlag = 1       }       .....     }     if( tu_mts_flag[x0][y0] &amp;&amp; ( cIdx == 0 ) )       mts_idx[x0][y0][ cIdx ]   } </pre>
ae(v)	
ae(v)	
ae(v)	
ae(v)	
ae(v)	
ae(v)	

1310

FIG. 13



**FIG. 14**

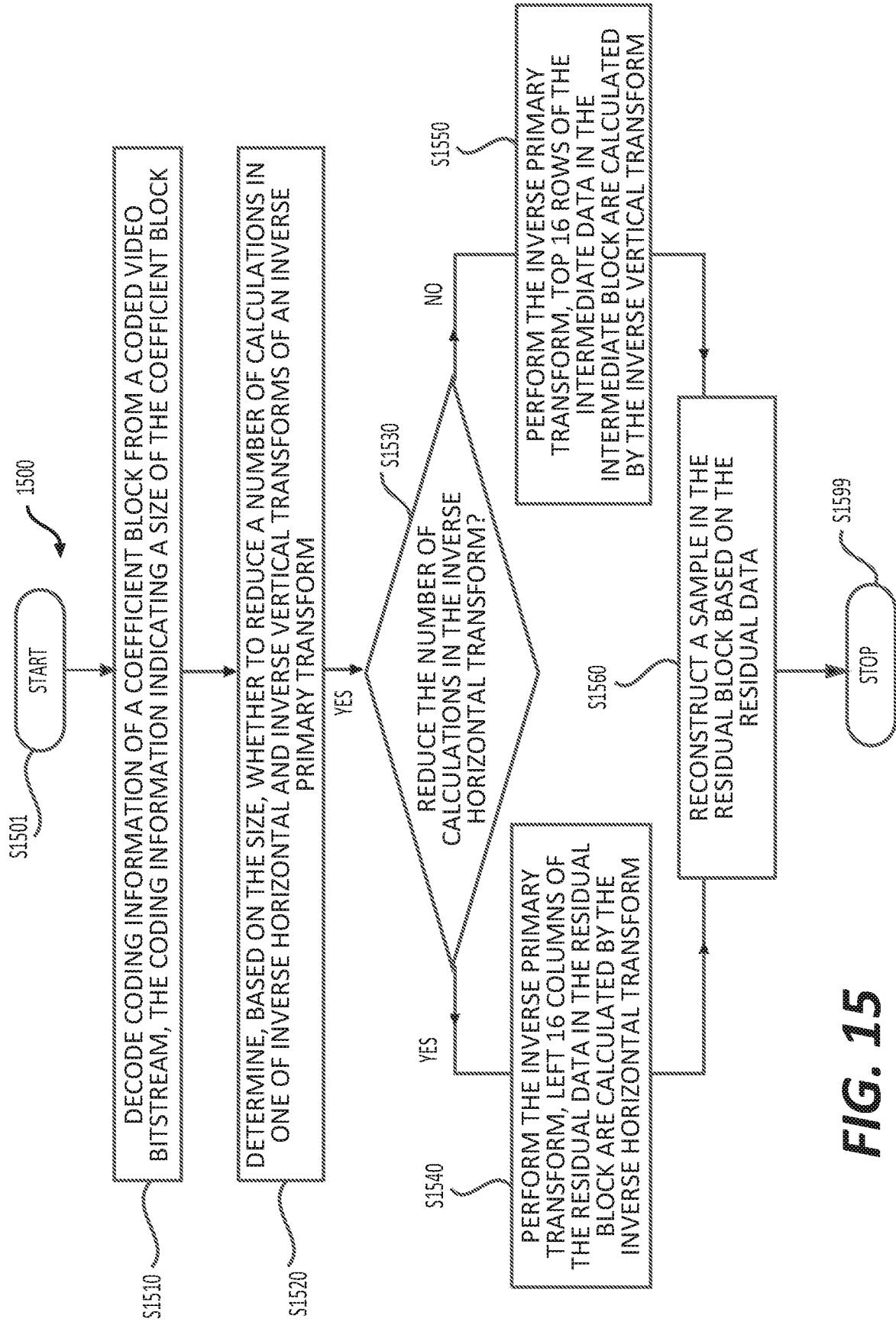
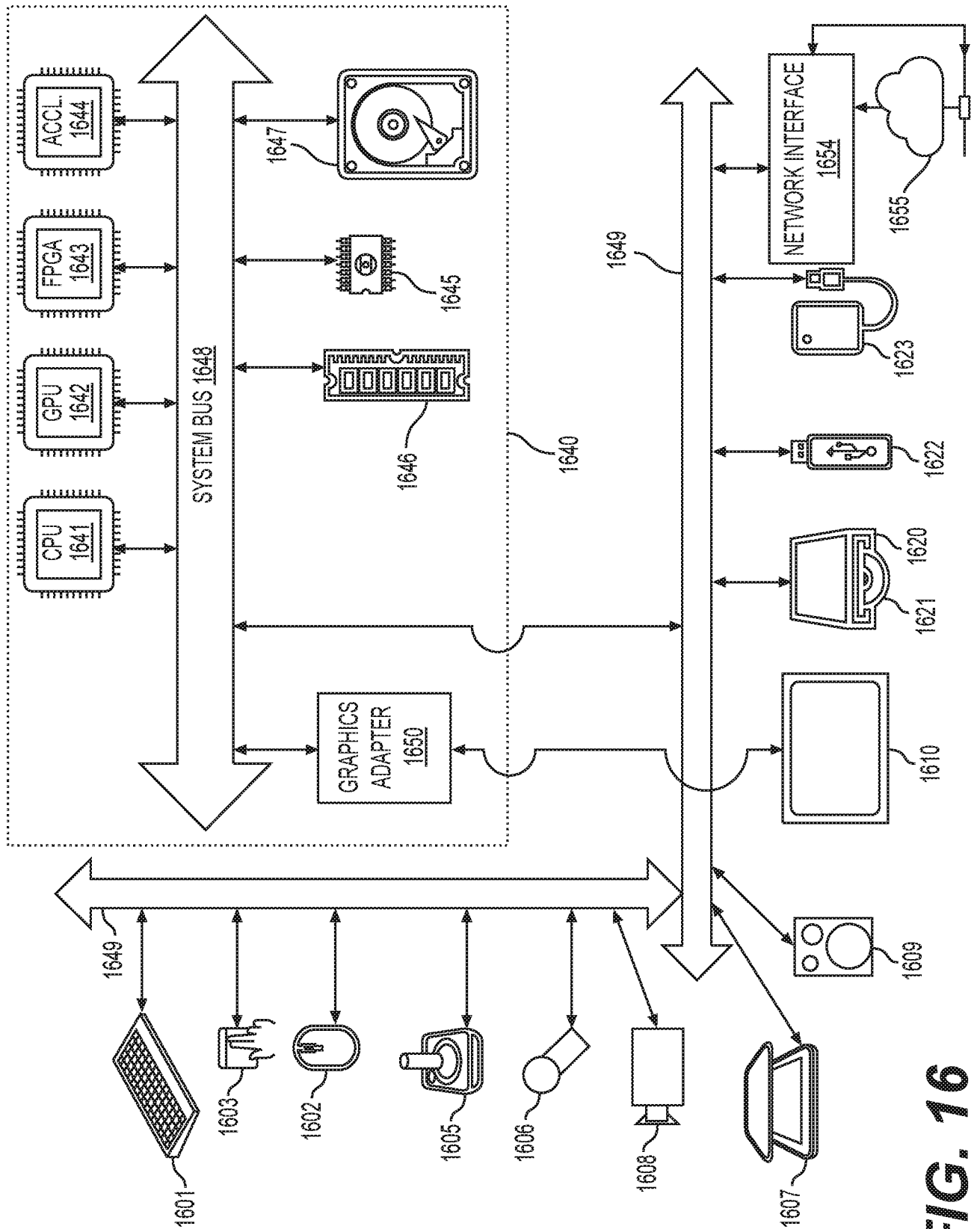


FIG. 15



**FIG. 16**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US20/21659

A. CLASSIFICATION OF SUBJECT MATTER

IPC - H04N 19/12, 19/122, 19/176, 19/132, 19/44, 19/102, 19/88, 19/166, 19/00 (2020.01)

CPC - H04N 19/12, 19/122, 19/176, 19/132, 19/44, 19/102, 19/88, 19/166, 19/00

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	US 2013/0003856 A1 (SAXENA, A et al.) 03 January 2013; figures 1 & 5, paragraphs [0039]-[0040], [0080], [0132], [0162]-[0167], [0172]-[0175], [0248], claims 21, 29, 31, 39	1, 3-4, 10, & 12-13
---		---
Y		2 & 11
Y	US 2017/0237985 A1 (SAMSUNG ELECTRONICS CO., LTD.) 17 August 2017; figure 12, paragraphs [0140]-[0143] & [0219]	2 & 11
A	US 2018/0020218 A1 (QUALCOMM INCORPORATED) 18 January 2018; see entire document	1-4 & 10-13
A	US 2018/0192056 A1 (DOLBY INTERNATIONAL AB) 05 July 2018; see entire document	1-4 & 10-13
A	US 2017/0019686 A1 (MEDIATEK INC.) 19 January 2017; see entire document	1-4 & 10-13

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

07 May 2020 (07.05.2020)

Date of mailing of the international search report

16 JUL 2020

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

Shane Thomas

Telephone No. PCT Helpdesk: 571-272-4300

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US20/21659

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

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

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

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

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

See extra sheet.

- 1.  As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
- 2.  As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
- 3.  As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
  
- 4.  No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:  
1-4 & 10-13

**Remark on Protest**

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

## INTERNATIONAL SEARCH REPORT

International application No.

PCT/US20/21659

\*\*\*-Continued from Box III: Observations where Unity of Invention is Lacking.\*\*\*-

This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fee must be paid.

Group I: Claims 1-4 & 10-13 are directed towards a method and apparatus for video decoding comprising: determining an order in which inverse transforms are to be performed.

Group II: Claims 5-9 & 14-18 are directed towards a method and apparatus for video decoding comprising: determining whether to reduce a number of calculations.

The inventions listed as Groups I-II do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons:

The special technical features of Group I include at least determining, based on the size of the coefficient block, an order in which inverse horizontal and inverse vertical transforms of an inverse primary transform are to be performed on transform coefficients of the coefficient block, wherein when the size of the coefficient block satisfies a condition, the inverse vertical transform is performed after the inverse horizontal transform is performed on the transform coefficients of the coefficient block, which are not present in Group II.

The special technical features of Group II include at least determining, based on the size of the coefficient block, whether to reduce a number of calculations in one of inverse horizontal and inverse vertical transforms of an inverse primary transform, the inverse vertical transform transforming transform coefficients of the coefficient block to intermediate data of an intermediate block, the inverse horizontal transform transforming the intermediate data to residual data in a residual block; wherein: when the number of calculations in the inverse vertical transform is determined to be reduced, top 16 rows of the intermediate data in the intermediate block are calculated by the inverse vertical transform and the remaining intermediate data in the intermediate block are zero; and when the number of calculations in the inverse horizontal transform is determined to be reduced, left 16 columns of the residual data in the residual block are calculated by the inverse horizontal transform and the remaining residual data in the residual block are zero, which are not present in Group I.

The common technical features shared by Groups I-II are a method or apparatus for video decoding, comprising: decoding coding information of a coefficient block from a coded video bitstream, the coding information indicating a size of the coefficient block; determining, based on the size of the coefficient block, information related to inverse horizontal and inverse vertical transforms of an inverse primary transform to be performed based on transform coefficients of the coefficient block to obtain residual data of a residual block; performing the inverse primary transform; and reconstructing a sample in the residual block based on the residual data.

However, these common features are previously disclosed by US 2018/0020218 A1 to QUALCOMM INCORPORATED (hereinafter 'QUALCOMM'). QUALCOMM discloses a method or apparatus for video decoding (a method performed by a video decoder 30 for decoding video; paragraph [0203], claim 1 of QUALCOMM), comprising: decoding coding information of a coefficient block from a coded video bitstream decoding a coded information, such as an EMT index or a prediction mode for a current coefficient block from an encoded video bitstream; FIG. 8, paragraphs [0203], [0210], claim 1 of QUALCOMM), the coding information indicating a size of the coefficient block (the coded information indicates a coefficient block width and height; paragraph [0210], claims 1, 9, & 10 of QUALCOMM); determining, based on the size of the coefficient block, information related to inverse horizontal and inverse vertical transforms of an inverse primary transform (determining, based on the coefficient block width and height, a threshold for selecting a horizontal and vertical transform pair of an inverse transform; paragraphs [0209]-[0211], claims 1, 9, & 10 of QUALCOMM) to be performed based on transform coefficients of the coefficient block to obtain residual data of a residual block (the inverse transform is to be applied to transform coefficients of the coefficient block to generate residual values for a residual block; paragraphs [0028], [0208]-[0210]); performing the inverse primary transform (performing the inverse transform; paragraph [0213]); and reconstructing a sample in the residual block based on the residual data (reconstructing samples of a video block based on the residual values; paragraphs [0028], [0209], [0213]-[0214]).

Since the common technical features are previously disclosed by the QUALCOMM reference, these common features are not special and so Groups I-II lack unity.